



ENOVIA DesignSync Administrator's Guide

3DEXPERIENCE 2022

Overview of DesignSync Administration

©2021 Dassault Systèmes. All rights reserved. 3DEXPERIENCE®, the Compass icon, the 3DS logo, CATIA, SOLIDWORKS, ENOVIA, DELMIA, SIMULIA, GEOVIA, EXALEAD, 3D VIA, BIOVIA, NETVIBES, IFWE and 3DEXCITE are commercial trademarks or registered trademarks of Dassault Systèmes, a French "société européenne" (Versailles Commercial Register # B 322 306 440), or its subsidiaries in the United States and/or other countries. All other trademarks are owned by their respective owners. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

**DASSAULT
SYSTEMES**

Table of Contents

Overview of DesignSync Administration.....	1
The syncmgr Account.....	1
DesignSync and SyncAdmin	1
SyncAdmin.....	1
DesignSync web administration	2
System Administration Tasks	2
Using ENOVIA Synchronicity DesignSync Data Manager Administrator's Guide Documentation	5
Before Reading this Guide.....	6
SyncServer Architecture.....	7
Overview of Enterprise Design Development.....	9
Prerequisites.....	9
Setup and Maintenance for Enterprise Developments.....	9
See Also	9
Getting Started	11
Registry Files	11
Executing SyncAdmin from UNIX	14
Executing SyncAdmin from Windows	16
Installation	19
Installing DesignSync Tools.....	19
ENOVIA Synchronicity DesignSync Data Manager Installation	19
Where to Find More Information	21
Downloading Updated Installation Instructions and Release-Specific Information ..	22

- Contacting ENOVIA 22
- Client Installation 23
 - Preparing for Installation 23
 - Running the GUI Client Install Script..... 27
 - Running the Command Line Client Install Script..... 29
 - Scripting DesignSync Client Installation..... 31
 - Starting the DesignSync Client..... 32
- Installing DesignSync Data Manager for Eclipse 33
 - Setting up your DSclipse Environment..... 33
 - Installing DSclipse for Eclipse 34
 - Upgrading DesignSync Data Manager for Eclipse Integration 35
 - Uninstalling DSclipse 35
- Installing DesignSync Data Manager for Simulink 35
 - Setting up your DesignSync Data Manager for Simulink Environment on UNIX.... 36
 - Setting up your DesignSync Data Manager for Simulink Environment on Windows
..... 37
 - Customizing Your DesignSync Environment for Simulink Integration 37
- Enabling DesignSync Data Manager for Visual Studio 37
- Configuring DSDFII..... 38
 - Prerequisites 39
 - Setting Up Cadence Recognition Libraries 39
 - Additional DesignSync DFII Configuration Requirements..... 40
 - User Setup for DesignSync DFII 41
- Configuring DSCD 42

DesignSync Data Manager Administrator's Guide

Custom Designer images	42
Configuring DesignSync CD	42
Uninstalling DesignSync on Windows.....	43
Uninstalling DesignSync versions prior to V6R2013 on Windows.....	43
Uninstalling the Current Version of DesignSync on Windows	43
General Installation and Configuration Topics Registry Files	44
Client Configuration.....	44
Setting up a DesignSync Server	45
Environment Settings for Users	45
Using the SYNC_DIR Environment Variable.....	47
Using the SYNC_CUSTOM_DIR Environment Variable	47
Using the SYNC_HOSTNAME Environment Variable.....	48
Related Topics	49
Understanding the Custom Hierarchy.....	49
Using DNS Aliases with DesignSync	54
Preparing for SyncServer Configuration	55
Considerations before installing a SyncServer.....	55
Server Setup Scenarios	57
UNIX Kernel Parameters for SyncServer	60
Performance Tuning.....	61
Apache Configuration Files	61
Creating the User and Directory Structure for SyncServer Installation.....	62
Configuring the SyncServer	63

Overview of Configuring SyncServer	63
Configuring the SyncServer	64
Configuring a Server to Run on Root Ports.....	71
Scripting SyncServer Setup	72
Sample SyncServer Scripted XML Installation File	81
Post-Configuration Tasks	83
Starting the SyncServer	83
Upgrade Considerations	84
Upgrading the Email Interface.....	85
Upgrading Email Note Triggers.....	85
Configuring SUID (UNIX only)	86
Understanding the SUID Setting	86
Enabling SUID for DesignSync	87
Enabling SUID for Specific Mirror or Cache Directories	89
Disabling SUID.....	90
Troubleshooting SUID.....	91
DesignSync Administrator (SyncAdmin).....	93
ENOVIA DesignSync Administrator Tool Overview	93
Command Line Interface.....	94
Site-Wide Configurations	94
Project Configurations.....	94
User Configurations	94
Select Registry	95

DesignSync Data Manager Administrator's Guide

General.....	97
General Options	97
Command Defaults	102
Communications	106
Default Fetch State Options	107
Diff Format Options	109
Exclude Lists.....	111
Logging Options	115
Modules Options	117
Module Cache Tab.....	120
Performance Options	122
Startup Options	127
Workspaces	129
Site Options	130
Site Options	130
Client Triggers.....	134
File Content.....	157
Links Options	160
Projects	164
Revision Control (RC) Notes	169
SCC Provider	175
3DPassport	176
Third Party Integration Options	176

Vault Types 180

Development Servers..... 184

Enterprise Servers 186

GUI Options 190

 GUI Options 190

 Initial Folder Options 192

 Display Options 192

 Command Bar Options..... 194

 History Options..... 195

 Columns Options..... 196

 Customize Diff Options 198

 Vault Browser..... 200

 Module Roots..... 200

 Adding a Module Root..... 202

GUI Customization..... 204

 Main Toolbar 204

 Module Toolbar 205

 Keyboard Options 207

 Custom Tools Options..... 209

 Commands..... 210

 States Options..... 211

 Command Options 213

 Tag Options 215

DesignSync Data Manager Administrator's Guide

Custom Columns.....	217
Diagnostics	222
Diagnostics Overview.....	222
Environment Variables	222
Installation Settings	224
Java Properties	224
Using the DesignSync Web UI	227
Overview of the DesignSync Web UI.....	227
Logging into the DesignSync Web UI	228
Viewing the Privacy Policy	228
Me Menu.....	228
Add Menu	228
System Administration of DesignSync Client	231
Configuring Client Default Settings	231
Configuring the Environment	231
Overview of Configuration Approaches.....	231
Using Environment Variables.....	231
Setting Command Line Defaults.....	235
Site-Wide Configuration	235
Project-Specific Configuration.....	236
Overview of Registry Files	237
Autoloading stcl Procedures	240
Managing Revision Control Files Between Windows and UNIX.....	241

Managing SyncServer Lists	242
SyncServer List Files	242
Example Server-List Files	244
Managing Symbolic Links	245
How DesignSync Handles Symbolic Links	245
Guidelines for Creating Symbolic Links.....	246
Revision Controlling Symbolic Links to Files.....	247
Revision Controlling Symbolic Links to Folders	248
Setting up Client Triggers	249
Triggers Overview	249
Creating Triggers	252
Supported Trigger Commands.....	256
DesignSync Provided Trigger Hooks	270
Example Client Triggers.....	279
Manipulating Triggers.....	282
Events Overview	289
Creating Events.....	292
Creating Event Properties	293
Manipulating Event Properties	294
Event Properties.....	296
System Administration of SyncServers	307
System Considerations.....	307
The tmp Directory.....	307

Stopping or Restarting a SyncServer at System Shutdown or Boot Time.....	307
Setting up Licensing	308
Setting Up Licensing for DesignSync Products.....	308
Understanding License Selection and Preparing the License File	310
Configuring the License Server.....	311
Setting Up a License Server and Default Site License.....	312
Integrating a License into an Existing FlexNet License Management Configuration	314
Using a <VendorDaemonName>_LICENSE_FILE Variable	315
Setting Up Licensing for a Client-Only Client-Side Vault.....	316
The License File.....	317
Understanding DesignSync Product License Management	319
Viewing the Status of License Activity.....	320
Starting Up and Shutting Down the License Server (Imgrd only)	320
Setting Up a Server-Specific License (Imgrd only).....	323
License Management Problems and Solutions	324
SyncServer Administration	327
Overview of SyncServer Administration	327
Setting Up a SyncServer.....	328
Resetting the SyncServer	328
Using the Administer Server Panel	329
Enterprise Design Synchronization Queue	331
Setting up Secure Communications.....	332
Overview of Secure Communications	332

Setting Access Controls for Secure Communications	333
Enabling Secure Communications (UNIX only).....	334
Disabling Secure Communications	335
Using Secure Communications	336
About HTTP Proxy	337
Working with HTTP Proxy Header Information	338
Automatic Proxy Discovery Hooks	345
Setting up User Profiles	346
What Are User Profiles?.....	346
Creating User Profiles	347
Changing User Passwords.....	348
Editing and Deleting User Profiles.....	349
Cleaning Up References to Deleted Users	350
Creating User Authentication Scripts	351
Enabling LDAP.....	353
Advanced LDAP Settings.....	355
Enabling 3DPassport	363
Access Control Overview.....	365
RevisionControl Notes	366
RevisionControl Notes Overview.....	366
Revision Control (RC) Notes Options.....	367
Enabling Note Generation	372
Enabling or Overriding RevisionControl Note Generation for Specific Servers	373

Setting Up Revision Control Triggers	374
Working with RevisionControl Notes	374
Mapping RevisionControl Notes	378
How Email Notification Works	382
Configuring Email	383
Editing the Email Trigger	394
Activating the Email Trigger	395
Customizing Email Triggers	396
Troubleshooting Email Triggers	399
Setting RevisionControl Note Generation	401
Setting Up Email Notification of Module RevisionControl Notes	403
Web Interface Configuration	404
Controlling HTTP Headers	404
Accessing Apache Document Root	406
SyncServer Aliases	406
Setting up Note Object Triggers	407
Creating Note Object Triggers	407
Editing Note Object Triggers	412
Setting Note Object Trigger Execution Order	416
Managing Vault Types	417
About Vault Types	417
Setting Vault Types	419
Changing Vault Types	423

Adding or Editing a Vault Association	426
Converting Vault Data	428
Converting a Vault Repository from CVS/RCS Format to DesignSync Format....	435
Exporting Vaults	441
Importing Vaults	441
Moving Vaults.....	442
Using the Vault Utilities	444
Backup and Restore	447
Backing Up Your Server.....	447
Restoring All Server Backup Data.....	452
Restoring Your Server Vault Data	454
Procedure for Defining Automatic Backups.....	458
Moving SyncServers.....	462
Before you move a SyncServer:.....	462
Steps to Take on the Original SyncServer	463
Steps to Take on the New SyncServer	463
After You Move the SyncServer.....	465
Moving Servers with Mirrors.....	466
Reducing the Size of the Transaction Log	467
Setting Up Enterprise Design User Mappings	467
Setting up User Name Mapping	468
Enterprise Design Synchronization Queue	469
Viewing the Enterprise Design Synchronization Queue	469

DesignSync Data Manager Administrator's Guide

- Related Topics 470
- Privacy Policy and GDPR Compliance..... 471
 - Privacy Policy 471
 - Creating a Privacy Policy 471
 - Updating a Privacy Policy 471
 - Removing a Privacy Policy..... 472
 - Viewing a Privacy Policy 472
- Data Replication 473
 - Introduction to Data Replication..... 473
 - Benefits of Using Data Replication..... 475
 - Understanding Data Replication 475
 - Related topics 476
 - Mirrors Versus Caches 476
 - Data Replication System 478
 - Setting Up Data Replication 478
 - Cleaning and Removing Data Replication..... 479
 - General Replicate Settings 481
 - Add Root Settings 482
 - Add Data to Replicate 483
 - Show Roots..... 485
 - Show Replicated Data Hierarchies..... 487
 - Show Disk Usage for Root 489
 - File Caches..... 490

- How DesignSync Manages Caches 490
- Setting Up a LAN Cache 497
- Setting Permissions on the Cache 498
- Moving a Cache 500
- Cleaning a Cache..... 500
- Tips for Administering LAN Caches 504
- How DesignSync Determines the Correct Project Cache 507
- Understanding Excluding IP from the Cache 510
- Adding or Removing an Object from Caching 512
- Module Caches..... 512
 - Procedure for Setting Up a Module Cache..... 513
 - Module Caches and Legacy Modules 515
 - Updating a Module Cache..... 515
 - Cleaning a Module Cache..... 516
 - Understanding Excluding IP from the Cache 517
 - Adding or Removing an Object from Caching..... 519
 - Updating a Legacy Module Mcache 520
- The Mirror System 523
 - Mirroring Overview 524
 - Architecture of the Mirror System..... 527
 - Using Scripted Mirrors..... 531
 - Creating Scripted Mirrors 532
 - Using Mirrors with Modules 533

- Finding Mirrored Data 534
- Viewing or Editing a Mirror 535
- Displaying Mirror Status 538
- Administering Mirrors 541
- Understanding Mirror Transaction Queuing and Transaction Failures 542
- Setting Up a Mirror Server 544
- Adding a Mirror..... 548
- Setting Permissions for the Mirror 554
- Resetting The Mirror Daemons 555
- Legacy Mirrors 555
- Module Administration 565
 - Overview of Module Administration Tasks 565
 - Moving Modules..... 566
 - Module Move Process..... 566
- Module Export and Import 568
 - Understanding Module Export..... 568
 - Understanding Module Import..... 569
 - Understanding Modules Reconnect 570
 - Procedure for Exporting/Importing a Module..... 570
- Changing Hierarchical References on Older Servers 572
- Module Views 572
 - Overview of Module Views 572
 - Creating Module View Definitions 573

Displaying the Contents of a Module View.....	576
External Modules.....	577
Overview of External Modules.....	577
Defining the External Module Interface	578
Using DesignSync Commands with External Modules.....	580
Sample External Module Tcl Script.....	586
Upload Archive.....	591
Upload Archive	591
Field Descriptions.....	591
Performance Optimizations	595
Performance Optimization Overview	595
Client-Side Optimization	595
Fetching Files from the Mirror or Cache.....	595
Fetching Files Directly to Your Work Area	596
Multithreading Optimization.....	597
Turning Off Compression	598
Using Delta Transfer Hooks to Tune Client/Server Communications.....	598
Turning Off Keyword Expansion.....	600
Linking Large Files.....	601
Server-Side Optimizations	606
Changing the DesignSync Temporary Directory	606
Creating Custom Utilities.....	609
Creating Custom Utilities	609

DesignSync Data Manager Administrator's Guide

DesignSync provided Utilities.....	609
Creating and Defining a Custom Utility	609
Anatomy of a Custom Utility	609
Creating the Shell Script Wrapper.....	610
Creating the TCL Code for the Utility	611
Creating the Documentation	612
Publishing Custom Utilities	615
Publishing the Custom Utilities.....	615
Format of the Generated Index Files.....	616
Format of the Generated HTML Files.....	616
Registry Files	619
Overview of Registry Files	619
Alphabetical List of Registry Keys.....	622
Client/Server Communication	634
Audit Trail Log Maximum File Size (ACDenyLogRotateSize).....	635
Audit Trail Log (ACLogDeny)	635
Maintenance Retry Attempts (MaintenanceRetryAttempts)	637
Maintenance Retry Interval (MaintenanceRetryInterval)	638
HTTP Proxies (ProxyNamePort)	639
Data Compression (TransferCompressed)	640
Delta Transfer Compression (TransferDelta)	641
Limit Transfer Timeout (TransferTimeout).....	642
Trusted Agents and Processes (TrustedAgents)	643

Registry Settings for DesignSync Clients	645
DesignSync Client Startup Registry Settings	645
DesignSync Client Commands Registry Settings.....	652
DesignSync Client Environment Registry Settings.....	702
Modules Registry Settings	716
Vendor Objects Registry Settings	732
DesignSync GUI Registry Settings.....	777
DesignSync diff Display Registry Settings	810
DesignSync Client Optimizations Registry Settings	820
Workspace Metadata Registry Settings	831
Development Areas Registry Settings.....	835
Registry Settings for SyncServers	835
Modules Registry Settings	835
Data Storage and Maintenance Registry Settings.....	839
Enterprise Design Settings.....	868
DesignSync Web UI Registry Settings	873
Enable ACAdmin (ACAdminEnabled)	877
Registry Settings for Mirrors	878
Repository Server Registry Settings	878
Mirror Administration Server Registry Settings	884
DesignSync Client's Mirror Functionality Registry Settings.....	926
Troubleshooting the Mirror System Registry Settings	930
Scripted Mirrors Registry Settings.....	934

Diagnostics and Troubleshooting	937
List of Error Messages.....	937
DesignSync Client Troubleshooting.....	958
Diagnostics Overview.....	958
About DesignSync Client Log Files.....	959
Command Line Clients Fail to Start.....	961
Could Not Locate Module.....	962
Error Accessing or Storing to Database	963
Files Missing from the Cache.....	964
Files are Missing from My Module Workspace.....	964
Running a DesignSync Client in Debug Mode	965
Interrupt Button Does Not Seem to Work.....	967
Performance Issues with DesignSync GUI Client	967
Starting a New Browser Process Each Time Help Is Invoked.....	967
Tagging Does Not Warn of Removed Objects	968
Troubleshooting Metadata Errors.....	968
Troubleshooting nsremote Issues	970
SyncServer Troubleshooting	971
About SyncServer Log Files.....	971
Running a SyncServer in Debug Mode	973
SyncServer Error Log File Messages.....	976
Troubleshooting Email Notification of RevisionControl Notes	977
Communication Errors Troubleshooting	977

Controlling Network Communication Timeouts	977
Proxy Errors	978
Mirror System Troubleshooting.....	979
Detect "Absent Files" (FindAbsentFiles)	979
First Steps for Troubleshooting the Mirror System	980
Mirror Log Files	981
Resetting The Mirror Daemons	984
Diagnosing Performance	985
Overview of syncdiag Utility	985
Diagnosing Performance.....	987
Running a DesignSync Client in Debug Mode	992
Running a SyncServer in Debug Mode	994
External Modules.....	996
Unable to locate external module interface command Error	996
Additional Information.....	997
Add/Edit Tool Pane.....	997
Customizing the List View.....	997
General Exclusions.....	998
Tcl Script for Importing Users	998
The CPU Team Subscribes to Email on a Hierarchy	1001
Robert Deletes an Email Subscription.....	1003
Robert Subscribes to RevisionControl Notes on Module Operations.....	1003
Robert Updates Email Subscriptions.....	1004

DesignSync Data Manager Administrator's Guide

- Using the -h Option to Specify a Unique Identifier for a Remote Mirror 1006
- Getting Assistance 1009
 - Using Help 1009
 - Getting a Printable Version of Help..... 1009
 - Contacting ENOVIA 1010
- Index 1011

Overview of DesignSync Administration

DesignSync administration primarily involves setting up and managing the DesignSync servers (SyncServers) and the vault. Usually, DesignSync administration is performed by a project leader, LAN administrator, or system administrator.

Team leaders typically set up the DesignSync environment for their teams, who will use DesignSync clients to access project data.

Although you do not need special privileges such as those of a superuser (root) or system administrator to manage DesignSync, you should create and own a syncmgr account.

The syncmgr Account

The syncmgr account is a user account you create to perform SyncServer administration. This user account owns the installation hierarchy, and you can run the server from this account. Creating this separate account is important because the SyncServer, like any other HTTP server, owns the files created by the server, such as the design project repositories. By setting up a syncmgr account to own the files, you minimize the potential for accidental modification or deletion of server files; if the syncmgr account owns the files, the files are not accessible by other users.

Depending on how you set up ownership of the SyncServer and the data for your site, the syncmgr can set up, start and stop servers, and customize site-wide settings.

Note: Although the creation of this separate administrator account for managing DesignSync is not required, we strongly recommend it. Installing the tools as root bypasses any of the existing UNIX security measures already in place at the site. In addition, a separate administration account allows selective administrative access, independent of root privileges.

DesignSync and SyncAdmin

As a DesignSync administrator, you perform many tasks using DesignSync in conjunction with SyncAdmin or using a Web browser to connect to the DesignSync web server.

SyncAdmin

The Synchronicity Administrator tool (SyncAdmin) is a standalone application that is part of the DesignSync distribution. SyncAdmin provides a Graphical User Interface (GUI) for DesignSync commands and utilities. SyncAdmin lets administrators and project leaders perform various DesignSync administrative tasks from a common interface without having to use commands or edit scripts.

SyncAdmin requires no installation, and if you use a Windows client, you can access SyncAdmin from the DesignSync **Tools** menu.

Some of the tasks available through SyncAdmin include:

- Defining projects for the members of your development team
- Enabling the generation of RevisionControl notes
- Setting up client triggers
- Defining where log files are stored
- Viewing DesignSync environment variable settings
- Customizing GUI settings and options

SyncAdmin tasks are also available through the command line by using the sregistry family of commands.

DesignSync web administration

DesignSync includes a set of web pages, the DesignSync Web UI, that you and your users access through a browser. After unpacking the DesignSync distribution and configuring a SyncServer, you can launch this management tool.

To launch the DesignSync Web UI, open your browser and specify the following URL in the location area:

```
http://<host>:<port>
```

For example: `http://myhost.mycompany.com:2647`

Refer to Installation section for information about configuring a SyncServer.

For more information on Using the DesignSync Web UI, see Overview of the DesignSync Web UI.

System Administration Tasks

DesignSync administrators commonly perform tasks such as these:

- **Install or upgrade DesignSync**

For an overview of installation configuration topics, see Installing DesignSync Tools.

- **Configure the DesignSync installation**

You and your users can configure your DesignSync and installation by setting options during installation, setting options within DesignSync, and by

using the SyncAdmin tool. You can add startup scripts and stcl procedures to your installation hierarchies so they can be used by a single user, a project team, or an entire site. All customizations that you make to your installation are isolated in a custom hierarchy; for a description of this hierarchy, see *Understanding the Custom Hierarchy*.

Within the DesignSync web interface, you can customize note types and set up email notification for revision control operations. For more information, see the **Customizing ProjectSync** section of *ProjectSync User's Guide*.

- **Set up a license server for the SyncServer**

DesignSync uses FlexNet License Management software for managing product licenses. To install and use these products, you must first install the FlexNet software and the DesignSync product license.

DesignSync products and FlexNet software allow you to set up license servers and license management in several different ways. See *Setting Up Licensing* for more information on setting up a license server and installing a license.

- **Set up secure communications (SSL) for the SyncServer**

DesignSync servers (SyncServers) are capable of establishing a secure communications channel with DesignSync clients (DesSync GUI, dss, dssc, stcl, stcl, or a web browser). This secure communications channel is built on the Secure Socket Layer (SSL) protocol and the RSA Public Key encryption algorithm.

Enabling the secure communications channel requires that you obtain an X.509 Certificate from a Certificate Authority (CA) vendor and install it into the configuration directory of SyncServer machine. Then during server configuration, you specify a secure (SSL) port in addition to the standard non-secure port. See *Setting Up Secure Communications* for information on how to set up this communications channel.

- **Create a boot script to start a SyncServer when the UNIX system is restarted, for the SyncServer**

DesignSync provides a script (`S90syncinc`) that you can use to automatically start a SyncServer at system boot time. If you choose, you can use the script to start a license server as well. See *Stopping or Restarting a SyncServer at System Shutdown or Boot Time* for more information on this script.

- **Set up access controls and security for the SyncServer**

As an administrator, you may want to limit access to certain DesignSync operations. You can control access to most DesignSync actions: checking in files, checking out files, removing locks, tagging files, retiring files, deleting versions from a vault, and so on. In addition, you can specify the level of user authentication required to access DesignSync data.

For increased security, you can require username/password authentication. The first time a user attempts to access a server, DesignSync prompts for the user's username and password. This username and password must correspond to the user profile. All users accessing the server must have user profiles on that server. For information on creating user profiles, see [Creating User Profiles](#).

See the [Access Control Overview](#) and the [User Authentication Access Controls](#) for information on DesignSync access controls and user authentication methods. For more information on access controls, see the [ENOVIA Synchronicity Access Control Guide](#).

- **Set up a cache or mirror directory for DesignSync clients**

As a DesignSync administrator, you can set up a file cache so that your team can access shared copies of project vault files. By doing so, you save disk space on the LAN, as well as the time for individual fetches of updated files.

Another option for managing design data on a LAN is to set up a mirror directory. Mirrors provide an easy way for multiple users to point to a defined data set for their project. A mirror always reflects the configuration defined for a project. For example, if a team is working with the Latest files on a branch, when a user checks in a new file version on that branch, the associated mirror directory is updated with that new Latest version. See [Mirrors Versus Caches](#) for a comparison of the two methods.

- **Perform an installation of DesignSync that enforces the read-only intent of cache and mirror directories for DesignSync clients**

In situations involving mirror directories, users must have write access to the mirror because a user's process sometimes updates the mirror directory when checking in a new version. On a UNIX platform, you can install DesignSync and set the User ID and Group ID bits so that users do not need write access to mirror directories. Caches can be similarly restricted. See "[SUID Configuration](#)" in the Installation document for more information.

- **Set up RevisionControl notes for the SyncServer**

As project leader or DesignSync administrator, you can enable DesignSync to generate RevisionControl notes during revision-control operations such as check in, check out, and populate. These notes have the built-in type RevisionControl and contain information about the operation, such as the name of the user, the command used to invoke the operation, and the time of command execution. For more information on RevisionControl notes, see RevisionControl Notes Overview.

Once you enable the generation of RevisionControl notes, you can use the generation of a RevisionControl note to trigger some other action, such as running a Tcl script. For information on this use of RevisionControl notes, see Creating Note Object Triggers.

- **Set up client triggers for DesignSync clients**

A trigger is a named action, generally a script or a program, that the revision control system runs when a specified event occurs in the system (for example, a file is checked in or tagged) and that itself causes an action.

A client trigger is a trigger (usually a Tcl script) executed on the DesignSync client. An example is a trigger that runs in response to a user's checkout command and that checks to make sure that the user has specified locked files before DesignSync proceeds with the checkout. DesignSync triggers can be defined to execute a Tcl script whenever a revision-control operation occurs.

As DesignSync administrator, you can use the SyncAdmin tool to create client triggers that apply to the entire site. For more information on client triggers, see Triggers Overview .

Using ENOVIA Synchronicity DesignSync Data Manager Administrator's Guide Documentation

This guide is single-sourced in HTML and generated to multiple locations.

- Integrated help - When you press F1 within the DesignSync Administrator (SyncAdmin on UNIX) application or click on the Help button on SyncAdmin dialog boxes, the DesignSync Data Manager's Administrator's Guide help appropriate for the location or dialog opens in your default Web browser.
- DesignSync Documentation - available from the **Dassault Systems** product group in the Windows **Start** menu or on UNIX, by pointing your web browser to `$SYNC_DIR/share/content/doc/index.html`

Note: References from the *ENOVIA Synchronicity DesignSync Data Manager Administrator's Guide* to the *ENOVIA Synchronicity Command Reference* guide always

link to the ALL version of the guide, which contain information about all working methodologies for DesignSync. For more information about the available working methodologies, see ENOVIA Synchronicity Command Reference.

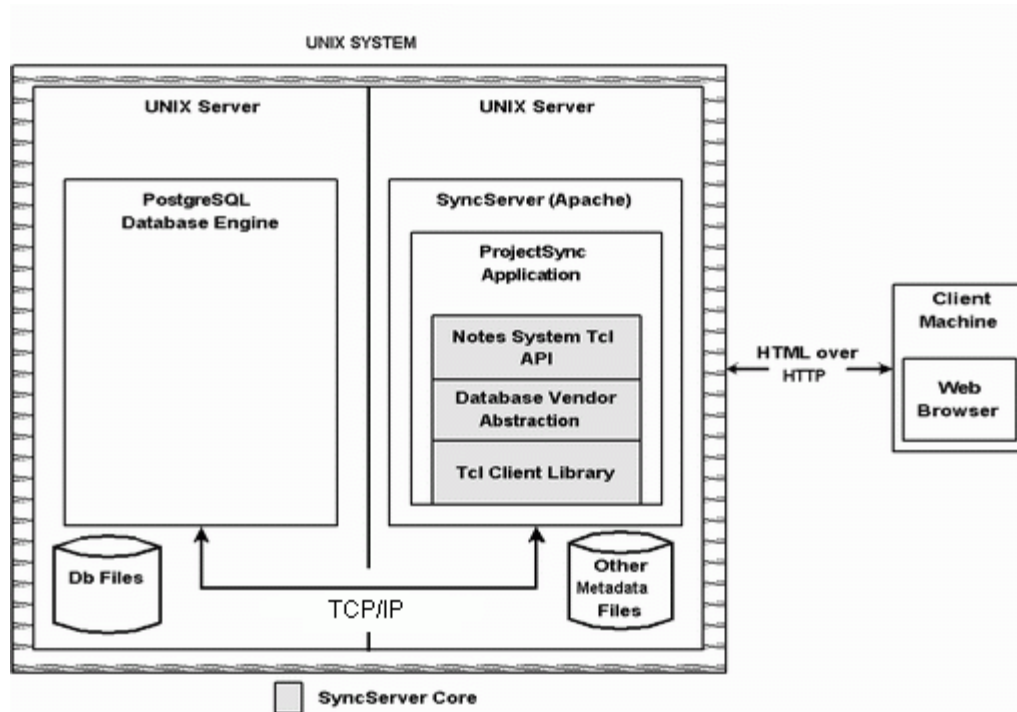
Before Reading this Guide

This guide explains the underlying concepts and tasks and procedures needed for setup and maintenance of your DesignSync server. As such, it does not have a prerequisite guide. However, for a thorough understanding of the DesignSync usage concepts and procedures for using DesignSync, you may refer to the following additional guide:

<i>ENOVIA Synchronicity DesignSync Data Manager User's Guide</i>	Describes the concepts and workflow for DesignSync in detail.
--	---

SyncServer Architecture

The SyncServer, is an HTTP server process that manages shared information and performs administrative functions such as user privilege validation. A SyncServer is an Apache web server that is extended with a DesignSync module. For DesignSync, the SyncServer controls access to design files.



The PostgreSQL database uses a client-server architecture, so the SyncServer is a client for the relational database.

The PostgreSQL database server must be on the same UNIX machine as the SyncServer and therefore can use a more optimized protocol to communicate with the client. Nonetheless, the model is still a client-server "within a box."

Apache's temporary storage area is sometimes used by the SyncServer when users perform DesignSync and ProjectSync operations. For example, copies of the files affected by a revision control operation may be stored in a temporary directory specified during the SyncServer installation, `/tmp` by default. When such operations are finished, the files in the temporary directory are deleted. When cleaning up your `/tmp` directory, take care not to remove in-progress (temporary) files needed by DesignSync or ProjectSync.

Note: Some operations do not use the temporary directory. See Changing the DesignSync Temporary Directory for more information.

Within the SyncServer module, the access controls are read by the SyncServer, not by client applications.

Overview of Enterprise Design Development

DesignSync data can be created and maintained enterprise-wide using the ENOVIA Enterprise Project Administrator functionality. They may also be created and maintained on a per server basis using the DesignSync Web interface.

Prerequisites

Any user who is creating or maintaining enterprise developments must have the access rights to use the enterprise design functionality and any specific commands and menus desired. For more information setting up access permissions for enterprise development, see Access Controls for DesignSync Projects.

Setup and Maintenance for Enterprise Developments

The administrator must define the enterprise servers are available to the users through the SyncAdmin interface as well as determine whether transactions will automatically synchronize to the enterprise server or if they will be pushed manually.

- Enterprise Servers - Define and continue the behavior of the enterprise servers.
- Enterprise Design Synchronization Queue - review and process any transactions in the synchronization queue.

See Also

Enterprise DesignSync Administration User's Guide

Getting Started

Registry Files

Setup and tool configuration information for the DesignSync tools is stored in registry files. Manually editing registry files is generally not recommended. An error while editing a registry file could cause DesignSync tools to function incorrectly. You make changes to the registry files by responding to questions during your software installation, setting options in a DesignSync client, or using the SyncAdmin GUI as a system administrator, project leader, or user. In some cases, you might need to make changes to a registry file directly.

SyncAdmin tasks are also available through the command line by using the sregistry family of commands.

Registry files exist on UNIX for DesignSync clients as well as for SyncServers. To make changes for both clients and servers, use the `SiteRegistry.reg` registry file, which is accessible to both DesignSync clients and SyncServers. The registry files are described below.

Client Registry Files (Windows and Unix)

The registry files that contain setup and tool configuration information for DesignSync clients are listed below in the order of precedence.

- `<SYNC_USER_CFGDIR>/UserRegistry.reg`

The `UserRegistry.reg` file contains a user's personal preference settings. These settings can be changed using the **Tools=>Options** menu in the DesignSync graphical user interface or SyncAdmin's "User" mode. By default, `<SYNC_USER_CFGDIR>` resolves to `<HOME>/synchronicity` on Unix and `%APPDATA%\Synchronicity` on Windows.

- `<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`

The `ProjectRegistry.reg` file lets a project leader set preferences for a project. To do so, the project leader creates a project directory and a `ProjectRegistry.reg` file within that directory. The project leader sets the `<SYNC_PROJECT_CFGDIR>` environment variable to the project directory and then customizes the `ProjectRegistry.reg` file using SyncAdmin. Project team members must set the `<SYNC_PROJECT_CFGDIR>` environment variable to the project directory so that their DesignSync clients can read the project registry.

- `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`

The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. The DesignSync installation creates a default `SiteRegistry.reg` file. This file contains the default settings that are inherited by every user. By using the SyncAdmin tool, a DesignSync tools administrator can override some of these default settings that will then be inherited by DesignSync clients and SyncServers. Users can override some of these defaults by choosing their own personal preferences in the DesignSync GUI.

Note: `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config`, which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`. `<SYNC_SITE_CNFG_DIR>` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `<SYNC_SITE_CNFG_DIR>` is not set, but `<SYNC_CUSTOM_DIR>` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`

The `SynRegistry.reg` file contains information that is required by the DesignSync tools to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Server Registry Files (Unix Only)

The registry files that contain set up and tool configuration information for SyncServers are listed below in the order of precedence.

- `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/PortRegistry.reg`

The `PortRegistry.reg` file contains preference settings that are defined for a particular host `<host>` and port `<port>` number for a specific `SyncServer`. Since multiple servers can be configured in the same `<SYNC_DIR>` installation, a separate `PortRegistry.reg` registry file is defined for each. `<SYNC_CUSTOM_DIR>` defaults to `<SYNC_DIR>/custom`.

If your server preferences are not port-specific, you can instead include your preferences in the site registry file,

`<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`, which is read by all servers. `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config`, which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`.

- `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`

The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. Use the site registry to set up default preferences for all servers, then you can use the port registry files to customize a particular `SyncServer`. The DesignSync installation creates a default `SiteRegistry.reg` file.

Note: `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config`, which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`. `<SYNC_SITE_CNFG_DIR>` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `<SYNC_SITE_CNFG_DIR>` is not set, but `<SYNC_CUSTOM_DIR>` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`

The `SyncRegistry.reg` file contains information that is required by the DesignSync tools to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the server registry files to take effect, you must restart your SyncServer. An alternative to restarting the SyncServer is to use the registry reset command in a server-side script to re-read the registry files.

Related Topics

Configuring a Multi-User Environment

DesignSync Environment Variables

Understanding the Custom Hierarchy

ENOVIA Synchronicity stcl Programmer's Guide: Working with Server stcl Scripts

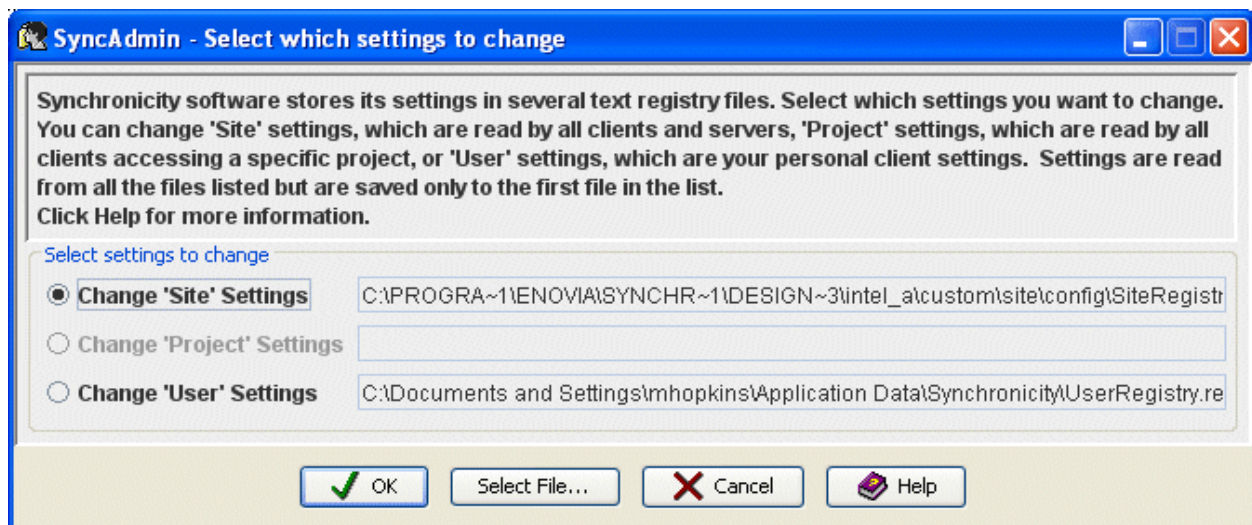
Executing SyncAdmin from UNIX

The SyncAdmin tool can be executed from either Windows or UNIX. When you make selections and then click **OK** or **Apply**, your selections are written to a registry file. If you execute the SyncAdmin tool as a LAN administrator in a UNIX environment, you can choose to write registry settings to a specific registry file. To execute SyncAdmin from UNIX, follow the step below:

Type `SyncAdmin` at the UNIX prompt and press **<ENTER>** on your keyboard.

The following SyncAdmin window displays.

Click on the image for more information about each field.



Change 'Site' settings

If you select this option, SyncAdmin stores registry settings in the site registry file. All DesignSync clients and SyncServers read the site registry file. Use the site registry file to store common site-wide settings. The site registry is stored in the `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` file where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`.

Change 'Project' settings

As a LAN administrator, you can use this option to create a new registry file or write to an existing one. As a project leader, you can use this field to specify the project registry file (`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`). If you specify a registry file with this field, DesignSync reads this registry file after reading the site registry file, but before reading the user registry file.

If the enterprise registry file (`EntRegistry.reg`) exists in the `<SYNC_ENT_CUSTOM>/config` field, DesignSync reads this registry file before reading the site registry file, the project registry file, and the user registry file.

Change 'User' settings

If you select this option, SyncAdmin reads and displays registry settings from the following registry files scanned in the order listed:

- The DesignSync registry file: `<SYNC_DIR>/share/SyncRegistry.reg`
- The enterprise registry file:
`<SYNC_ENT_CUSTOM>/config/EntRegistry.reg` only if you have set the `<SYNC_ENT_CUSTOM>` environment variable and the `EntRegistry.reg` file exists.
- The site registry file: `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`
- The project registry file:
`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg` only if you have set the `<SYNC_PROJECT_CFGDIR>` environment variable and have created a `ProjectRegistry.reg` file.
- The user registry file: `<SYNC_USER_CFGDIR>/UserRegistry.reg` where `<SYNC_USER_CFGDIR>` defaults to `<HOME>/synchronicity`

If any of the displayed values are changed, then those changes are stored back in the `UserRegistry.reg` file.

OK

Clicking on the **OK** button accepts the registry file(s) information you selected and brings up the SyncAdmin tool.

Select File

Clicking **Select File** allows you to choose a specific project registry file.

Cancel

Clicking on the **Cancel** button closes the SyncAdmin tool.

Help

Clicking on the **Help** button displays the help for the SyncAdmin tool.

Related Topics

Registry Files

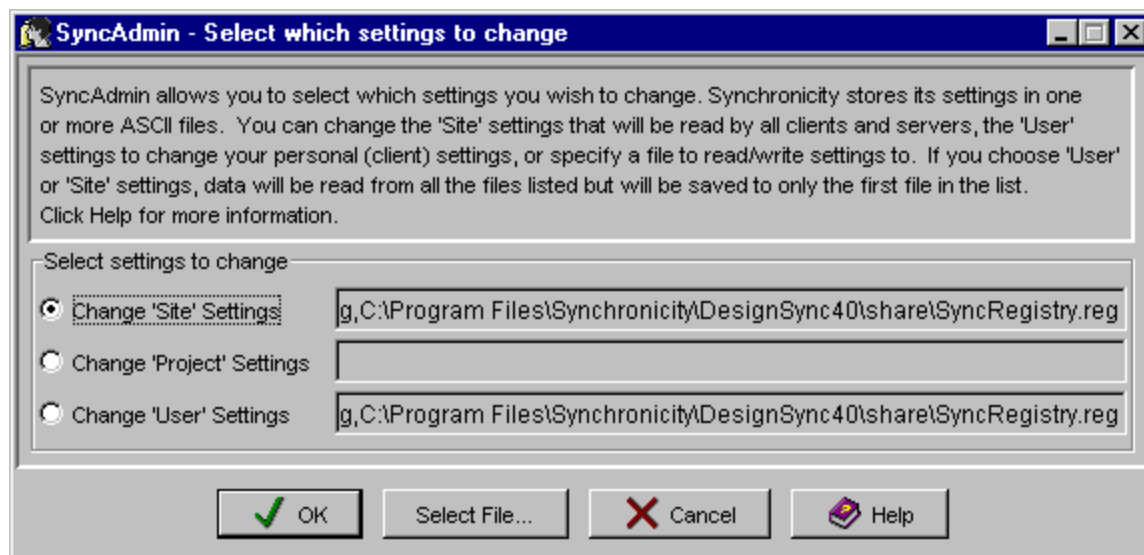
Executing SyncAdmin from Windows

The SyncAdmin tool can be executed from either Windows or UNIX. To execute SyncAdmin from Windows, follow the step below:

Select **Start =>Programs =>ENOVIA Synchronicity DesignSync Data Manager V6R2011 =>SyncAdmin**.

The following SyncAdmin window displays.

Click on the image for more information about each field.



Change 'Site' settings

If you select this option, SyncAdmin stores registry settings in the site registry file. All DesignSync clients and SyncServers read the site registry file. Use the site registry file to store common site-wide settings. The site registry is stored in the `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` file where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`.

Change 'Project' settings

As a LAN administrator, you can use this option to create a new registry file or write to an existing one. As a project leader, you can use this field to specify the project registry file (`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`). If you specify a registry file with this field, DesignSync reads this registry file after reading the site registry file, but before reading the user registry file.

If the enterprise registry file (`EntRegistry.reg`) exists in the `<SYNC_ENT_CUSTOM>/config` field, DesignSync reads this registry file before reading the site registry file, the project registry file, and the user registry file.

Change 'User' settings

If you select this option, SyncAdmin reads and displays registry settings from the following registry files scanned in the order listed:

- The DesignSync registry file: `<SYNC_DIR>/share/SyncRegistry.reg`
- The enterprise registry file:
`<SYNC_ENT_CUSTOM>/config/EntRegistry.reg` only if you have set the `<SYNC_ENT_CUSTOM>` environment variable and the `EntRegistry.reg` file exists.
- The site registry file: `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`
- The project registry file:
`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg` only if you have set the `<SYNC_PROJECT_CFGDIR>` environment variable and have created a `ProjectRegistry.reg` file.
- The user registry file: `<SYNC_USER_CFGDIR>/UserRegistry.reg` where `<SYNC_USER_CFGDIR>` defaults to `%AppData%\Synchronicity`

If any of the displayed values are changed, then those changes are stored back in the `UserRegistry.reg` file.

OK

Clicking on the **OK** button accepts the registry file(s) information you selected and brings up the SyncAdmin tool.

Select File

Clicking **Select File** allows you to choose a specific project registry file.

Cancel

Clicking on the **Cancel** button closes the SyncAdmin tool.

Help

Clicking on the **Help** button displays the help for the SyncAdmin tool

Related Topics

Registry Files

Installation

Installing DesignSync Tools

As a DesignSync administrator, it is likely that you will install the DesignSync tools for a number of users at your site or on your project team. There are many installation and configuration options for you to choose from to set up an optimal environment. Here are some installation and configuration topics to help you decide how best to customize your environment.

- Installation/Upgrade Procedure -- Both the installation and upgrade processes are documented as part of the release-specific information available in the Program Directory.

Important: You need to unpack the new distribution in a new area instead of overlaying the distribution over the existing DesignSync hierarchy.

- Server Setup Scenarios -- This topic provides a number of scenarios you can use as a model for setting up permissions for your servers.
- Using SUID to configure the directory permissions. To understand and set up SUID, see Understanding the SUID Setting.
- License Management -- To set up licensing, see Setting Up Licensing for DesignSync Products.
- User, Site, and Project Configurations -- You can customize the environment at various levels -- for a user, for a complete site, and for particular projects. See Configuring a Multi-User Environment and User Configurations for details.
- Custom Hierarchy -- All customizations that you make to your installation are isolated in a custom hierarchy. In this way, when you later upgrade your DesignSync software, or if you need to install the software for multiple platforms, your customizations are retained. For details about the custom hierarchy, see Understanding the Custom Hierarchy.
- Secure Communications -- DesignSync software supports a secure communications channel built on the Secure Socket Layer (**SSL**) protocol and the RSA Public Key encryption algorithm. See Overview of Secure Communications for more information.

ENOVIA Synchronicity DesignSync Data Manager Installation

This document explains how to install the ENOVIA Synchronicity DesignSync Data Manager products, script the installation process, and perform initial configuration.

Table of Contents

Where to Find More Information

Downloading Updated Installation Instructions and Release-Specific Information

Contacting ENOVIA

Client Installation

Preparing for Installation

Upgrade of existing DesignSync Data Manager Installations

Obtaining a License

Downloading the Distribution

Install the SyncServer into its own User Account (UNIX only)

Use an Empty Directory for the Installation

Installing DesignSync in a non-default directory

Moving the DesignSync custom area and updating the \$SYNC_CUSTOM_DIR variable

Moving and Linking to the DesignSync custom area

Updating the DesignSync Client on

Installing DSclipse for Eclipse

Upgrading DesignSync Data Manager for Eclipse Integration

Uninstalling DSclipse

Installing DesignSync Data Manager for Simulink

Setting up your DesignSync Data Manager for Simulink Environment on UNIX

Setting up your DesignSync Data Manager for Simulink Environment on Windows

Customizing Your DesignSync Environment for Simulink

Enabling DesignSync Data Manager for Visual Studio

Configuring DSDFII

Prerequisites

Setting Up Cadence Recognition Libraries

Setting the UNIX Library Path

Using sync_cds_install

Additional DesignSync DFII Configuration Requirements

Windows	User Setup for DesignSync DFII
Upgrading DesignSync Data Manager for Visual Studio	Configuring DSCD
Running the GUI Client Install Script	Custom Designer images
Running the Command Line Client Install Script	Configuring DesignSync CD
Scripting DesignSync Client Installation	Uninstalling DesignSync
<i>Creating the User Intentions File</i>	Uninstalling DesignSync versions prior to V6R2013 on Windows
<i>Running the Scripted DesignSync Client Installation</i>	Uninstalling the Current Version of DesignSync on Windows
Starting the DesignSync Client	General Installation and Configuration Topics
<i>Starting the DesignSync Clients on UNIX</i>	Registry Files
<i>Starting the DesignSync Clients on Windows</i>	Client Configuration
	Setting up a DesignSync Server
Installing DesignSync Data Manager for Eclipse	
Setting up your DSclipse Environment	
<i>Customizing Your DSclipse Environment on UNIX</i>	
<i>Customizing Your DSclipse Environment on Windows</i>	

Where to Find More Information

This document provides installation and upgrade instructions for DesignSync servers and clients on UNIX and Windows. The *ENOVIA Synchronicity DesignSync Data Manager Administrator's Guide* provides information on customizing the DesignSync server (SyncServer), licensing considerations, and other SyncServer topics.

To download the release-specific information, point your browser to:

<http://media.3ds.com/support/progdir/>

Note: You may be required to use your login in order to access information on the 3ds support site.

Release-specific information consists minimally of the following documents:

- Product Enhancement Overview - containing a list of new features and enhancements.
- General and Open Issues - containing any known release issues, platform support information, platform configuration information, and system configuration recommendations.
- Closed issues - containing a list of the issues closed in this released.

If you already have a SyncServer installed, you can access the entire online documentation collection in either the HTML or PDF format by pointing your browser to

<http://<host>:<port>/syncinc/doc/index.html>.

Downloading Updated Installation Instructions and Release-Specific Information

To download the release-specific information or the most current installation instructions:

1. Open the following URL in your Web browser:
<http://media.3ds.com/support/progdir/>
 Note: You may be required to use your login in order to access information on the 3ds support site.
2. Select the appropriate release version, for example:
 Select Product Group: 3DEXPERIENCE
 Select Level: V6R2021x
 Select Sub-Level: Golden
 Note: When you select Level, the Program Directory automatically selects the Sub-Level as the most current version for the selected release Level.
3. In the left-hand frame is a list of all available documentation, organized by type and then by Product Domain.

Contacting ENOVIA

For solutions to technical problems, please use the 3ds web-based support system:

<http://media.3ds.com/support/>

From the 3ds support website, you can access the Knowledge Base, General Issues, Closed

DesignSync Data Manager Administrator's Guide

Issues, New Product Features and Enhancements, Tech Tips, Message Board discussions, and download PDF copies of the product documentation for your product release. If you are not able to solve your problem using this information, you can submit an Incident that will be answered by an ENOVIA Synchronicity Support Engineer.

- If you are not a registered user of the 3ds support site, send email to ENOVIA Customer Support requesting an account for product support:

`enovia.matrixone.help@3ds.com`

with the subject line:

Request for access to 3DS support site

Client Installation

DesignSync consists of two separate installation packages: the client installation and the server setup. This section covers unpacking the distribution and installing the client.

Before you install the ENOVIA Synchronicity DesignSync software, make sure that you have installed all patches for the installation platform and that the platform is supported by the DesignSync software.

Preparing for Installation

Upgrade of existing DesignSync Data Manager Installations

The upgrade to the ENOVIA Synchronicity DesignSync Data Manager release is supported from version V6R2010 installations and later. To upgrade a pre V6R2010 installation, you must upgrade it to release V6R2010 before proceeding with the upgrade to the current version.

Note: ENOVIA recommends that upgrades from the previous releases be done using the latest available patch level for that release.

Obtaining a License

Before you run the install script, obtain a license for your SyncServers.

Note: The licenses are entirely controlled by the SyncServer during server access. You do not require a license for each client system.

To obtain a license:

1. Gather appropriate information. For a purchased license, you need to know:
 - o Purchase order number.
 - o Host name and hostid of the computer where you plan to install the product.
2. Contact your ENOVIA representative to obtain a valid license file.
3. Your License Administrator will process your request and send the licenses to you by email, with instructions on how to install the licenses.
4. Install the license and set up license management for your licensing configuration. For an overview of licensing configurations, see "Setting Up Licensing for DesignSync Products" in the DesignSync Administrator's Guide.

Downloading the Distribution

The DesignSync administrator should log into 3ds.support site from the OS you intend to install on and download the software. You are prompted for your customer login.

<http://www.3ds.com/support>

Tip: DesignSync recommends downloading the distribution to the platform you intend to install on. You can download to any UNIX platform for a UNIX installation or any Windows platform for a Windows installation. If you are installing on both UNIX and Windows systems, you may download once, to the machine of your choice, but you must uncompress the distribution only on the desired installation platform.

Uncompressing the software on one OS and then transferring the uncompressed files to another OS may cause installation problems.

1. Click **Download** to launch the Download page.
2. Click **Access Your Download** to open the Download My Supported Software page.
3. Select the appropriate 3DEXPERIENCE and V6 software level.
Note: DesignSync uses a single distribution containing ALL operating systems to allow you to download all your distributions in a single operation, so you should not specify an OS to order.
4. Navigate to **3DEXPERIENCE ENOVIA DesignSync** and click **Show** to display the download file.
5. Click the download file to save the file locally.

If you have any problems with your download, see the support website:

<http://www.3ds.com/Support>

Or email Support at:

Europe: Europe.SupportCenter@3ds.com

America: AG.NA-Helpdesk@3ds.com

Install DesignSync into its own user account (UNIX only)

Create a separate user account (referred to as "syncmgr" in this document) to facilitate ease of system administration and to avoid accidental modification or deletion of server files by other users. This user account owns the installation hierarchy and is also used for running the server. Do not set disk quotas on the syncmgr account.

Use an empty directory for the installation

The distribution must be installed into an empty directory. To set up this area:

1. Log in on your UNIX or Windows system as the account who will administer the DesignSync installation. On UNIX, this is usually syncmgr.
2. On UNIX, you may need to set permissions to allow world read and execute access to the installation area.

```
%umask 022
```

Note: If you are importing custom data from a previous ENOVIA Synchronicity DesignSync installation, and allow project leaders to set up and administer their own servers, set the umask to 002. This allows members of the project leaders Unix group to continue to access their servers after the import.

3. Create an installation directory, for example:
4. Create a temporary directory on UNIX and copy the distribution into it. For example:

```
%mkdir -p enovia/designsync/rel_V6R2021x  
%mkdir -p /tmp/V6R2021x  
%cd /tmp/V6R2021x  
%cp ~/<distribution_filename.tar.gz> .
```

5. Unpack the distribution:
UNIX: `% gunzip -c <distribution_filename.tar.gz> | tar -xvf -`
Windows: Unzip the distribution into a temporary installation directory by double-clicking the zip file and selecting a directory.

Installing DesignSync in a non-default directory

When you install DesignSync to a directory outside of the Program Files directory structure, the full SyncAdmin functionality is immediately available. When prompted during installation, select an alternate directory structure, for example c:\ENOVIA\Synchronicity\DesignSyncV6R2021x.

Moving the DesignSync custom area and updating the \$SYNC_CUSTOM_DIR variable

If you have a single DesignSync version installed on a machine or you are sharing the custom directory among all the installed distributions on the machine (not recommended), you may use the \$SYNC_CUSTOM_DIR variable to link to the moved custom directory. If you have more than one version of the DesignSync client on your machine, use the procedure described in Moving and Linking to the DesignSync custom area, to link to your custom directory.

Note: If you set the \$SYNC_CUSTOM_DIR variable before you install, DesignSync will

automatically create the custom directory in the location specified in the variable. If you set the \$SYNC_CUSTOM_DIR variable after installation, you must move the custom directory to the location specified in the variable.

To set the \$SYNC_CUSTOM_DIR variable:

1. Open the System Properties panel and select the Advanced tab.
2. Select the Environment Variables button.
3. In the System variables section, review the list to see if a SYNC_CUSTOM_DIR variable exists. If no variable exists, press New in the System variables section. If the variable does exist, press Edit.
4. For variable name, enter SYNC_CUSTOM_DIR. For variable value, enter the new location of the custom directory, for example:
c:\ENOVIA\Synchronicity\DesignSyncV6R2021x\custom.
5. If you have already installed DesignSync, close any open DesignSync clients, move the custom directory to the location defined in the variable and restart the clients.

Moving and Linking to the DesignSync custom area

If you have already installed DesignSync, prefer to keep DesignSync with the other programs in the Programs Files directory hierarchy, or have more than one DesignSync client version installed, you can move the DesignSync custom directory to an alternate location, if it is not already outside the Program Files directory cone, and create a link to the directory:

Note: You must already have DesignSync installed before beginning this procedure.

1. Move your \$SYNC_CUSTOM_DIR\custom directory hierarchy to a new location.
By default the custom directory is located at:

```
c:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x\win_b64\custom
```

2. Open a Command Prompt window and cd to the directory where DesignSync is installed, by default,

```
c:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x\win_b64\
```

3. Create a link from DesignSync to the new custom directory location.
mklink custom "<new_custom_dir_location>"

For example:

```
> mklink custom "c:\ENOVIA\Synchronicity\DesignSyncV6R2021x\win_b64\custom"
```

Note: If you receive a permissions error when attempting to create the link, consult your Windows system administrator for assistance. You or your system administrator may need to elevate the shell to administrator privileges.

Upgrading the DesignSync Client on Windows

If you are upgrading from a previous version of DesignSync on Windows, you may have certain settings or features that should be removed before beginning the Windows upgrade.

If you have the SYNC_DIR variable defined on Windows, you should delete the variable.

To delete the variable:

1. On your Windows system, navigate to **Start | Control Panel | System | Advanced | Environment Variables**
2. Locate the variable, which should be defined in the User Variable section, and press **Delete**.

Upgrading the DesignSync Visual Studio Integration

If you had installed DesignSync VisualStudio integration, you should remove it using the Windows uninstall.

To remove a previous DSVS version:

1. Close any open DesignSync, Visual Studio, or applications with DesignSync SCC enabled for the duration of the uninstall.
2. Choose **Start | Control Panel | Add or Remove Programs**.
3. Select the DesignSync version from the list of installed programs.
4. Select the Change option. This launches the DesignSync installer.
5. Select the Modify option and press Next>. This opens the Select Features window.
6. Unclick the checkbox for the DSVS integration. If the checkbox is already unchecked, DSVS is not installed. Press **Next>** to begin uninstalling DSVS. Note: You do not need to uninstall DesignSync to upgrade.
7. After DSVS has been removed, press Finish to close the installer.

Running the GUI Client Install Script

The client installation can be performed either using a GUI interface or a command line interface. The interfaces perform the exact same operation, so you may use whichever installation is easier for you. If you need to change any options after installation, you can use SyncAdmin to make those changes.

To run the DesignSync installation script:

1. Log in as syncmgr on UNIX or as a user with administrative privileges on Windows.
Note: When you install on Windows 7 or Vista, you must elevate to administrator privileges.

2. Change or navigate to the unpack directory, for example:

```
%cd /tmp/V6R2021x
```

3. Run the installation script:

UNIX: `StartGUI.sh`

Windows: `Setup.exe`

4. Click **Next>** when prompted.
5. Choose the destination folder for the installation. The default directory is dependent on the operating system, for example:

UNIX: `/usr/ENOVIA/Synchronicity/DesignSyncV6R2021x`

Windows: `C:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x`

To use a different directory, click the Browse button to navigate to the directory you want to install the software in, or type the full path to the installation location in the dialog box. For example:

`D:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x`

Click **Next>** to continue.

6. If this installation is an upgrade, the upgrade allows you to select whether to import customizations from a previous release. If you decide to import customizations, the next page allows you to specify the installation directory for the release to import customizations from. Enter the directory name or browse to select the previous installation directory. There is no default value.

Click **Next>** to advance to a dialog box containing the path to the three directories where customization files are kept. These fields are prefilled with the selected version's default customization directories. If you stored your custom directories in a different location use the browse button to locate and specify the customization directories or type the full path to the directories in the dialog box.

Note: On UNIX, the import also checks the three specified custom directories for hard links left over from backups and symbolic links left over from module imports. If there are any, you will see a message telling you how to fix the problem and allowing you to rerun the check and continue with the install.

Click **Next>** to continue.

7. On UNIX systems only, choose whether to enable SUID. Using SUID supports DesignSync mirror and cache functionality by restricting mirror and also cache directory access to the syncmgr account and allowing users to access the mirror and cache directories only through DesignSync. If you are using mirrors or caching on your UNIX system, it is highly recommended you enable SUID on your client executables. SUID is enabled by default.

Click **Next>** to continue.

8. On UNIX only, you can enable the integrations to the following third party tools:

- Cadence Design Systems tools and data
- Synopsys Milkyway tools and data (**Note:** The integration is still visible on the interface, however it is considered deprecated.)
- Synopsys Custom Designer tools and Data

If you have purchased any of these add-on products, select the appropriate box to install the integration. These are disabled by default. Some of the integrations provide additional message boxes to provide additional information on how to enable the integration. Click **Next>** to continue.

9. DesignSync displays a summary indicating what installation activities it will perform. If the settings are correct, click **Install>**. If you need to change any settings, click **<Back** to get to the page that controls the feature you want to change and make the change.
10. DesignSync displays a progress bar showing installation progress. After the installation has completed, press Finish to close the installer. Click the checkbox to open SyncAdmin if you want to do any additional configuration. On UNIX, you also have the option to open the INSTALL_DesignSync.txt file that provides directions on locating the SyncServer Configuration information in the *ENOVIA DesignSync Data Manager Administrator's Guide*.

Note: As part of the Windows installation, DesignSync creates the shortcuts to run the DesignSync executables from the **Start** Menu under Dassault Systemes DesignSync 3DEXPERIENCE R2021x.

If you want to install DSclipse, see section Installing DesignSync Data Manager for Eclipse. For information on starting your client, see Starting the DesignSync Client.

Running the Command Line Client Install Script

The client installation can be performed either using a GUI interface or a command line interface. The interfaces perform the exact same operation, so you may use whichever installation is easier for you. If you need to change any options after installation, you can use SyncAdmin to make those changes.

To run the DesignSync installation script:

1. Log in as syncmgr or, on Windows, as a user with administrative privileges.
2. Navigate to your unpack directory on Windows, for example:
`%cd /tmp/RV6R2021x`
3. Run the installation script:
UNIX: `startTUI.sh`
Windows: `StartTUI.exe`
4. The client reports echoes the location of the installation logs locations and introduction to client installation. Press `Enter` to begin the installation, or `q` followed by `Enter` to exit the installation without installing.
Note: You can press `q` followed by `Enter` during installation at any time to quit without installing. You can type `b` at any time during installation to go back to the previous page.

5. Enter the location to install the release. The default value is:
 UNIX: `/usr/ENOVIA/Synchronicity/DesignSyncV6R2021x`
 Windows: `C:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x`
 To use a different directory, type the full path to the installation location in the dialog box. For example:
`D:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x`
6. If this installation is an upgrade, the upgrade allows you to select whether to import customizations from a previous release. Type the number of the option or press Enter to accept the default (indicated with the *):
`1 (*) Do not Import Customizations`
`2 () Import Customizations from an earlier Installation`

If you elect to import customizations continue to Step 7. If you do not import important customizations, skip to Step 8.
7. If you elect to import customizations, at the prompt, type the installation directory of the previous release and press Enter.
 At the prompt, type the custom directory of the previous installation or press Enter to accept the default. The default value is the `custom` subdirectory of the installation directory.
 At the prompt, type the custom site directory of the previous installation or press Enter to accept the default. The default value is the subdirectory `custom\site` of the installation directory.
 At the prompt, type custom site configuration directory of the previous installation or press Enter to accept the default. The default value is the subdirectory `custom\site\config` of the installation directory.
Note: On UNIX, the import also checks the three specified custom directories for hard links left over from backups and symbolic links left over from module imports. If there are any, you will see a message telling you how to fix the problem and allowing you to rerun the check and continue with the install.
8. On UNIX systems only, choose whether to enable SUID. Using SUID supports DesignSync mirror and cache functionality by restricting mirror and also cache directory access to the syncmgr account and allowing users to access the mirror and cache directories only through DesignSync. If you are using mirrors or caching on your UNIX system, it is highly recommended you enable SUID on your client executables. SUID is enabled by default. To accept the default press `Enter`. To disable SUID, press `1` to toggle the option off, then press `Enter`.
 Click **Next>** to continue.
9. On UNIX only, you can enable the integrations to the following third party tools:
 - `1 [] Cadence Design Systems tools and data`
 - `2 [] Synopsys Milkyway tools and data`
 - `3 [] Synopsys Custom Designer tools and Data`

If you have purchased any of these add-on products, select the appropriate number for the option. After you have selected all the third-party tools you

integrate with, press `Enter`. These are disabled by default. Some of the integrations provide additional message boxes to provide additional information on how to enable the integration.

10. DesignSync displays a summary indicating what installation activities it will perform. If the settings are correct, press `Enter`. If you need to change any settings, click `b` to get to the page that controls the feature you want to change and make the change.
11. DesignSync displays a progress bar showing installation progress. After the installation has completed, press `Finish` to close the installer. Click the checkbox to open SyncAdmin if you want to do any additional configuration. On UNIX, you also have the option to open the `INSTALL_DesignSync.txt` file that provides directions on locating the SyncServer Configuration information in the *ENOVIA DesignSync Data Manager Administrator's Guide*.

Scripting DesignSync Client Installation

DesignSync features the ability to prepare a list of the client responses determined to be appropriate for the system by the DesignSync administrator and use that script to run the client install with the pre-programmed responses. The file containing the responses can be distributed site-wide to provide a uniform installation that requires less time and effort to install.

Creating the User Intentions File

When DesignSync is installed, with either the GUI or command line installer, all of the user's responses are stored in an XML file (`UserInteractions_CODE.xml`). This file is located in the DesignSync installation directory in the subdirectory `InstallData`, for example:

```
<sync_install_dir>\InstallData
```

You can modify the `UserInteractions_CODE.xml` file to be used as a response file for additional installations of the same distribution. You should select the options that match the installation flow you want to replicate when you capture this file. Edit `UserInteractions_CODE.xml` to contain the proper set of responses.

IMPORTANT: DO NOT CHANGE THE ORDER OF THE QUESTIONS! The flow is required as is in order to function properly. Changing the responses to illegal values or adjusting the order of the questions can result in unpredictable behavior and installation failure.

The most common edit you will need to make is to specify the installation directory. Find the XML similar to the following and edit the value parameter for the `SetPath` tag.

```
<SetVariable name="TARGET_PATH" setByUser="true" setType="default">  
<SetPath value="C:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x"/>  
</SetVariable>
```

IMPORTANT: The UNIX UserIntentions_CODE.xml file is different from the Windows version of the same file because there are different questions in the different install flows. When you distribute the UserIntentions script, be sure you are sending out the appropriate version for each user, or create two versions, `UserIntentions_UNIX.xml` and `UserIntentions_Win.xml` to be certain the users can identify which to use.

Running the Scripted DesignSync Client Installation

Running the Scripted DesignSync client installation requires you to copy the UserIntentions.xml to the system(s) on which you are installing the client and running the installation in scripted mode.

To run the scripted DesignSync Client Installation

1. Copy the UserIntentions file to the the temporary directory where you unpacked the distribution (in the same directory as the StartTUI.sh/StartTUI.exe file):

Important: This is not the same as the SYNC_DIR directory. The SYNC_DIR variable points to the platform subdirectory of the installation directory.

2. Run the installation:

UNIX: `$ StartTUI.sh --silent UserIntentions_UNIX.xml`

Windows: `$ StartTUI.exe --silent UserIntentions_Win.xml`

Note: The two dashes (--) preceding the silent option are required.

Tip: The installation returns all the logging and installation information to the console, as if the user were performing an interactive login. To avoid those messages, you can redirect the output to a null device. The log files remain intact if you wish to review the installation results later.

UNIX: `$ StartTUI.sh --silent UserIntentions_UNIX.xml > /dev/null`

Windows: `C:> StartTUI.exe --silent UserIntentions_Win.xml > NUL`

Starting the DesignSync Client

Starting DesignSync Clients on UNIX

Before starting Synchronicity DesignSync client applications on UNIX, each user must do the following:

1. Set the environment variable SYNC_DIR to `<install_path>/<platform>`.
The platform directories are:
 - Linux - linux_a64
 - Windows 64Bit - win_b64
2. Add `<SYNC_DIR>/bin` to your PATH variable.

To start the DesignSync Graphical Interface on UNIX, at the prompt, type:

DesignSync Data Manager Administrator's Guide

```
% DesSync &
```

To start the DesignSync shells, at the prompt, type one of the following:

```
%dss  
%dssc  
%stcl  
%stclc
```

To start SyncAdmin, at the prompt, type:

```
% SyncAdmin &
```

For more details about DesignSync shells, see "Invoking a DesignSync Shell" in the DesignSync Help topics.

Starting DesignSync Clients on Windows

You can start your DesignSync client from the **Start Menu**..

1. Choose **Start -> Programs -> Dassault Systemes DesignSync 3DEXPERIENCE R2021x** to access the DesignSync tools menu.
2. Select the program you want to start.

Installing DesignSync Data Manager for Eclipse

ENOVIA Synchronicity DesignSync Data Manager Eclipse Integration (DSclipse) is a client plug-in that integrates DesignSync with the Eclipse IDE. The DSclipse Integration provides DesignSync revision control functions from the Eclipse IDE directly, without opening the DesignSync client application.

The DSclipse plug-in is installed through Eclipse using the Eclipse Install wizard.

If you have already installed DSclipse for a previous DesignSync version, use the upgrade procedure described in Upgrading DesignSync Data Manager for Eclipse Integration.

Setting up your DSclipse Environment

The DSclipse environment may require some customization depending on your operating system platform.

Customizing Your DSclipse Environment on UNIX

The DSclipse environment may require some customization, including an update to the LD_LIBRARY_PATH and an update to the eclipse.ini.

DesignSync provides a script that, among other changes, updates the

LD_LIBRARY_PATH.

All Dsclipse users should source the following script before starting the Eclipse client:

```
$SYNC_DIR/.syncinc
```

For example: source \$SYNC_DIR/.syncinc

Tip: Add the line to source the .syncinc file to the .login script or to the configuration file for your shell (.cshrc, .profile etc.) for users who use Dsclipse.

DesignSync requires a change to the eclipse.ini file if you are running on a LINUX OS and have your system set up for Cadence collection object recognition in DesignSync. These settings increase the stack and memory size available to Eclipse.

1. Edit the eclipse.ini file to add the following lines:

```
-vmargs
-Xms40m
-Xmx256m
-Xss1m
```

2. Save the eclipse.ini file.

Customizing Your Dsclipse Environment on Windows

The Dsclipse environment for Windows requires you to add DesignSync to your Windows path in order to take advantage of the Dsclipse functionality.

To add DesignSync to your Windows path:

1. Open the **System** Properties panel and select the **Advanced** tab.
2. Select the **Environment Variables** button.
3. In the System variables section, select the Path variable and press **Edit**.
4. Add the DesignSync path to the **Variable value**.
64 Bit architecture: `c:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2021x\win_b64\bin`
5. Select **OK** to save the change.

Installing Dsclipse for Eclipse

The installation procedure is the same on both UNIX and Windows, but may differ slightly depending on your version of Eclipse.

Important: This installation requires internet access to www.eclipse.org. The Eclipse IDE may need to download required libraries from the Eclipse archives. Make sure there are no firewalls or proxy servers that will block this access.

1. Start Eclipse.
Tip: Start Eclipse with the `-clean` option.
2. From the menu, select **Help | Install New Software...** to open the Install wizard.

3. Select the **Add...** button at the end of **Work with:** field. This opens the **Add Repository** dialog.
Tip: For ease of use, you can specify "DesignSync 2021x" for the **Name** field on the **Add Repository** dialog.
4. On the Add Repository dialog, specify the **Local...** button.
5. In the Browse for Folder dialog, navigate to the `e_ide` sub-directory.
For example:
UNIX: `$SYNC_DIR/e_ide` or
Windows (64Bit): `C:\Program Files\ENOVIA\Synchronicity\DesignSync2021x\win_b64\e_ide`
and select **OK**.
This adds the DesignSync eclipse directory to the site list. When you add it, it shows up unselected.
6. Select **ENOVIA Synchronicity DesignSync Data Manager Eclipse features** and press **Next**.
7. On the **Install Details** page, select **Next**.
8. Select "**I accept the terms in the license agreement**" and select **Finish** to launch the install.
9. If you get a "Security Warning" dialog, select **OK**. (The software came with your Dassault Systemes distribution.)
10. After the install completes, you will be prompted to restart Eclipse
11. Press **Yes** to restart Eclipse, or **Apply** to begin using the DesignSync plug-in without restarting Eclipse.

Upgrading DesignSync Data Manager for Eclipse Integration

If you are upgrading an installation that used the DesignSync Data Manager for Eclipse Integration, you need to uninstall the Eclipse integration, close Eclipse and reopen with the `-clean` option, and install the new version.

Uninstalling DSEclipse

1. Start Eclipse, if it is not already running.
2. From the Help menu, select **Install New Software...** This opens the Install wizard.
3. Select the What is already installed? link. It is located right above the wizard buttons.
4. From the Installed Software tab, select "**ENOVIA Synchronicity DesignSync Data Manager**" and click **Uninstall**. This launches the Uninstall dialog.
5. Select "**ENOVIA Synchronicity DesignSync Data Manager**" and select **Finish**. You will be prompted to restart Eclipse.

Installing DesignSync Data Manager for Simulink

ENOVIA Synchronicity DesignSync Data Manager Simulink Integration is a client plug-in that integrates DesignSync with Matlab. The Simulink integration provides DesignSync revision control functions from the Matlab interface directly, without opening the DesignSync client application.

Setting up your DesignSync Data Manager for Simulink Environment on UNIX

The Designsync Data Manager for Simulink environment may require some customization depending on your operating system platform.

The Simulink environment may require some customization, including an update to the LD_LIBRARY_PATH

DesignSync provides a script that, among other changes, updates the LD_LIBRARY_PATH.

All Simulink users should source the following script before starting the Simulink client:

`$SYNC_DIR/.syncinc`

For example: `source $SYNC_DIR/.syncinc`

Customizing Your Matlab Environment

The DesignSync Data Manager for Simulink integration requires some initial customization in MATLAB.

To customize MATLAB to recognize DesignSync:

1. Start MATLAB.
2. Add DesignSync Data Manager for Simulink integration jar to the MATLAB class path as follows:


```
>> edit(fullfile(prefdir, 'javaclasspath.txt'));
<sync_install_dir>/linux_a64/classes/DSsimulink.jar
```
3. Add the DesignSync install bin directory to the MATLAB library path, librarypath, as follows:


```
edit(fullfile(prefdir, 'javalibrarypath.txt'));
<sync_install_dir>/linux_a64/bin
<sync_install_dir>/linux_a64/lib.Linux
<sync_install_dir>/linux_a64/code/bin
>> exit;
```
4. Close Matlab.

Starting Matlab

All Simulink users should source the following script before starting the Simulink client:

`$SYNC_DIR/.syncinc`

For example: `source $SYNC_DIR/.syncinc`

To start Matlab:

1. Source the .syncinc file
`source $SYNC_DIR/.syncinc`
2. Run Matlab:
`matlab &`

Setting up your DesignSync Data Manager for Simulink Environment on Windows

The DesignSync Data Manager for Simulink integration requires some initial customization in MATLAB.

To customize MATLAB to recognize DesignSync:

1. Start MATLAB.
2. From the command window, run the following commands:

```
>> edit (fullfile(prefdir, 'javaclasspath.txt'));  
Add: <sync_install_dir>\win_b64\classes\DSsimulink.jar  
>> edit (fullfile(prefdir, 'javalibrarypath.txt'));  
Add: <sync_install_dir>\win_b64\bin  
>> exit;
```
3. Add `<sync_install_dir>\win_b64\bin` to the start of your Windows PATH system environment variable.
4. Close or restart Matlab.

Customizing Your DesignSync Environment for Simulink Integration

The DesignSync Data Manager for Simulink integration requires that you operate DesignSync in a non-locking model. If this is not the typical work environment, you must update DesignSync to use the non-locking model by default on each client that will use the integration.

To enable the non-locking model:

1. Start SyncAdmin.
2. From the General tab, verify that "Check out read only when not locking" option is non checked.
3. Click Apply and close SyncAdmin.

Enabling DesignSync Data Manager for Visual Studio

DesignSync Data Manager for Visual Studio is enabled through SyncAdmin after DesignSync is installed. During the DesignSync client installation, the Visual C runtimes for DSVS and the DSVS files are installed on the system so that DSVS can be

enabled at any time and does not require the installation media or rerunning the installation media.

To Install DesignSync Data Manager for Visual Studio:

1. Close any SCC integrated applications.
2. Start SyncAdmin.
Start -> Programs -> Dassault Systemes V6R2020x -> SyncAdmin
3. Select **Change 'Site' Settings**. Click **OK**.
4. Select **Site Options -> SCC Provider**
5. Select the appropriate SCC integration:
 - SCC Integration (multiple providers)** - DesignSync Source Code Control integration for applications allowing multiple SCC providers. For example, Visual Studio 2008 and 2010.
 - SCC Integration (single provider)** - DesignSync Source Code Control integration for applications allowing only a single SCC provider. For example, Access 2007. If you aren't sure whether your application supports a single SCC provider or multiple SCC providers, consult the documentation provided with your SCC compliant application.
6. Click **OK** to save the option and close SyncAdmin or **Apply**, to save the option and continue to make changes in SyncAdmin.
7. Open your SCC compliant application to use the integration.
Note: When you open the SCC compliant application, you may need to turn on source control within the application or change the source control provider to the DesignSyncSCC

The DSVS environment for Windows requires you to add DesignSync to your Windows path in order to take advantage of the functionality.

Note: If you are using both Eclipse and DSVS on your Windows system, you only need to perform this operation once.

To add DesignSync to your Windows path:

1. Open the **System** Properties panel and select the **Advanced** tab.
2. Select the **Environment Variables** button.
3. In the System variables section, select the Path variable and press **Edit**.
4. Add the DesignSync path to the **Variable value**.
`c:\Program Files\ENOVIA\Synchronicity\DesignSyncV6R2020x\win_b64\bin`
5. Select **OK** to save the change.

Configuring DSDFII

All the necessary components of the DesignSync DFII product are installed as part of the standard UNIX installation of ENOVIA Synchronicity DesignSync Data Manager. You only need to configure DesignSync DFII if you use Cadence tools.

Prerequisites

The Cadence installation must be on the user's path for correct object recognition to take place.

During DesignSync client installation, you should have specified using DSDFII during the third-party integrations option by enabling, "[Cadence Design Systems tools and data](#)." If you did not select the option during installation, you can enable DSDFII collections using SyncAdmin.

To enable DSDFII in SyncAdmin:

1. Start SyncAdmin, for example:
`% SyncAdmin &`
2. Select **Change Site Settings** and click **OK** to start SyncAdmin.
3. Select **Site Settings | Third Party Integration**.
4. On the Third Party Integration panel, click **Enable Cadence Design Systems' Collections** and select **OK**.

DesignSync must be configured so that fetches are read-only. If DesignSync is not configured so that fetches are read-only, the "Read Only Local (get/keep)" settings for checkin and checkout from DesignSync DFII are ignored and the local files will be editable. You set this preference after installation using the SyncAdmin tool. All users inherit the site setting and should not override this setting (as is possible from SyncAdmin's General dialog box).

To set fetches as read-only in SyncAdmin:

1. Start SyncAdmin, for example:
`% SyncAdmin &`
2. Select **Change Site Settings** and click **OK** to start SyncAdmin.
3. Select **General**.
4. Select **Check out read only when not locking** and select **OK**.

Setting Up Cadence Recognition Libraries

The Cadence IC system requires that any DM system working with GDM provide a dynamically linked library. This library must either be installed in the Cadence installation using the `sync_cds_install` program or it must be made available through the library path.

Setting the UNIX Library Path

You can make the necessary libraries available to the Cadence system by setting the library path. To use this method, each user must ensure that the `GDM_USE_SHLIB_ENVVAR` environment variable is set:

```
setenv GDM_USE_SHLIB_ENVVAR 1
```

Also, users must ensure that their library paths include the DesignSync installation. Users set the library path using the `LD_LIBRARY_PATH` environment variable.

Note: The following library path includes both 32bit and 64bit libraries. Cadence recommends including both sets in the library path.

Set the LD_LIBRARY_PATH (or SHLIB_PATH) path to include the following directories:

```
<cadence_install_dir>/tools/lib
$SYNC_DIR/cds/lib/Linux
$SYNC_DIR/cds/lib/Linux/64bit
```

Using sync_cds_install

The sync_cds_install utility is used to install the library into your Cadence distribution. Since sync_cds_install must place a file into the Cadence installation, the program must be run by a user with write access to the Cadence installation area, for example, the cdsmgr user.

To run sync_cds_install, execute the program:

```
$SYNC_DIR/bin/sync_cds_install
```

and follow the prompts to specify the Cadence installation area(s).

The sync_cds_install utility installs the necessary libraries. If you have Cadence and DesignSync installed for multiple architectures, you must execute sync_cds_install on each architecture or installation.

After you run sync_cds_install, a message displays indicating that DesignSync DFII has been configured.

Additional DesignSync DFII Configuration Requirements

When the install script finishes, you must complete the following steps to make DesignSync DFII fully functional.

1. Verify that DesignSync works. In a shell window, type `%dssc`
You should see the following prompt: `dss>`
This prompt indicates that the software will work correctly.
Type `"exit"` to exit the dssc shell.
NOTE: If you do not see the `dss>` prompt, check that your SYNC_DIR environment variable has been set correctly. For information on setting the SYNC_DIR variable, see the *ENOVIA Synchronicity DesignSync Administrator's Guide*.
2. Comment out any DMTYPE lines in the installation cdsinfo.tag file:
`<cds_root>/share/cdssetup/cdsinfo.tag`
This action keeps processes for other data management tools from starting unnecessarily. Library-specific cdsinfo.tag files must have DMTYPE set to 'sync' for DesignSync DFII to work correctly. Setting DMTYPE to 'sync' in the installation cdsinfo.tag is not necessary (although not harmful).

3. Configure the online documentation by creating a link from the Cadence documentation hierarchy to the DesignSync documentation hierarchy:

```
% cd <cds_directory>/local/doc  
% ln -s $SYNC_DIR/share/content/doc syncinteg
```

User Setup for DesignSync DFII

All users must have their environments properly set up to access the DesignSync software from DFII.

1. DesignSync DFII relies on Cadence software to properly recognize and handle Cadence data. Users must have the Cadence executables directory in their PATH environment variable (so that DesignSync DFII can locate the cds_root executable). For example, they might have the following lines in their .cshrc file:

```
# Cadence installation directory is /usr/cds  
setenv PATH /usr/cds/tools/bin:/usr/cds/tools/dfII/bin:${PATH}
```

2. Users must define a SYNC_DIR environment variable that points to their DesignSync installation directory. Users must also have the DesignSync executables directory (<SYNC_DIR>/bin) in their PATH environment variable. For example, users might have the following lines in their .cshrc file:

```
setenv SYNC_DIR /usr/syncmgr/syncinc/linux_a64  
setenv PATH ${SYNC_DIR}/bin:${PATH}
```

Note: The SYNC_DIR variable changes based on the platform on which DesignSync is installed.

3. Cadence versions 6.1.2.500.17 and higher include an option to reduce the informational data requested from the server when DesignSync operations are performed. Using this option can greatly improve performance. To enable this option, set DD_GDM_OPTIMIZE to true. Users can include this setting in their shell initialization; for example, users can include this line in their .cshrc file:

```
setenv DD_GDM_OPTIMIZE true
```

4. If you do not want file-based operations to appear in the menus when operating on modules, you can enable the syncOnlyModulesData SKILL variable. You cannot set a SKILL variable from the UNIX shell so you need to put the following line in your .cdsinit file BEFORE you load the dssInit.il file:

```
syncOnlyModulesData=t
```

5. Verify that DSDFII initialization code loads after any other SKILL(TM) functions that create user triggers. The DSDFII code creates a dynamic trigger function that includes the user's predefined triggers and failure to load it last can result in the user's trigger functions not being set.
6. Users must have the following line in their .cdsinit file to load the DSDFII integration automatically at startup:

```
load(strcat(getShellEnvVar("SYNC_DIR") "/cds/skill/dssInit.il"))
```
7. Each user must have access to at least one SyncServer. The user does not need to be the owner of the server. The project leader or individual users

can optionally specify the SyncServer(s) that users will access. Doing so facilitates the selection of SyncServers from the Synchronicity dialogs in DFII. See the DesignSync DFII documentation for information on creating `sync_servers.txt` files.

Each user can now invoke DFII. The Synchronicity menu is now available in the CIW and design editor windows.

By default, Library Manager has its own forms and functions that then communicate with DesignSync DFII through the GDM layer. You can optionally configure Library Manager to call Synchronicity forms and functions directly. See the "Configuring Library Manager To Call Synchronicity Forms" topic within the *ENOVIA Synchronicity DesignSync Data Manager DSDFII User's Guide* for details.

Configuring DSCD

All the necessary components of the DesignSync CD product are installed as part of the standard ENOVIA Synchronicity DesignSync Data Manager installation on UNIX. You only need to configure DesignSync CD if you use Synopsys tools that store data in the Synopsys Custom Designer database format.

Custom Designer images

All the *.xmp files located in the distribution at `$$SYNC_DIR/dscd/images` should be copied or linked to an images subdirectory within one of the Synopsys Custom directories, for example `$$SYNOPTSYS_CUSTOM_SITE/images`. For more information on where the Synopsys custom files are placed, consult the Synopsys documentation.

Configuring DesignSync CD

During DesignSync client installation, you should have specified using DesignSync CD during the third-party integrations option by enabling, "`Synopsys Custom Designer tools and Data.`" If you did not select the option during installation, you can enable DSCD using SyncAdmin.

To enable DSCD in SyncAdmin:

1. Start SyncAdmin, for example:
`% SyncAdmin &`
2. Select **Change Site Settings** and click **OK** to start SyncAdmin.
3. Select **Site Settings | Third Party Integration**.
4. On the Third Party Integration panel, click **Enable Custom Designer Open Access Collections and Synopsys' Custom Designer interface** and select **OK**.

To run the DSCD integration automatically at startup, all users must have the following line in their .cdesigner.tcl file:

```
source [file join $::env(SYNC_DIR) "dscd/tcl/dscdInit.tcl"]
```

Uninstalling DesignSync on Windows

The procedure for uninstalling DesignSync releases has changed on Windows. This section documents both the former procedure, for removing any old-style clients, and the new procedure for removing the current client installation.

Uninstalling DesignSync versions prior to V6R2013 on Windows

This procedure uninstalls the DesignSync clients and DSVS. The DSclipse integration can be uninstalled from within DSclipse as documented in Uninstalling DSclipse.

1. Close any open DesignSync, clients, and, DSVS was installed, any Visual Studio, or applications with DesignSync SCC enabled for the duration of the uninstall.
2. Choose **Start -> Control Panel -> Add or Remove Programs**.
3. Select the DesignSync version from the list of installed programs.
4. Select the **Remove** option. This launches the DesignSync uninstaller.
5. Confirm the Remove action by pressing **Yes**, when prompted.
6. After DesignSync has been removed, press Finish to close the installer.

Uninstalling the Current Version of DesignSync on Windows

This procedure uninstalls the DesignSync clients and DSVS. The DSclipse integration can be uninstalled from within DSclipse as documented in Uninstalling DSclipse.

1. Close any open DesignSync, clients or clients using the DesignSync integrations, such as DSclipse or DSVS.
2. If you are using DSVS, you must disable it using **SyncAdmin | Site Options | SCC Provider**, then close SyncAdmin.
3. Open the command window, for example:
Start | All Programs | Accessories | Command Prompt
4. Change to the installation directory, stop the SyncDaemon and unregister it:

```
c:> cd <installation_directory>\<platform>\bin  
C:> syncdadmin stop -force  
C:> syncd /unregister (optional if you are uninstalling for the purposes of upgrading your software)
```

Note: Running syncd /unregister is not a silent operation. If you are running a scripted uninstall or operating from a remote console, this may not be a desirable operation, and is not necessarily if you are uninstalling with the intent to upgrade the DesignSync client.

5. Remove the shortcuts to DesignSync from the **Start** Menu.
6. Delete the installation directory.

General Installation and Configuration Topics Registry Files

Registry files exist for DesignSync clients as well as for SyncServers. To make changes for both clients and servers, use the SiteRegistry.reg registry file, which is accessible to both DesignSync clients and SyncServers. See the "Overview of Registry Files" topic in the DesignSync Administrator's Guide for more details.

Client Configuration

The client settings are typically configured by an administrator who sets up the DesignSync clients for many users. The information provided by the administrator is stored in a site-wide registry file, which is inherited by every user running DesignSync applications. The administrator can use SyncAdmin to change the client configuration from its default settings. Once the clients are configured, users on UNIX need only to add <SYNC_DIR>/bin to their PATH variable.

One of the client settings is the location of the cache directory for shared caching of files from the server. This mechanism minimizes the disk space used when more than one person is referencing the same version of a file. Rather than all users having their own file copies, a cache can be established that is shared among a number of users having access to a common file area. Using the cache, if two users execute a shared checkout of the same version of a file, the first user causes the file version to get cached (copied into the cache) and a link is created in this user's working area. The second user's checkout causes a link to be created in this user's working area to the same file in the cache. Because the client must be accessible from a common disk area, the default location for the cache is a directory parallel to the directory in which DesignSync is installed named sync_cache; for example your directory structure would look like this:

```
/home/DesignSyncV6R2021x/syncinc/linux_a64
```

/home/DesignSyncV6R2021x/syncinc/sync_cache

Note: When the cache directory is created, its permissions are set to 2777 (wide-open permissions and the set-group-ID bit set). See the *ENOVIA Synchronicity DesignSync Data Manager Administrator's Guide* for information about changing cache permissions.

Another client setting found in SyncAdmin determines whether a file checked out without a lock will be read-only or read/write. The default file permission is read-only. All users inherit the permission setting, but individual users can change their personal setting from the SyncAdmin's General dialog box.

Setting up a DesignSync Server

The DesignSync Server setup and configuration is documented in the *ENOVIA Synchronicity DesignSync Data Manager Administration Guide* built into the DesignSync documentation. This documentation includes:

- Prerequisites and OS-specific tuning parameters
- SUID configuration
- License server setup information
- Running the sync_setup to configure the server

To open the *ENOVIA Synchronicity DesignSync Data Manager Administration Guide*:

Windows: choose **ENOVIA Synchronicity Documentation** from the Dassault Systems product group in the **Start** menu.

UNIX: Point your Web browser to: file:///SYNC_DIR/share/content/doc/SyncAdmin/syncadmin.htm

Copyright (c) 1997-2020 Dassault Systemes. All Rights Reserved.

Environment Settings for Users

Once you install the DesignSync software, you must ensure that each user's environment is configured properly. You can configure the environment of all users at your site. See the Overview of Configuration Approaches. At a minimum, you must ensure that the following environment settings are in place for each user.

- If users are running on UNIX, they must have these environment variables set:

Variable	Description
SYNC_DIR	Path to the directory of the DesignSync

	installation. For information on setting and using the SYNC_DIR variable, see Using the SYNC_DIR Environment Variable.
HOME	Path to user's login account's top-level directory. This variable is typically already defined for users when they log in.
SYNC_CUSTOM_DIR	Path to the server customization area. This area is where all the customizations for DesignSync components are stored. For more information on setting and using SYNC_CUSTOM_DIR, see Using the SYNC_CUSTOM_DIR Environment Variable
SYNC_SITE_CUSTOM	Path to the site custom area. The default value is <code><SYNC_CUSTOM_DIR>/site</code> .
SYNC_SITE_CNFG_DIR	Path to the site custom configuration area. This is used, among other things, to store licensing information. The default value is <code><SYNC_SITE_CUSTOM>/config</code> .
SYNC_DOMAINNAME	An alias by which the host machine is known. This variable is typically set by the .syncinc script located in \$SYNC_DIR and used to determine how the client communicates. If the SYNC_DOMAINNAME cannot be set automatically, some of the command line client startup scripts may fail. For more information on setting the \$SYNC_DOMAINNAME, see Command Line Clients Fail to Start.
SYNC_JRE_DIR	Path to the Java run-time environment used to support the DesSync graphical client. DesignSync provides a default verified JRE. You may use your own JRE as long it is a compatible version that contains JavaFX.

- The executables directory (`<SYNC_DIR>/bin`) must be in users' PATH environment variables. For example, users might have the following line in their `.cshrc` files.

```
setenv PATH ${SYNC_DIR}/bin:${PATH}
```

- To set these or other standard environment variables for all users of your UNIX installation, you can add them to `<SYNC_DIR>/syncinc.custom`. See Using Environment Variables for details.

Note: For multi-platform installations, a .syncinc.custom file is needed for each SYNC_DIR.

Using the SYNC_DIR Environment Variable

On UNIX the SYNC_DIR Environment variable is used to identify the DesignSync installation directory. When you install the DesignSync distribution, as described in the *ENOVIA Synchronicity DesignSync Data Manager Installation*, you install it into an empty directory. This is the install path. The SYNC_DIR variable must be set to the install path/platform:

```
<install path>/linux_a64
```

For example:

```
% setenv SYNC_DIR /tools/DesignSyncV6R2013/linux_a64
```

Using the SYNC_CUSTOM_DIR Environment Variable

All of the customizations for the DesignSync components are stored in one directory hierarchy. By default, the install script creates a directory for this purpose:

- Windows: c:\Program Files\ENOVIA\Synchronicity\<DesignSyncVersion>\custom
- UNIX: \$SYNC_DIR/custom

You can instead use the SYNC_CUSTOM_DIR environment variable to use a different directory for the custom area. Use the SYNC_CUSTOM_DIR environment variable under these circumstances:

- If you have more than one version of ENOVIA Synchronicity DesignSync installed at one time, or you want to install DesignSync for multiple platforms, you can use the SYNC_CUSTOM_DIR to point to a different custom directory for each. The directory hierarchy and file names in the custom directory may change from release to release.
- If you plan to install the ENOVIA Synchronicity DesignSync software on a read-only disk after you untar the installation, use the SYNC_CUSTOM_DIR variable to locate the customizations on a read/write disk. With this scenario, use a different path for SYNC_CUSTOM_DIR for each installation and each time you upgrade each installation. See *Projects with a CAD Manager Managing the Location*.
- When you upgrade, the sync_setup script gives you the option of migrating your existing custom hierarchy to the new SYNC_CUSTOM_DIR location.
- If you intend for different project leaders or users to own parts of the custom hierarchy, use SYNC_CUSTOM_DIR to locate the custom area outside of the installation directory. For example, you might have project leaders own and administer individual server setups. With this scenario, use the same path

for the `SYNC_CUSTOM_DIR` environment variable for each installation and for each upgraded installation. You must retain the same `SYNC_CUSTOM_DIR` in this case because if the administrator (`syncmgr`) migrates the `SYNC_CUSTOM_DIR` area to the new installation, that administrator might inherit ownership of the server setups. (Most systems require root access to preserve file ownership.) With this scenario, you must upgrade all servers at the same time.

To use `SYNC_CUSTOM_DIR` to specify a different directory to store customizations, you must specify the full path of the directory. The `syncmgr` must have write permissions to this directory. If `SYNC_CUSTOM_DIR` is going to be used, you must set it before running `sync_setup`.

Notes:

- The new installation modifies settings in the custom area; if you set `SYNC_CUSTOM_DIR` to the same location as a previous installation, the custom area will no longer be compatible with previous software versions.
- If you use `SYNC_CUSTOM_DIR`, this environment variable will need to be set by `syncmgr` and ALL users before they run DesignSync clients or SyncServers. The easiest way to accomplish this is to set `SYNC_CUSTOM_DIR` in the shell script file:

```
$SYNC_DIR/.syncinc.custom
```

You may need to create this file if it does not already exist. A sample `.syncinc.custom` file might look as follows:

```
#!/bin/csh -f
setenv SYNC_CUSTOM_DIR
/home/tools/enovia/designsync/rel_V6R2020x/custom
```

If your installation is stored on a read-only disk, you will not be able to create a `<SYNC_DIR>/ .syncinc.custom` script. Instead, use the `SYNC_CLIENT_SCRIPT` environment variable to point to your custom shell script.

Using the `SYNC_HOSTNAME` Environment Variable

The `SYNC_HOSTNAME` environment variable allows you to use an alias as the `SYNC_HOSTNAME`, instead of the default installation machine name, to provide automatic fail-over, easier manual fail-over, and script portability. The hostname can be set to a host alias, for example:

Related Topics

[Overview of Configuration Approaches](#)

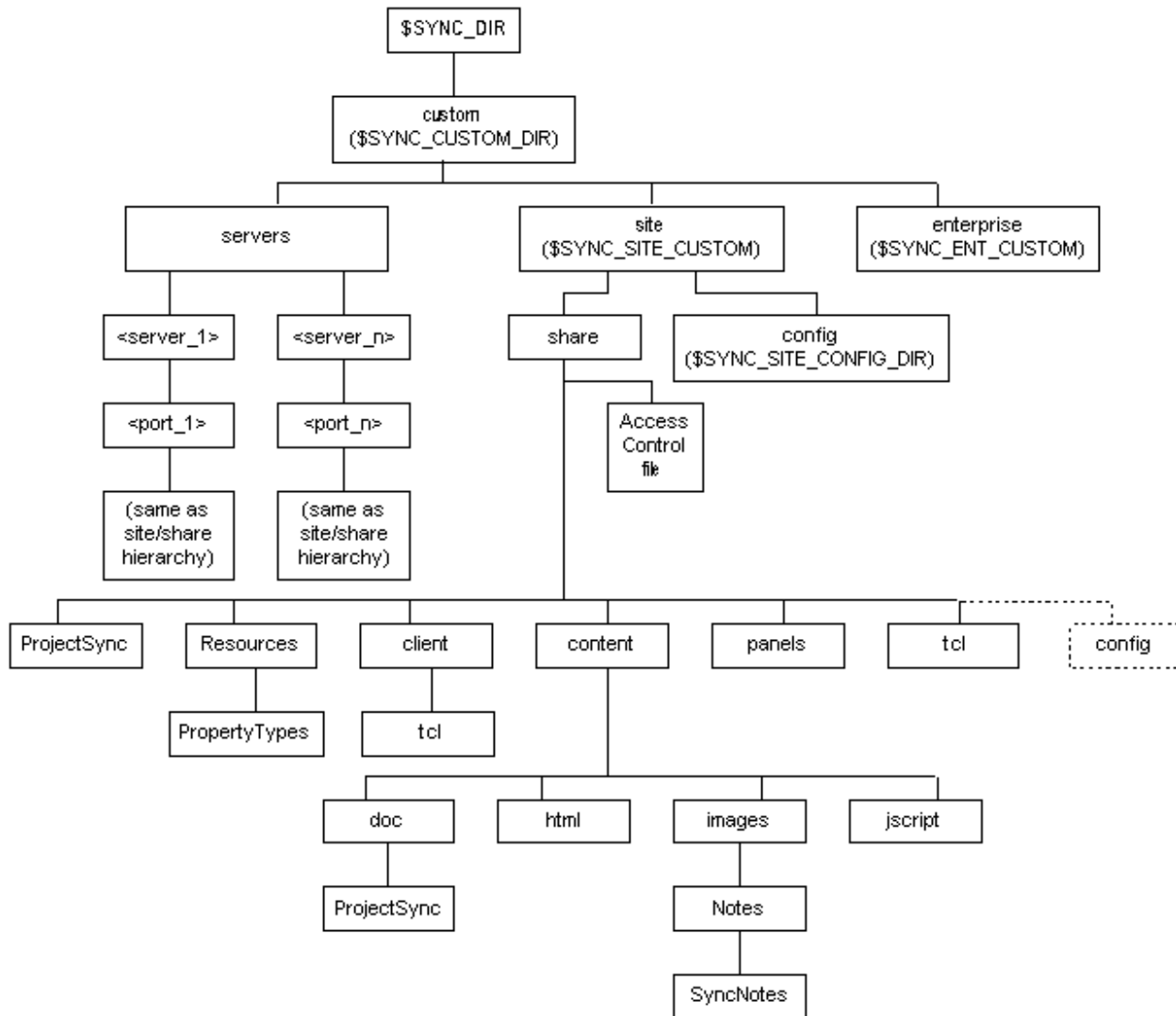
[ENOVIA Synchronicity Administrator Tool Overview](#)

Understanding the Custom Hierarchy

DesignSync administrators and users can customize their product installations by setting options during installation, setting options within DesignSync and ProjectSync, as well as by running SyncAdmin. They can also add startup scripts and stcl procedures to their installation hierarchies to be used by an entire site, a project team, or a single user, customize note types and set up email notification for revision control operations.

The custom hierarchy is created automatically on UNIX. The installation procedure on Windows creates only those directories required during installation. Windows DesignSync administrators and users must create custom directories manually as they add customizations to their installations. For example, a user must create a <SYNC_CUSTOM_DIR>/site/share/client/tcl directory manually to store Tcl procedures for Tcl autoloading. See Autoloading Tcl Procedures for more information. A fully customized Windows installation has the same directory structure as that shown because the administrator or user has manually created these custom directories.

The figure below illustrates the custom hierarchy. Click on a directory or file to see its description.



\$SYNC_DIR Directory

`$SYNC_DIR` represents the installation directory. You must set the `SYNC_DIR` environment variable before running DesignSync on UNIX. Your `PATH` environment variable must include `<SYNC_DIR>/bin`. See *Installing DesignSync* for more information.

custom Directory

The `custom` directory stores the customizations for the DesignSync installation. All of the server and site-wide settings reside in this directory.

servers Directory

The `servers` directory is used to configure web servers for the installation. When a server is configured with the `sync_setup` script, the script puts the server's configuration settings in this directory.

For each machine configured to run a server, the script creates a subdirectory at this level in the hierarchy. Under each machine's subdirectory is a directory for each port configured. These port directories contain the actual configuration files used for specific servers.

site Directory

The `site` directory stores customizations for all clients and servers using the installation. If you use a site-wide startup file (for example, `dsinit.dss`), put it in this directory. Each user who wants to use the startup file must specify it in DesignSync.

enterprise Directory

The `enterprise` directory is intended for enterprise-wide customizations. The custom `enterprise` area is a placeholder for future development.

share Directory

The `share` directory contains platform-independent content. Files in this directory are shared among all servers that use the installation directory.

config Directory

The `config` directory stores configuration information for the installation.

An optional file, `sync_servers.txt`, can be placed in this directory. If you want servers to appear under DesSync's Site Servers folder, you must specify them in `sync_servers.txt`. For more information about this file, see SyncServer List Files.

On UNIX platforms, these files are stored in the `config` directory:

- The site-wide registry file, `SiteRegistry.reg`. This file holds settings for all clients and servers using this installation.
- The site-wide license file, `syncinc.lic`. If you use a site wide license file, the `start_license_server` script and the "Configure a server" selection in the `sync_setup` installation script both look in this directory for the license file. For more information about a site-wide license, see Setting Up a License Server and Default Site License.

AccessControl File

The `AccessControl` file contains definitions of site-wide access control privileges. For more information on this file, see the [Access Control Overview](#).

ProjectSync Directory

This directory contains customizations specific to ProjectSync. In this directory, you can find any or all of the following files:

- `Configuration` - Used to configure background colors and menu appearance.
- `customize.psm` - Used to add customized menu entries to the default selections.
- `QueryFormat` - Used to define the appearance of query result tables.

For more details these files, see the [ProjectSync User's Guide](#).

Resources directory

The `Resources` directory contains property type definition files for the site. These property types are typically used in note type definitions. See the topic *ProjectSync User's Guide: What Are Property Types* for more information.

client Directory

The `client` directory contains settings that are specific to DesignSync client processes (`DesSync`, `dssc`, `dss`, `stcl`, `stcl`).

content Directory

The `content` directory contains information directly accessible via a DesignSync server. For security reasons, only files under this subdirectory of the site-wide custom directory are visible to the server. The server defines an alias, `/syncsite`, which points to this directory.

panels Directory

The `panels` directory stores DesignSync and ProjectSync Web UI panel definitions.

See the [ProjectSync Advanced Customization Guide](#) for information on customizing the layout and behavior of the DesignSync Web UI panels.

tcl Directory (in the share Directory)

The `tcl` directory contains any Tcl scripts that you want to execute from a DesignSync server. For security reasons, only files that reside in `tcl` directories within the installation search path are visible to these servers.

PropertyTypes Directory

The `PropertyTypes` directory contains the property type definitions.

tcl Directory (in the client Directory)

The `tcl` directory contains any Tcl scripts that you would like automatically loaded when a client process (for example, DesSync or dssc) starts up. Only Tcl procedures should be placed in these Tcl scripts.

In addition, you can put the `autoload.tcl` file in this directory. The `autoload.tcl` file is sourced when a client process starts. In this file you should put any needed alias commands.

For more information on the `autoload.tcl` file, see [Autoloading Tcl Procedures](#).

doc Directory (in the content Directory)

The `doc` directory stores documentation visible to a DesignSync server.

html Directory (in the content Directory)

The `html` directory is where you put custom pages destined for display in the DesignSync Web UI. Since these pages are directly accessible from the SyncServer, their content should be static and should not depend on any processing from other sources (such as Tcl scripts).

images Directory (in the content Directory)

The `images` directory contains image files for display in the DesignSync Web UI. Typically, this area holds image files for custom note types. DesignSync menus use these image files for each note type defined.

See the [ProjectSync User's Guide](#) for information on creating custom note types.

jscrip Directory

The `jscrip` directory stores JavaScript files used on the DesignSync Web UI forms. JavaScript is a scripting language that can be used to extend and enhance the functionality of pages displayed in a web browser.

ProjectSync Directory (in the doc Directory)

The `ProjectSync` directory stores online documentation visible to a SyncServer server.

Notes Directory (in the images Directory)

The `Notes` directory stores images specific to note types. When displaying an image to represent a note type, DesignSync searches for a gif file, `SyncNotes/<Note Type name>/Add.gif`, relative to the `images/Notes` directory.

See the ProjectSync User's Guide for information on creating custom note types.

SyncNotes Directory

The `SyncNotes` directory stores images specific to note types. When displaying an image to represent a note type, DesignSync searches for a gif file, `<Note Type name>/Add.gif`, relative to the `images/Notes/SyncNotes` directory.

See the ProjectSync User's Guide for information on creating custom note types.

Using DNS Aliases with DesignSync

A best practice recommendation is to use DNS alias names for server host names. When you use a DNS host name for a server, you can move the server without disrupting its DesignSync clients if the clients have set their vault to the alias and not to a SyncServer host and port. (See Moving SyncServers for information.)

If you are changing the alias for a server with mirrors, you must disable all mirrors before changing the alias.

You can specify a DNS alias during the initial server configuration of the product or by re-running the server configuration script. (For detailed instructions, see the ENOVIA Synchronicity DesignSync Data Manager Installation.)

1. To run the server configuration script, log in as `syncmgr`, change to your DesignSync installation directory, and enter:

```
% ./sync_setup
```

2. At the Main Installation menu, choose "Configure/Upgrade a Server."

When the script asks you to enter a valid DNS name for the host, enter the alias you want to use for the server you are configuring. For example:
`syncserver.mycompany.com`.

If you do not enter an alias, the SyncServer assumes the hostname of the machine you are logged in to.

The server's configuration settings are stored in the servers directory.

Related Topics

Understanding the Custom Hierarchy

Moving Vaults

Moving SyncServers

Preparing for SyncServer Configuration

Considerations before installing a SyncServer

Before you install or upgrade your ENOVIA products for your site, you should review the following considerations:

Licensing

ENOVIA Synchronicity DesignSync requires a license before you can proceed with installing the software. For comprehensive information on obtaining, installing, and configuring licensing, see [Setting Up Licensing for DesignSync Products](#).

Secure Communications

SyncServers provide the option to use a secure communications channel for data communication. For comprehensive details about secure communications, see [Overview of Secure Communications](#).

Mirrors and Caches

We recommend that you enforce the read-only intent of mirrors and caches, which is the default behavior. For comprehensive information about this methodology, see [Introduction to Data Replication](#).

Customized Scripts and Files

If you want to create a specified area for customized scripts and files away from the installation site, set the `SYNC_CUSTOM_DIR` variable before installing the software and provide your users with the location. The users should define their `SYNC_CUSTOM_DIR` variable to point to the desired location before they use their DesignSync client software.

Tip: You can define the `SYNC_CUSTOM_DIR` in a common user profile to ensure that all users point to the correct directory.

Upgrade Warnings

The upgrade to the ENOVIA Synchronicity DesignSync Data Manager release is supported from version V6R2010 installations and later. To upgrade a pre V6R2010 installation, you must upgrade it to release V6R2010 before proceeding with the upgrade to the current version.

Note: ENOVIA recommends patching the existing installations to the highest patch level available before updating to the new release version.

You cannot revert to a previous major version of ENOVIA Synchronicity DesignSync after installing a new version. As a precaution, before upgrading, back up your `server_vault` and `server_metadata` hierarchies. If, for any reason, you need to revert to a previous version, you can restore the backed up data.

During installation, the `sync_setup` script installs a new `httpd.conf` server configuration file and backs up the original. The script then advises you to migrate any custom values. Follow the prompts in `sync_setup` to migrate your custom values and upgrade each of your SyncServers.

The installation automatically upgrades PostgreSQL's configuration, if needed. Warnings might be generated during the upgrade, but as long as they are not categorized as fatal, the PostgreSQL server will start and operate. It is recommended that the administrator review the PostgreSQL upgrade warnings and address them as needed.

Suppressing Server Version Information

You can choose to suppress the display of the Apache server information during installation through the use of directives in the `httpd.conf` file. To limit the information in the header to the absolute minimum, edit the `httpd.conf` file:

Include a new directive of `ServerTokens` with a value of `ProductOnly`.

Set the `ServerSignature` directive to a value of `Off`.

This produces the following server heading:

Server: Apache

Complete server version information can be viewed in the `error_log` file.

Disk Quotas

ENOVIA recommends NOT applying disk quotas to the user that owns the SyncServer being installed or to "syncmgr" if "syncmgr" has been assigned as the owner of the server. Using quotas as a disk space budgeting mechanism may stop a file from being checked in because a user's disk quota is exceeded, even when there is enough disk space to write the file.

Server Setup Scenarios

The following scenarios describe how to set permissions for setting up, starting, and stopping servers for an installation. The scenarios address situations where team members other than the syncmgr have responsibility for SyncServers and data management. The steps in the scenarios are generalized steps. You can use the UNIX set-user-ID (SUID) bit to enforce the read-only intent of mirrors and caches with these scenarios. See Understanding the SUID Setting, for more information.

Default Steps for All Scenarios

Projects with syncmgr Owning Everything

Projects with syncmgr Owning Servers

Projects with Project Leaders Owning Servers

Projects with a CAD Manager Managing the Location

Default Steps for All Scenarios

Follow these steps regardless of which scenario your team intends to use:

1. Create a user: `syncmgr`
2. Create a group: `projlead`
3. Make syncmgr's primary group `projlead`.
4. Unzip/tar the software as user `syncmgr`.
5. Run the installation process (using the `StartGUI.sh` or `StartTUI.sh` scripts) as the user `syncmgr`.

Projects with syncmgr Owning Everything

This is typically the way DesignSync is setup. It provides access to the data by all users, but the syncmgr account is responsible for all server maintenance and owns all data repositories. In this environment:

- Only the syncmgr can set up, start, and stop servers.
 - Only the syncmgr can customize site-wide settings.
 - All data (metadata and repository) for all the servers in an installation is stored under one directory (syncdata).
1. Ensure that syncmgr is the only member of group projlead.
 2. Set the [user:group (access)] as follows: (This may require several calls to chmod and chown.)


```
<SYNC_DIR>/bin.<ARCH>/httpd [syncmgr:projlead (0700)]
<SYNC_CUSTOM_DIR>/servers [syncmgr:projlead (0755)]
.../syncdata [syncmgr:projlead (0700)]
```

Projects with syncmgr Owning Servers

It is common to store all data for a project on a separate disk or partition. This scenario allows the server data to be stored with the rest of the project while still allowing the syncmgr to control the server. In this environment:

- Only the syncmgr can set up, start, and stop servers.
- Only the syncmgr can customize site-wide settings.
- All data (metadata and repository) for a server is stored under a project directory.

Setting up a project with syncmgr owning the servers:

1. Create a group for the project: `proja`.
2. Identify a user who will be the project leader for the project, for example, `projamgr`.
3. Make `projamgr` and all members of the project team a member of the `proja` group.
4. Make `projamgr` a member of the `projlead` group.

Note: This step is required only when you are also using the SUID option.
5. Create a directory to contain all data for the project. Set the [user:group (access)] on the project directory as [`projamgr:proja (2771)`].
6. Create a directory to contain the server data (`syncdata`) somewhere below the project directory. Set the [user:group (access)] on the `syncdata` directory as [`syncmgr:proja (750)`].

Notes:

- Do not make syncmgr a member of each project group so he can write directly into a directory under the project through the group

permissions. There is a limit to the number of groups that syncmgr can belong to.

- The Set Group -ID bit was not set on the syncdata directory because the syncmgr is not a member of the group proj_a. The s bit will not propagate if the user is not a member. Thus, files created under syncdata will have their group set to the primary group of syncmgr. This setting may be important if you are using disk quotas on the project data.

7. Run the sync_server setup as syncmgr.

8. Set the [user:group (access)] as follows:

```
<SYNC_DIR>/bin.<ARCH>/httpsd [syncmgr:projlead (0700)]
```

```
<SYNC_CUSTOM_DIR>/servers [syncmgr:projlead (0755)]
```

These access settings will allow anyone to read the server log files if there is a problem.

Note: *For scenario 1 and 2 above: DesignSync does not support SUID on httpsd mode. You cannot set up your environment to have any user start any server while only the syncmgr can set up and stop servers.

Projects with Project Leaders Owning Servers

It is common to store all data for a project on a separate disk or partition. This scenario allows the server data to be stored with the rest of the project and the project leader to set up, start, and stop the servers for his project. Use SYNC_CUSTOM_DIR when setting up this scenario to facilitate future upgrades of the software. See Section 3.2, "Using the SYNC_CUSTOM_DIR Environment Variable on UNIX," for more details.

- Only the projmgr can set up, start, and stop servers.
- Only the syncmgr can customize site-wide settings.
- All data (metadata and repository) for a server is stored under a project directory. Use SYNC_CUSTOM_DIR.

Note: You cannot use the SUID bit option on the httpsd program with this scenario. The project leader must always start and stop the server.

Setting up each project with the project leader owning the server:

1. Create a group for the project: proj_a.
2. Identify a user who will be the project leader for the project, for example: projamgr.
3. Make projamgr and all members of the project team a member of the proj_a group.
4. Make projamgr a member to the projlead group.
5. Create a directory to contain all data for the project. Set the [user:group (access)] on the project directory as [projamgr:proj_a (2770)]
6. Create a directory to contain the server data (syncdata) somewhere below the project directory. You will need to specify this path during the project

setup for the server metadata and vault data locations. Set the [user:group (access)] on the syncdata directory as [projamgr:proja (2750)].

7. Set the [user:group (access)] as follows:

```
<SYNC_DIR>/bin.<ARCH>/httpsd [syncmgr:projlead (750)]
<SYNC_CUSTOM_DIR>/sync_server_list [syncmgr:projlead
(664)]
<SYNC_CUSTOM_DIR>/servers [syncmgr:projlead (2775)]
```

8. Run sync_setup as projamgr.

Projects with a CAD Manager Managing the Location

In this scenario the CAD manager (cadmgr) untars the distribution and places it on a read-only disk. The installation is then customized by the syncmgr. The cadmgr and syncmgr usernames must be members of the same UNIX group.

Note: This scenario can be combined with any of the other scenarios above.

To do this task successfully, the cadmgr should untar the distribution on a read-write disk and run the client installation. When running the client installation, the cadmgr should accept all default answers.

After running the client installation, the cadmgr places the distribution on the permanent read-only disk for the site.

When the cadmgr has completed these steps, the syncmgr can set the SYNC_CUSTOM_DIR environment variable to a read-write disk.

For more details about the SYNC_CUSTOM_DIR variable, see Environment Settings for Users. After setting SYNC_CUSTOM_DIR, the syncmgr can run the client installation procedure or sync_setup to configure the clients or servers as necessary. During this process, the syncmgr becomes the owner of the installation and is able to update customizations on an as-needed basis.

Note: If the syncmgr forgets to set SYNC_CUSTOM_DIR, the sync_setup script will insist that the cadmgr owns the distribution and will not allow the syncmgr to continue with the customizations.

UNIX Kernel Parameters for SyncServer

SyncServers use PostgreSQL to manage various server-side metadata. PostgreSQL places specific requirements on several OS kernel parameters that scales with usage. Specifically, these parameters are related to shared memory and system semaphores

and need to be adjusted relative to the number of SyncServers that are running on a particular host.

Linux allows manual tuning and modification of kernel parameters. The maximum limit for kernel parameters can vary from version to version of some OSs. Setting the required parameters is outside the scope of DesignSync administration and should be left to the appropriate system administrator, although DesignSync provides a handy tool, `sync_config`, for calculating the minimum required values of these parameters.

The `sync_config` utility is located in the `$SYNC_DIR/bin/` directory.

If you are unsure how to set any OS-specific parameters, contact your UNIX system administrator.

Performance Tuning

Because DesignSync uses a web server to provide functionality, you may need to adjust some of the configuration parameters on the web server to match the work environment. For example, you may want to increase the default values for larger project teams, or, if you are unable to dedicate a server for DesignSync, you may want to decrease the default values to balance the system resources.

`<SYNC_CUSTOM_DIR>/servers/<host>/<port>/conf/httpd.conf` contains parameters used to configure the httpd server. In order for changes to this file to take effect, the server should be stopped (by running `stop_sync_server`) and then restarted (by running `start_sync_server`).

Four values of interest are

- MinSpareServers
- MaxSpareServers
- StartServers
- MaxClients.

These variables are described in the `httpd.conf` file.

Apache Configuration Files

DesignSync uses Apache as the base of implementation for the SyncServer. Customers who customize their Apache configuration files are encouraged to familiarize themselves with the Apache configuration changes described on the Apache organization's Web site: <http://httpd.apache.org/docs-2.0/upgrading.html>.

For example, the following directives are not supported; if they are present in the `httpd.conf` file, they will cause the server to fail to start:

- ServerType
- ResourceConfig
- AccessConfig
- Port (replaced by the Listen directive)
- SSLLog
- SSLLogLevel

Creating the User and Directory Structure for SyncServer Installation

This section lays out a set of best practices for installing SyncServers on UNIX. They are not required, but they are encouraged. Using these best practices provides you with a smoother upgrade path for future releases, and can prevent problems maintaining your SyncServer.

Install the SyncServer into its own user account

Create a separate user account (referred to as "syncmgr" in this document) to facilitate ease of system administration and to avoid accidental modification or deletion of server files by other users. This user account owns the installation hierarchy and is also used for running the server. Do not set disk quotas on the syncmgr account.

Use a separate area for the SyncServer

Use a separate area for your SyncServer installations.

Note: If you have installed the client on the machine that will host the SyncServer, you may have already unpacked the distribution into the correct location.

To set up this area:

1. Log in as syncmgr.
2. Set permissions to allow world read and execute access to the installation area.

```
%umask 022
```
3. Create an installation directory. For future upgrades, use a separate area for each ENOVIA Synchronicity DesignSync installation.
For example:

```
%mkdir -p designsync/rel_V6R2013
```
4. Change into the new directory:

```
%cd designsync/rel_V6R2013
```
5. Copy the distribution into the new directory:

```
%cp ../../<distribution_filename.tar.gz> .
```
6. Unpack the distribution:

```
% gunzip -c <distribution_filename.tar.gz> | tar -xvf
```


-
- The files are unpacked into the install sub-directory.
7. Run the client installation procedure.
 8. After the client installation has been run on your UNIX server machine, the SyncServer files are present in the designated installation directory, in a specific platform sub-directory. The platform sub-directories are as follows:
 - Linux - linux_a64
 9. Create a startup file named `.cshrc.sync` in your home directory (optional) to ensure the correct path for the DesignSync components.

IMPORTANT: After installation is complete, all users will need to follow this step. The file contains the following two lines:

```
% setenv SYNC_DIR designsync/rel_V6R2013/<platform>
% set path = ($SYNC_DIR/bin $path)
```

10. To source the `.cshrc.sync` file automatically when you log on, add the following line to your `.cshrc` file:

```
% source <user_dir>/.cshrc.sync
```
11. Verify that the environment is correctly set by issuing these commands:

```
%source <user_dir>/.cshrc
% dssc syncinfo syncDir
```

The path that you see should point to the installation directory.

Configuring the SyncServer

Overview of Configuring SyncServer

When you run the DesignSync installation, all the files for the SyncServer are placed into the installation directory in the appropriate places automatically, but this does not mean that the server is immediately ready to use.

The SyncServer requires information about your environment to provide both the basic and advanced functionality in DesignSync. The SyncServer creation and setup script, `sync_setup`, performs the following actions:

- Sets how licensing for DesignSync is managed. For more information, see [Setting Up Licensing for DesignSync Products](#).
- Sets the communications ports the DesignSync server uses.
- Sets the directories DesignSync uses for vault and metadata storage.
- Sets the administrator email contact for SyncServer problem notification.
- Sets the temporary directory for any temporary files needed by DesignSync.
- Sets whether this server will function as a Mirror Administration Server (MAS). For more information about mirror server and administration, see [Mirroring Overview](#).
- Sets whether this server will manage repositories to be mirrored to other sites as a Repository Server (RS). For more information about mirror servers, see [Mirroring Overview](#).

- Sets up the PostgreSQL server.

The setup can be performed interactively, by the system administrator, or scripted for uniformity and reuse. To more information on interactive server setup, see [Configuring the SyncServer](#). For more information on setting up scripted server setup, see [Scripting SyncServer Setup](#).

Note: Client setup and configuration is done either through the DesignSync installation process or using SyncAdmin.

Configuring the SyncServer

This is the procedure for configuring the SyncServer. To install the SyncServer, you must complete all the steps in the install script. You can re-run the script to achieve the best setup for your site. This procedure walks you through a manual SyncServer configuration. You can alternately set up a script to automatically configure your servers using a set of predefined settings. For more information on scripting server setup, see [Scripting SyncServer Setup](#).

Notes:

- Before you install the ENOVIA Synchronicity DesignSync software, make sure that you have installed all patches for the platform and that the platform is supported.
- If this is an upgrade, you must stop the SyncServer before upgrading the server.
- A SyncServer best practice is to have one user account own both the installation hierarchy and run the server. See [Server Setup Scenarios](#), for more details.
- If you want to create a specified area for customized scripts and files away from the installation site, set the SYNC_CUSTOM_DIR variable before installing the software. For more information, see the *ENOVIA Synchronicity DesignSync Data Manager Installation* in the Program Directory..

To run the DesignSync installation script:

1. Log in as syncmgr.
2. Change to your installation directory.

```
%cd /<install_path>/<platform>
```

If you have set the SYNC_DIR environment variable, issue this command:

```
%cd $SYNC_DIR
```

3. Run the setup script.

```
%./sync_setup
```

The script presents information and questions to complete a typical installation at your site. Use the following steps to help you complete the installation.

4. Verify the correct path to the installation.

If you have not set `$SYNC_DIR`, then the install script recognizes the directory it was started from as the installation directory.

5. Verify that you are logged in as the owner of the installation. If this is incorrect, type `exit` and log in as the user (`syncmgr`) who will own the installation.
6. Answer `[n]` to the question

```
"Do you want server set-up access to be determined by
UNIX group permissions? (y/n) [n]"
```

You can set UNIX group permissions for different setup scenarios at your site. The install script lists the steps necessary to do this. However, for a default installation, this is unnecessary. If you think this might be a useful feature for your site, see [Server Setup Scenarios](#).

7. The following main installation menu appears: What would you like to set up or configure?
[1] Set up default licensing
[2] Configure/Upgrade a server

[E] Exit

If you want to set up a default license for your site, choose 1, Otherwise, choose option 2.

You can exit out of the install script at any time and re-run the script to finish or fine tune your installation.

Setup Default Licensing

When you choose to set up default licensing, the install script gives you basic information about FlexNet licensing. For more details about various licensing situations, see [Setting Up Licensing for DesignSync Products](#).

To set up default licensing, follow these steps:

1. Choose 1 (Set up default licensing) from the install script main menu.
2. Answer `[y]` to:
"Would you like licensing set up for you? (y/n) [y]: "

3. Specify the way the server will find the correct licenses. Your choices are:

- [1] Specify a license file.
 - [2] Specify a directory containing license files.
 - [3] Specify the license server(s) host and port.
- [E] Exit from default license set up

If you have a single license file, select 1 and enter the full path to your license file, including the file name. The setup script copies the license file into the `<SYNC_DIR>/custom/site/config` directory and renames the file to `syncinc.lic`. (Default)

If you have a directory containing a set of license files, select 2 and enter the path to the directory containing the `.lic` license files. The files are not checked for accuracy by the selection procedure.

If you know the hostname and port number of your license server(s), and want your server to obtain your license from the server, select 3 and enter the hostname and port number in the following format: `<host>:<port>`
`[<host>:<port> <host>:<port>]`

Specifying three hostname and port number pairs provides redundancy should your primary license server not be available when needed. The installation script does not provide validation of the hostname and port number or the syntax when entered.

Important: You can specify either one hostname and port pair or three hostnames and port pairs. Note: If you select option 2 or 3, you cannot use the `start_license_server` script to start the license manager, you must manually start the `lmgrd`. For information on starting the license server, see [Starting Up and Shutting Down the License Server](#).

You can configure specific server licenses when you perform the "Configure/Upgrade a Server" task. When you have completed these steps, you are returned to the installation main menu.

Configure/Upgrade a Server

The SyncServer is an HTTP server with added capability to support DesignSync's `sync://` protocol. The server automatically loads the appropriate runtime libraries when needed. During installation you are prompted for the directory paths to the server's metadata and vault directories. The script also asks for information required to configure the Apache server, which is input to Apache configuration files. If you have not yet set up a site license and you want to use this methodology, you should do so before configuring your server. See [Setup Default Licensing](#), " to configure licensing.

DesignSync Data Manager Administrator's Guide

Note: If you are configuring more than one server, you need to re-run the install script, choose Configure/Upgrade a Server, and specify the correct port for each server.

See Server Set-up Scenarios for options for setting up server permissions.

Three dedicated TCP ports are needed per SyncServer, for its cleartext communication, optional SSL communication, and for its communication with the PostgreSQL database. If a port you specify (in the configuration steps below) is already in use, attempting to start the server will fail, with an error such as:

```
(125)Address already in use: make_sock: could not bind to
address 0.0.0.0:7100 no listening sockets available, shutting
down
```

To determine whether a port is available:

- Check whether the port number is a reserved port
`% grep {portNum} /etc/services`
- Check whether the port is already in use
`% netstat -a | grep {portNum}`

To configure server settings:

1. Choose 2 ("Configure/Upgrade a server") from the install script main menu.
2. Verify that you are the owner of the installation (syncmgr). If you are not the owner, the install script exits. You must log in as the owner to complete this step.
3. Specify the Server clear-text port. The SyncServer uses port 2647 unless you specify otherwise. For port numbers lower than 1024, the setup will prompt you to confirm the port numbers, since root access is required. If you use a root port for the SyncServer, there are some post-configuration steps require to ensure proper working functionality. For more information, see [Configuring a Server to Run on Root Ports](#).
4. Specify the SSL port. By default, no secure communication channels are configured. For more information about SSL ports see [Overview of Secure Communications](#).
5. Verify the path to the server vault. The location of the server vault directory can be set to any path that is accessible from the machine where the server will be run. The directory should be owned by the same \$USER who runs the server process. Files checked into the server reside under this directory.
6. Verify the path to the metadata directory. The location of the server metadata directory can be set to any path that is accessible from the machine where the server will be run. Do not specify a relative path to the metadata directory. The directory should be owned by the same \$USER who runs the server process. The relational database files corresponding to the server's metadata reside under this directory.

7. Verify the correct email address (syncmgr) for problem notification.
8. Specify the way the server will find the correct licenses. Your choices are:

```
[1] Use the default site license path.*
[2] Specify a license file.
[3] Specify a directory containing license files.
[4] Specify the license server(s) host and port.
[5] Use the environment variable $LM_LICENSE_FILE.
[6] Use "<path>/<licensefilename>.lic".
```

```
[E] Exit the server setup
```

```
* - A default site license has not been set up. You can
specify a default license by using sync_setup to 'Set up
default licensing'.
```

Choose option 1 to use the default site license. If you have not set up default licensing, an asterisk and a note appear next to option 1:

```
* - The default site license file does not exist. You
can specify a default license file by using sync_setup
to 'Set up licensing'.
```

If you have previously set up licensing for this server, you see the 6th option.

```
[6] Use "<path>/<licensefilename>.lic".
```

To set the default site license, you must exit the server setup, and choose Set up default licensing from the main menu. See Setup Default Licensing, for details. After you have chosen Set up default licensing from the menu, select Configure/Upgrade a Server from the main installation menu to choose the default site license.

Choose option 2 to set up a server-specific license. Specify the path to the license you want to use.

Note: When you choose this option, and after you complete the installation, you must start the specific license server.

Choose option 3 to integrate the new DesignSync product license with any existing FlexNet licenses. Specify the path to the directory containing the license files. The files are not checked for accuracy during the installation procedure.

Choose option 4 to specify the hostname and port number of your license server(s). DesignSync will query the license server for the license information. Enter the hostname and port number in the following format:
 <host>:<port> [<host>:<port> <host>:<port>]

Specifying three hostname and port number pairs provides redundancy should your primary license server not be available when needed. The

installation script does not provide validation of the hostname and port number or the syntax when entered. You must enter either one host/port pair or three host/port pairs.

Choose option 5 to use the `$LM_LICENSE_FILE` variable to refer to the `LM_LICENSE_FILE` containing the list of directories where the license files are maintained.

Chose option 6, if available, if you have already set up licensing for this server and want to use the license already defined.

Note: If you specify options 3, 4, or 5, you cannot use the `start_license_server` script to start the license manager. You must start the `lmgrd` manually. For more details on licensing and starting the license manager, see the Starting Up and Shutting Down the License Server.

9. Enter the names of other users you wish to notify about possible license expiration, and the time when you want notification sent. Entering a time before the license expiration gives you adequate time to renew licenses.
10. Specify a valid DNS host name. The default host name is displayed. This is the name that the server sends back to clients that contact it. For example, you are installing on host name `<mymachine>` and you work for a company called `<mycompany>`, the Server DNS host name is `<mymachine.mycompany.com>`.

You can use an alias for the DNS host name. This provides the flexibility to move the alias without requiring server configuration in the event of a change to the underlying hardware, such as an upgrade, or system failure. You can enter the alias at this prompt. For more details, see Moving SyncServers.

An example of an alias is: `syncserver.mycompany.com`. For more details, see Using DNS Aliases with Synchronicity Tools.

11. Specify a temporary directory.

All Platforms: SyncServers require the use of a temporary directory on a local file system to accommodate temporary files created by DesignSync. The `sync_setup` script prompts you for a value for the `TMPDIR` environment variable. The `TMPDIR` area is used for file transfer between server and client. You should use a local directory, such as `/tmp` for temporary files to optimize performance and minimize failure as a result of network issues. You can identify local file systems by running the `mount` command.

Note: An error message displays if your `TMPDIR` is not local.

Tip: For best performance, specify a local temporary file system (`tmpfs`), such as `/tmp`.

12. Specify if the server will administer mirrors. This server can be used to administer mirrors at the site (LAN) where the server is running. We call this type of server a Mirror Administration Server (MAS). All mirrors must be administered by a MAS. Enable this server as a MAS? (y/n) [n]
13. Specify if the server will be a Repository Server.

This server can be used to manage repositories that will be mirrored out to other sites. We call this type of server a Repository Server (RS) for mirrors. Enabling a server as an RS will allow notifications of changes to the data in the repositories to be sent to the mirror locations.

Enable this server as an RS? (y/n) [n]

14. If you are using the email interface, DesignSync will upgrade it to the correct version. If you have a custom version of emailinter.conf, you must update the conf file to the latest version. For more information see, [Upgrading the Email Interface](#).
15. Set up the PostgreSQL server port and password.

PostgreSQL server manages DesignSync metadata and notes database. A TCP/IP port is required to communicate with the PostgreSQL server.

Setting up PostgreSQL server ...

Enter a TCP/IP port for the PostgreSQL server. You can either accept the default value, or specify another port. Generally you can use any available port between 1025 and 65535.

PostgreSQL port number [5432]:

Set the database password. It must be at least six characters long, and you may use only alphanumeric characters and underscore. PostgreSQL password:

After the port and the database password have been specified, the installation configures the PostgreSQL server and initializes its database tables:

```
Initializing database, please wait...
>> Starting database server . . . . .
>> Creating database syncdb
>> Creating plpgsql internal language
>> Updating library path
>> Creating infrastructure tables
>> Setting database password
>> Stopping database server
```

When server configuration is complete, the script displays a success message and instructs you how to start the SyncServer.

When you have completed these steps, you are returned to the install main menu.

Choose E, (exit),

You must start the SyncServers before running client applications or scripts that use the SyncServers. For information on starting the SyncServer, see [Starting the SyncServer](#).

Configuring a Server to Run on Root Ports

For ports lower than 1024, the server must be started as root. For example, you may want to configure a server to run on the default http ports (cleartext port 80 and SSL port 443), so that users can conveniently access the server without specifying a port number.

To configure a server that will run as root:

1. Configure the SyncServer as syncmgr, not as root, as described in [Configuring the SyncServer](#), specifying the desired cleartext (and optionally SSL) ports. For port numbers lower than 1024, sync_install will prompt you to confirm the port numbers, since root access is required.

2. Enter:

```
%su - root
```

This will set the user to root, and will also set up the environment as if you had logged in as root.

3. Enter:

```
%csh
```

This switches to C shell and sources root's .cshrc file.

4. As root, execute the following commands. These can also be placed in root's .cshrc file. The commands set up the path for the server, including to tar and gzip, which are typically located in /usr/local/bin.

```
%setenv SYNC_DIR <installation_path>/syncinc  
%set path=($SYNC_DIR $SYNC_DIR/bin /bin /usr/bin  
/usr/local/bin $path)
```

Where the installation path includes the platform information, for example:

```
/home/service/DesignSync/linux_a64
```

or

```
C:\Program
```

```
Files\ENOVIA\Synchronicity\DesignSyncV6R2021x\win_b64\
```

5. If SYNC_CUSTOM_DIR is set for the syncmgr, also set it for root. For example:

```
%setenv SYNC_CUSTOM_DIR /home/syncmgr/custom
```

Also as root, add this setting to your environment:

```
%setenv SYNC_USER_CFGDIR /home/syncmgr/.synchronicity
```

Note: Both of these settings also can be placed in root's .cshrc file.

6. Change to your server's logs directory:

```
%cd $SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
```

7. As root, create these log files in the logs directory:

```
%touch error_log ssl_log ssl_misc_log access_log
httpd.pid
```

8. As root, open permissions to the log files, and to the logs directory:

```
%chmod -R 777 ../logs
```

9. If you want to configure the server as a MAS and you are using SUID, the mirror files and directories must have the syncmgr as the owner. Note: The mirror files and directories may have their group set to the group specified in the httpd.conf file (default "nobody")."

10. Still as root, start the server.

```
%cd $SYNC_DIR/bin
%start_sync_server <port>
```

11. View the server's error_log, to confirm that the server started successfully.

```
%cat
$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/error_log
```

Scripting SyncServer Setup

This is the procedure for configuring the SyncServer using scripted replies instead of stepping through the process manually. To install the SyncServer, you must complete all the steps in the install script. Scripting SyncServer setup allows you to deploy multiple servers with identical setup, reducing time to setup and risk of errors.

The user inputs for the SyncServer configuration script can be obtained from:

1. Options specified on the command line.
2. Server settings set in the XML file provided to the SyncServer setup script.
3. Site settings set in the XML file provided to the SyncServer setup script.
4. Environment Variables set on the server.
5. Default value, if one exists, or can be calculated.

If there is no other values set and a default value doesn't exist or can't be derived for the option, the script will exit without completing setup and display an error message that identifies the missing value that caused the script to fail.

Default and Calculated Values

Some values have assigned or calculated default values in the script. If no other value is specified, the default value is used.

The following options have default values:

Option	Default Value	Description
SYNC_HOSTNAME	hostname	Hostname
SYNC_DOMAINNAME	host \${SYNC_HOSTNAME}	Domain for hostname.
USER	whoami	Username
SYNC_CUSTOM_DIR	\${SYNC_DIR}/custom	Path to folder for custom installation stuff
SYNC_SITE_CUSTOM	\${SYNC_CUSTOM_DIR}/site	Path to folder for site custom stuff
SYNC_SITE_CNFG_DIR	\${SYNC_SITE_CUSTOM}/config	Path to folder for site configuration info
SYNC_ENT_CUSTOM	\${SYNC_CUSTOM_DIR}/enterprise	Path to folder for enterprise custom stuff
SYNC_ARCH	`\${SYNC_DIR}/bin/sync_arch` = "Linux"	DesignSync defined string denoting the type
SYNC_SERVER_PORT	2647	DesignSync port.
SYNC_SERVER_SSL_PORT	2649	DesignSync port
SYNC_PG_PORT	5432	PostgreSQL port
SYNC_SRV_VAULT	\${SYNC_DIR}/../../syncdata/\${SYNC_HOSTNAME}/\${SYNC_SERVER_PORT}/server_vault	DesignSync path to vault folder
SYNC_SRV_METADT	\${SYNC_DIR}/../../syncdata/\${SYNC_HOSTNAME}/\${SYNC_SERVER_PORT}/server_metadata	DesignSync path to metadata folder
AllowGrpSrvSetup	False	Allowed through the group access rights

SetupDefaultLicense	None	Set default s license
ConfigureServer	False	Need to conf (re-configure current or ne DesignSync ser
SetupSSL	False	Need to setu extra SSL po
SetupX509Certificate	None	Provide X.50 certificate for communicati you specify th value "Extern then the follo two keys, ExternalX509 and ExternalX509 can be used specify the lo of the certific
ExternalX509pem	None	The pem filer for the public version of the X.509 certific including the
ExternalX509key	None	The key filer for the privat version of the X.509 certificate, file including the
ServerMgr	\${USER}	Server mana
serverGroup	nobody	Server group
serverMgrEmail	\${USER}@"\${SYNC_DOMAINNAME}	Server mana email
SetupLicense	None	Set port licen
LicenseExWarnPeriod	7	Warn period days) when l is near to exp
DNSHost	\${SYNC_HOSTNAME}.\${SYNC_DOMAINNAME}	Server DNS
TmpDir	/tmp	Temporary directory.

MASMirrorsEnabled	false	MAS Mirrors enabled
RSMirrorsEnables	false	RS Mirrors enabled.
3DPassportServerEnabled	false	CAS Authentication enabled.
3DPassportAuthenticatedTimeout	12	CAS Authentication Session time in hours.
EnterpriseServerEnabled	false	Enterprise server enabled
StartDesignSyncServer	false	Start the DesignSync server automatically after a successful

Environment Variables

The following environment variables, if set, will be used by the system:

Note: You can overwrite the environment variables with setting in the XML file or on the command line.

Option	Default Value	Description
SYNC_DIR		Path to the DesignSync server directory
SYNC_CLIENT_SCRIPT	\${SYNC_DIR}/.syncinc.custom	Path to custom client script to source
SYNC_HOSTNAME	hostname	Hostname
SYNC_DOMAINNAME	host \${SYNC_HOSTNAME}	Domain for hostname.
USER	whoami	Username
HOME		User's home directory (~)
SYNC_USER_CFGDIR	\${HOME}/.synchronicity	Folder to store user Synch files
SYNC_USER_CONFIG	\${ SYNC_USER_CONFIG }/user.cfg	Path to

		optional user configuration file
SYNC_AUTOMOUNT	/tmp_mnt	Prefix automounter may use in paths
SYNC_CUSTOM_DIR	\${SYNC_DIR}/custom	Path to folder for custom installation stuff
SYNC_SITE_CUSTOM	\${SYNC_CUSTOM_DIR}/site	Path to folder for site custom stuff
SYNC_SITE_CNFG_DIR	\${SYNC_SITE_CUSTOM}/config	Path to folder for site configuration info
SYNC_ENT_CUSTOM	\${SYNC_CUSTOM_DIR}/enterprise	Path to folder for enterprise custom stuff
SYNC_ARCH	`\${SYNC_DIR}/bin/sync_arch` = "Linux	DesignSync defined string denoting the OS type
SYNC_JRE_DIR	\${SYNC_DIR}/jre.\${SYNC_ARCH}	Path the JRE folder.
SYNC_SERVER_PORT	2647	DesignSync server port.
SYNC_SERVER_SSL_PORT	2649	DesignSync SSL port
SYNC_PG_PORT	5432	PostgreSQL port
SYNC_SRV_VAULT	\${SYNC_DIR}/../../syncdata/\${SYNC_HOSTNAME}/\${SYNC_SERVER_PORT}/server_vault	DesignSync path to vault folder
SYNC_SRV_METADT	\${SYNC_DIR}/../../syncdata/\${SYNC_HOSTNAME}/\${SYNC_SERVER_PORT}/server_metadata	DesignSync path to metadata folder
SYNC_SERVER_3DPASSPORT		URL to the 3DPaspport authentication

		server
SYNC_SERVER_ENTERPRISE		URL to the Enterprise server

Format of the XML File

All of the parameters should be in a single XML file which is specified on the command line, or by an install script that calls the XML file as an argument. For a sample XML file, see Sample SyncServer XML file.

The XML file is composed of three primary sections:

- XML Header
- Site Settings
- Server Settings

Determining Whether a setting should be in the Site or Server Section

The XML script can be used to configure a single server or a group of servers at once. When you are configuring multiple servers, place any server-specific settings in the SetServer section, and any general settings in the SetSite section. The settings in the individual server sections override the matching setting in the site section.

For example, if all of your SyncServers use the default DesignSync ports, 2647/2649, then place the port setting configuration lines in the Site section, like this:

```
<SetSite>
...
<SetVariable name="SYNC_SERVER_PORT">
    <SetInteger value="2647"/>
</SetVariable>
<SetVariable name="SYNC_SERVER_SSL_PORT">
    <SetInteger value="2679"/>
</SetVariable>
...
```

```
</SetSite>
```

If specific servers use a different port, but in general the default port is used, you can set the default port in the Site setting section, and then for specific servers that don't use the default port, define the port within the server settings for the individual servers.

If every server uses a different port number, you would not set the port number in the Site setting section, you would set the port in each of the server setting sections.

XML Header

The XML file starts with an XML header identifying the XML version and encoding and the DesignSync version (also called the Level in the Program Directory).

```
<?xml version="1.0" encoding="utf-8" ?>
<SetConfig release="version">
```

Site Settings

The site setting section begins and ends with a SetSite open/close tag set:

```
<SetSite> [...] </SetSite>
```

The site settings are settings that are common to all servers being configured. For more information on when to place the variable in the Site Settings or the Server settings, see Determining Whether a setting should be in the Site or Server Section. In between the SetSite tags, the Site settings can take a few different variable forms. For variables that use a path name, you set the value with Setpath, as shown below:

```
<SetVariable name="EnvironmentVariable">
<SetPath value="path">
</SetVariable>
```

For variables that are numbers, such as port numbers, set the value with SetInteger, as shown below:

```
<SetVariable name="SYNC_SERVER_PORT">
    <SetInteger value="PortNumber"/>
</SetVariable>
```

For variables that are string values, set the value with SetString, as shown below:


```
<SetVariable name="SetupDefaultLicense">
    <SetString value="File"/>
</SetVariable>
```

Server Settings

The server setting section begins and ends with a SetServer open/close tag set.

```
<SetServer> [...] </SetServer>
```

Server settings, are for specific server configurations. For more information on when to place the variable in the Site Settings or the Server settings, see Determining Whether a setting should be in the Site or Server Section. Each SetServer section should begin with the name of the server.

```
<SetServer>
<SetVariable name="SYNC_HOSTNAME">
<SetString value="servername"/>
</SetVariable>
```

Like site settings, server settings can take a few different variable forms. For variables that use a path name, you set the value with Setpath, as shown below:

```
<SetVariable name="Setting">
<SetPath value="path">
</SetVariable>
```

For variables that are numbers, such as port numbers, set the value with SetInteger, as shown below:

```
<SetVariable name="Setting">
    <SetInteger value="PortNumber"/>
</SetVariable>
```

For variables that are string values, set the value with SetString, as shown below:

```
<SetVariable name="Setting">
```

```
<SetString value="StringValue"/>
```

```
</SetVariable>
```

Running SyncSetup Script

This is the procedure for configuring the SyncServer. Unlike the procedure described in Configuring the SyncServer, this procedure focuses on specifying a configuration file containing some or all of the settings for the sync server.

Notes:

- Before you install the ENOVIA Synchronicity DesignSync software, make sure that you have installed all patches for the platform and that the platform is supported.
- If this is an upgrade, you must stop the SyncServer before upgrading the server.
- A SyncServer best practice is to have one user account own both the installation hierarchy and run the server. See Server Setup Scenarios, for more details.

1. Log in as syncmgr.
2. Change to your installation directory.

```
%cd /<install_path>/<platform>
```

If you have set the SYNC_DIR environment variable, issue this command:

```
%cd $SYNC_DIR
```

3. Run the setup script using the -config option to specify the XML file. Any settings that you wish to pass to the server that are not specified in the configuration XML file, can be specified as key/value pairs on the command line.

IMPORTANT: If any options are missing; meaning then are not specified on the command line, present in the file, or set as environment variables prior to running the script, the server setup fails.

```
%./sync_setup -config <config>.xml [<key>=<value>[ ... ]]
```

For example, this command uses the ServerSetup.xml configuration file and, after the command completes successfully, will start the server as indicated by the StartDesignSyncServer.

```
:
$./sync_setup -config ServerSetup.xml
StartDesignSyncServer=True
```

Related Topics

Sample SyncServer Scripted XML Installation File

Configuring the SyncServer

Configuring a Server to Run on Root Ports

Sample SyncServer Scripted XML Installation File

This is a sample SyncServer Scripted XML file using the R2019x server. To use this file as the basis your own scripted server, please do not copy the text from this file, instead copy the uncommented SampleScriptedSyncServerInstallationFile.xml (\$SYNC_DIR/share/content/doc/mergedProjects/SyncAdmin/SampleScriptedSyncServerInstallationFile.xml) the name and location of your choice, where it can be made available for modification and use.

Tip: You can revision control your modified version to track changes as you evolve your version.

The XML file starts with an XML header identifying the XML version and encoding and the DesignSync version (also called the Level in the Program Directory. This sample file uses R2019x as the version.

```
<?xml version="1.0" encoding="utf-8" ?>
<SetConfig release="R2019x">
    <SetSite>
        <SetVariable name=" SYNC_DIR">
            <SetPath
value="/home/tvio11/R420.Linux1/syncinc/linux_a64"/>
        </SetVariable>
    <SetVariable name="SetupDefaultLicense">
        <SetString value="File"/>
    </SetVariable>
    <SetVariable name="SiteLicensePath">
```

```
                <SetPath
value="/home/tvio11/lic/syncinc.lic"/>
                </SetVariable>
                ...
</SetSite>

<SetServer>
                <SetVariable name=" SYNC_HOSTNAME">
                        <SetString value="walvrh55mon"/>
                </SetVariable>
<SetVariable name=" SYNC_SERVER_PORT">
                        <SetInteger value="30014"/>
                </SetVariable>
                <SetVariable name="SYNC_ SERVER_SSL_PORT">
                        <SetInteger value="30015"/>
                </SetVariable>
                ...
</SetServer>

<SetServer>
                <SetVariable name="SYNC_HOSTNAME">
                        <SetString value="walvrh54mon"/>
                </SetVariable>
<SetVariable name="SYNC_ SERVER_PORT">
                        <SetInteger value="30016"/>
```

```
</SetVariable>

<SetVariable name="SYNC_ SERVER_SSL_PORT">
    <SetInteger value="30017"/>
</SetVariable>

...

</SetServer>

...

</SetConfig>
```

Post-Configuration Tasks

Starting the SyncServer

When you have completed the install script, you need to start both the license server and any SyncServers you have configured. The `start_sync_server` program starts the server process (`httpsd`). This server process must always be run by the same `$USER` (`syncmgr` in this document). The `$USER` who starts the `httpsd` process owns all the files created and modified by the server. The default owner (`user`) permissions are set to read/write, with read-only permissions for everyone else.

Notes:

- You must run `start_sync_server` for each SyncServer; when `start_sync_server` prompts you for a port number, you must enter a single port number.
- You can start more than one SyncServer automatically during system boot time, by customizing the sample Synchronicity server startup script included with the distribution. The sample script is:
`$SYNC_DIR/share/examples/S90syncinc`

Read the header information in this file for information on how to customize this file.

To start the servers, follow these steps:

1. As `Syncmgr`, if you have not already done so, start the license server. On the license server machine, enter:
`% start_license_server -syncserver <hostname>:<port>`
where `-syncserver` specifies information that the script uses to locate the

license file. For example:

```
% start_license_server -syncserver
ASICprojectserver:30038
```

NOTE: The license server must reside on the machine specified in the SERVER line of the license.

The script starts the FlexNet license manager daemon (lmgrd) which reads the license.

2. As syncmgr, start the SyncServer. On the SyncServer machine, make sure the <SYNC_DIR>/bin directory is in your directory path and enter:

```
%start_sync_server [port number].
```

The start_sync_server script:

- Defines the environment variable LM_LICENSE_FILE as the path for syncinc.lic. If the LM_LICENSE_FILE variable is already defined, the script adds the syncinc.lic path to the end of that definition.
 - Starts the specified SyncServer. NOTE: Not all start up messages display on your screen; however, all start up information is stored in an error_log file.
3. To review the last entries in the error_log file, type the following:

```
% tail
<SYNC_CUSTOM_DIR>/servers/<machine>/<port>/logs/error_log
```

Upgrade Considerations

After upgrading a server to the latest version, you should take the following steps before using DesignSync.

1. All users should clear their browser cache before accessing a newly upgraded server. Otherwise, cached data can cause unexpected results or errors on certain panels.
2. Similarly, server administrators should clear their JRE cache in their user application area. Some of the Administrator panels have been re-implemented and will not work with the old, cached JAR files.

On Windows, open the Java Control Panel from the Control Panel and click on "Delete Files" in the "Temporary Internet Files" section on the "General" tab. Select "Downloaded Applets" and then OK.

On UNIX, the JRE cache usually resides in your home directory, in the \$HOME/.java/deployment/cache/javapi/v1.0/jar/ folder.

NOTE: DS administrators are *highly* encouraged to send a reminder to all users about the browser and JRE cache clearing requirements.

3. The upgrade of specific DesignSync, hcm modules, or module vault data to modules can be performed at any time after the server is configured. Please ensure that all users check in their modifications and release any locks prior to a module upgrade. The upgrade procedure does not preserve user locks during the upgrade.
4. If you are upgrading an installation that used the DesignSync Data Manager for Eclipse Integration, you need to update your Eclipse plug-in after you install DesignSync. For information on updating the DSclipse plug-in, see the *ENOVIA Synchronicity DesignSync Data Manager Installation* in the Program Directory.

Upgrading the Email Interface

If you are using the email interface and have a custom version of emailinter.conf, you must update your conf file to the latest version. You can perform this step either before or after configuring your SyncServer.

To upgrade your email configuration file:

1. Move or copy your emailinter.conf file to a backup location or name, for example:

```
> mv $SYNC_CUSTOM_DIR/site/share/config/emailinter.conf  
$SYNC_CUSTOM_DIR/site/share/config/emailinter.bak
```
2. Copy the emailinter.conf file included with the release to the custom area.

```
> cp $SYNC_DIR/share/config/emailinter.conf  
$SYNC_CUSTOM_DIR/site/share/config/emailinter.conf
```
3. Review the customizations in your backup file and add them, as appropriate to the emailinter.conf file.

Upgrading Email Note Triggers

If you upgrade a system prior to V6R2010 on which Email Note Triggers was configured, you must perform the following additional configuration step:

Any registry key in the server or site registry containing a reference to Email trigger compiled tcl code (*Trigger.tbc) should be changed to reference uncompiled tcl code (*Trigger.tcl).

For example the key:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Triggers\Server\hcmNoteAttach\FileName="hcmNoteAttachTrigger.tbc"
```

Must be changed to:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Triggers\Server\hcmNoteAttach\FileName="hcmNoteAttachTrigger.tcl"
```

Configuring SUID (UNIX only)

Understanding the SUID Setting

This section presents the issues and conclusions behind the SUID methodology.

Understanding the Issue with the SUID Settings

DesignSync mirror and cache directories implement sharing methodologies that rely on the contents of the directories being read-only for all users. When files are not checked out with a lock, users have symbolic links in their work areas to shared files in the mirror/cache directory, and the permissions for files in the mirror/cache are read-only. However, the user who causes a file to be fetched into the mirror/cache is that file's owner. This file ownership issue enables a user to change -- accidentally (through a tool such as Emacs) or deliberately -- the file permissions and to edit files in the mirror/cache.

Solving the SUID Issue

The only way to prevent users from modifying or deleting files in a common directory, such as a mirror or cache directory, is to limit ownership of the directory and its contents to a specific user, called "syncmgr" in this documentation. If only syncmgr has write access to the contents of the mirror/cache directory, there is no way for other users to modify or delete any files.

This approach, however, would not work with the current DesignSync implementation because users would not be able to fetch files into the cache/mirror directory.

The solution is to take advantage of UNIX's set-user-ID (SUID) bit, which lets a user create and access files and directories as a different user; this allows a DesignSync user to perform the mirror/cache fetch operations as syncmgr.

How SUID Works

A UNIX process has two user IDs associated with it: the real user and an effective user. In general, all files created by a process are owned by the effective user. In most cases, the real and the effective user IDs are the same. If the SUID bit is set on the file permissions of an executable file, the effective user of the process executing the file is the owner of the executable file, not the person who invoked the process. You set the SUID bit using the chmod command as follows:

```
% chmod u+s <filename>
```


SUID Applied to DesignSync

If the executables that update the cache/mirror (DesignSync, syncd, and so on) are owned by syncmgr and the SUID bit is set, then the effective user id of the process is syncmgr irrespective of who invoked the executable. Therefore, files that are fetched into the mirror/cache directory by a user have ownership and permissions as though they were fetched by syncmgr.

By itself, the SUID mechanism is not sufficient, because files in the users' work areas should retain the ownership and permissions of the user; only mirror and cache files need to be owned by syncmgr. Fortunately, UNIX provides a system call, `setuid(uid)`, that can switch the effective user between syncmgr and the real user. When a user executes syncd, DesignSync, or one of the other SUID executables, the process will have the effective user id of syncmgr (the owner of the executable on disk). The process itself can then switch between the user and syncmgr as the effective user:

1. At startup, the effective user is set to the real user as is appropriate for all normal activity.
2. Whenever a file has to be fetched into the mirror or cache directory ('ci -mirror', 'co -share', and so on), the effective user is set to syncmgr. The files created in the cache/mirror are therefore owned by syncmgr.
3. When the write is complete, the effective user is set back to the real user. Links to the mirror/cache that are created in the user's work area are therefore owned by the real user.

Enabling SUID for DesignSync

The DesignSync installation allows you to enable SUID during the initial installation procedure. If you did not enable SUID when you installed DesignSync, use the following procedure to manually enable SUID for the installation.

Note: This procedure does not enable SUID for cache and mirror directories. To enable SUID for cache and mirror directories, see [Enabling SUID for Specific Mirror or Cache Directories](#). The procedures may be performed in any order.

Important: The file system on which DesignSync is or will be installed must have SUID enabled. Refer to the man page for the 'mount_nfs' command for details:

```
% man mount_nfs
```

Note: If you are running on Linux, you must stop all the clients and shut down syncd using 'syncdadmin close' before running the `suidenable` script. In addition, if the server administers mirrors (MAS) or is a repository server for another mirror (RS), the

server must also be shut down. Otherwise, 'suidenable' will not be able to modify the in-use executables.

To enable SUID for the DesignSync server:

1. If you are retrofitting an existing installation, stop the SyncServer:

```
% stop_sync_server <port>
```
2. Install the SyncServer as syncmgr in your chosen directory path. Doing so ensures that all related executables are owned by syncmgr.
3. You should have already defined SYNC_DIR and added SYNC_DIR/bin to your PATH when installing the software.
 Enter the following command:

```
% suidenable [-syncdir <path>]
```

Use the '-syncdir' option to suidenable if the value of the SYNC_DIR variable should not be immediately resolved. For example, let's say you prepare a software distribution configured with your company's default settings, for all sites within your company to use for their software installations. SYNC_DIR is defined as the symbolic link /nfs/syncinc. The link /nfs/syncinc is guaranteed to exist at every site, but the value the link resolves to will vary for each site. You want suidenable to preserve the unresolved /nfs/syncinc link. Without the '-syncdir' option, 'suidenable' will resolve the /nfs/syncinc link to the path that link points to for your site, at the time suidenable is run.

For suidenable to apply to the /nfs/syncinc link, run:

```
% suidenable -syncdir /nfs/syncinc
```

If your installation is already configured to use SUID, any new service packs installed will run suidenable on the same paths that previously had suidenable run on them. To retrofit an existing SUID installation, run the same 'suidenable -syncdir' command as above. Once the '-syncdir' option is used, subsequent service pack installations will run suidenable on the unresolved link (the value that was specified to '-syncdir').

suidenable sets SUID bits on the following executables:

- syncd
- DesignSync
- stcl (not stcl)
- dssc (not dss)
- syncfisrvpp

If you move the location of your SYNC_DIR, suidenable must be rerun.

Restrictions:

- `suidenable` must be run by the owner of installation.
 - `suidenable` must be run when no server/client processes are currently running.
 - Installation paths are limited to 256 characters.
4. If you want to define specific mirror or cache directories now, continue to section 10.2.2, otherwise start the SyncServer (as `syncmgr`).

```
% start_sync_server <port>
```

Enabling SUID for Specific Mirror or Cache Directories

This procedure recommends setting directory permissions to 755, which is the most straightforward approach to enabling proper mirror and cache access because it guarantees that all users will have read/execute access to the cache (or mirror). However, this approach may not provide the level of security you need. You can choose to implement more restrictive permissions on cache directories by leveraging UNIX groups and the set-group-ID bit. For more information, see the Setting Permissions on the Cache. Note that when using the SUID bit, you can also apply the set-group-ID methodology to further restrict access to mirror directories.

Once DesignSync is installed, all cache directories and mirror directories are created by `syncmgr` with the correct permissions: 755 when SUID is enabled, 777 when SUID is not enabled.

Important: The file system on which DesignSync is or will be installed must have SUID enabled. Refer to the man page for the 'mount_nfs' command for details:

Note: You can create project specific caches, in which only members of a project team have permission to access its cached data. See Setting Permissions on the Cache and Project-Specific Configuration for more information about setting cache permissions, and how to set up a project cache.

```
% man mount_nfs
```

To Enable SUID for Specific Mirror or Cache directories:

1. If the SyncServer is running, stop the SyncServer:

```
% stop_sync_server <port>
```
2. If DesignSync has not been installed, create the cache directory as `syncmgr`, for example:

```
% cd /usr1/Projects/Caches  
% mkdir ASIC_Cache
```
3. If `syncmgr` does not already own any existing cache or mirror directories, change the owner to `syncmgr` as follows:

```
% cd <parent_directory_of_cache_or_mirror_directory>  
% chown -R syncmgr <cache_or_mirror_directory>
```

For example:

```
% cd /usr1/Projects/Caches
% chown -R syncmgr ASIC_Cache
```

Note: Typically you must be logged in as root to execute the chown command.

4. Set the directories containing the mirrors and caches to be writable only by owner as follows:

```
% find <cache_or_mirror_directory> -type d -exec chmod
755 {} \;
```

For example:

```
% find ASIC_Cache -type d -exec chmod 755 {} \;
```

Note: The find command sets the permissions of the top-level mirror or cache directory and all subdirectories to be writable by only syncmgr.

5. Start the SyncServer (as syncmgr).

```
% start_sync_server <port>
```

Disabling SUID

If you have enabled SUID but do not require the functionality, you can disable it.

Note: This procedure assumes that the \$SYNC_DIR environment variable exists and is pointing to the correct SyncServer installation.

Disabling SUID:

1. If the SyncServer is running, stop the SyncServer:


```
% stop_sync_server <port>
```
2. Disable SUID for the DesignSync client executables. The executables are located in the \$SYNC_DIR/bin and \$SYNC_DIR/jre/bin directories for your operating system.


```
% cd $SYNC_DIR/bin.<OS>
% chmod u-s stlc dssc syncd syncfsvpp
% cd $SYNC_DIR/jre.<OS>/bin
% chmod u-s DesignSync
```

For example:

```
% cd $SYNC_DIR/bin.sol2
% chmod u-s stlc dssc syncd syncfsvpp
% cd $SYNC_DIR/jre.sol/bin
% chmod u-s DesignSync
```

3. Set the directories containing the mirrors and caches to be writable by everyone:

```
% chmod 777 <cache_or_mirror_directory> [...]
```

For example: `% chmod 777 IC_Cache`

Notes: If your system requires a different security setting, use the security settings required by your system however be aware of the impact it may have on DesignSync in the Overview section.

4. Start start the SyncServer (as syncmgr).
`% start_sync_server <port>`

Troubleshooting SUID

Problem: Your attempt to start a client application (DesSync, syncd, stcl, etc.) fails.

Your attempt to start a client application (DesSync, syncd, stcl, etc.) fails with this error:

```
fatal: libole42.so: open failed: No such file or directory
```

Cause: Ensure that the links to your SUID installation are correct.

Note: You should change the <version> in the installation path when you upgrade DesignSync.

DesignSync Administrator (SyncAdmin)

ENOVIA DesignSync Administrator Tool Overview

The ENOVIA DesignSync Administrator tool (SyncAdmin) is a graphical user interface that lets system administrators, project leaders, and users easily configure ENOVIA Synchronicity DesignSync® Data Manager clients and SyncServers for a site, a project, or individual use.

The DesignSync Administrator is part of the DesignSync software distribution and is available on both UNIX and Windows platforms. The options in this tool do not apply to DesignSync DFII.

To use the DesignSync Administrator, you select whether you want to change the DesignSync site, project, or user settings:

- If you indicate that you want to change the site settings, SyncAdmin reads and displays the current settings of the site registry file.
- If you are using project settings (having set `SYNC_PROJECT_CFGDIR`) and indicate that you want to change the project settings, SyncAdmin reads all of the applicable registry files, letting you override the settings in your project registry.
- If you indicate that you want to change the user settings, SyncAdmin reads in all of the applicable registry files, letting you then override those settings in your user registry.

After you make your configuration preferences using SyncAdmin, the tool writes your preferences to the specified registry file.

Important: Windows 7 provides updated administrative features that might require the user running SyncAdmin making site-wide changes to be logged in with Administrator privileges before running SyncAdmin to make the changes.

The first time you start SyncAdmin, it will open on the General pane.

Note: In order for DesignSync client changes you make with SyncAdmin to take effect, you must restart the DesignSync client (DesSync graphical interface, `syncd`, `dssc`, or `stcl`). In order for changes made to the server registry files to take effect, you must restart your SyncServer.

As an administrator installing DesignSync software, you answer questions that define many installation-wide setup defaults. As you answer these questions, information is written to the site registry file. Using the SyncAdmin tool, you can make changes to the registry without the risk of file corruption. Any changes that you make using SyncAdmin

(in the Change Site Settings mode) replace the site-wide settings for all DesignSync clients and SyncServers on your LAN. Some LAN settings are only available on UNIX.

Command Line Interface

SyncAdmin tasks are also available through the command line by using the sregistry and SyncAdmin commands on UNIX.

Site-Wide Configurations

As a UNIX LAN administrator, you must have permission to write to the site-wide registry file, <SYNC_SITE_CNFG_DIR>/SiteRegistry.reg to make LAN-wide changes with SyncAdmin. (<SYNC_SITE_CNFG_DIR> resolves to <SYNC_SITE_CUSTOM>/config, which, in turn, resolves to <SYNC_CUSTOM_DIR>/site/config.) Without write permission to this file, the choices available in the SyncAdmin tool are limited to those that affect projects or individual DesignSync clients (users).

Project Configurations

You can use SyncAdmin to create a project and specify its vault and cache directory. As a project leader, you can use SyncAdmin to set up a project for your team. You create a project configuration directory to store the ProjectRegistry.reg file. Team members must set the <SYNC_PROJECT_CFGDIR> environment variable to the project configuration directory to access the project settings.

User Configurations

As a user, you can make changes to your individual work environment using SyncAdmin. If you do not have permission to write to the site-wide registry file, you will only be able to use the SyncAdmin tool to define individual preferences such as your browser and editor.

Related Topics

Registry Files

Executing SyncAdmin from UNIX

Executing SyncAdmin from Windows

Configuring a Multi-User Environment

Understanding the Custom Hierarchy

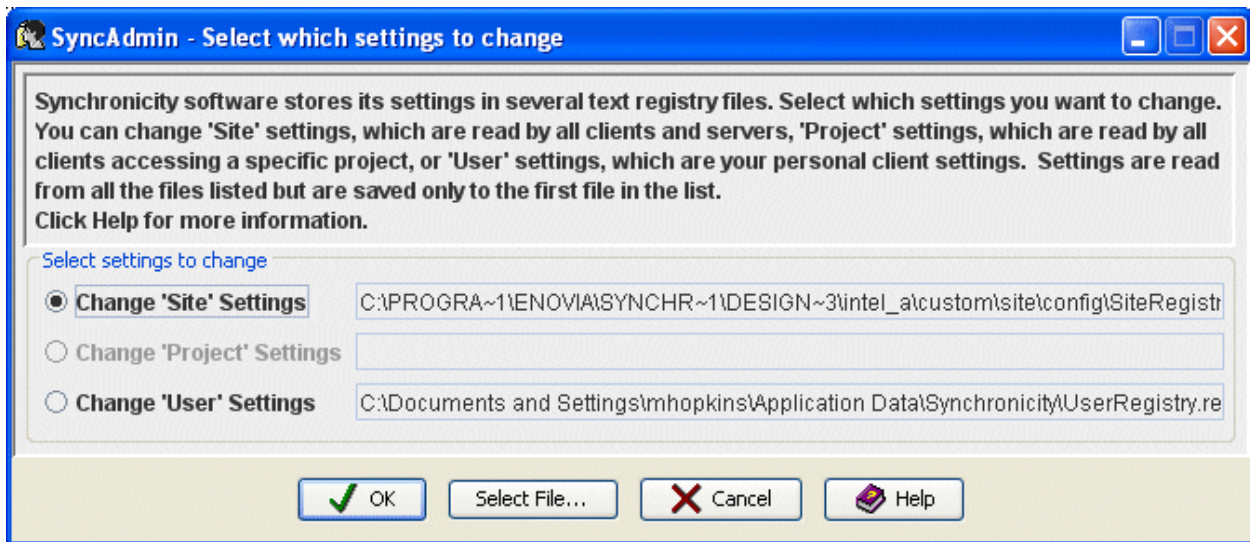
DesignSync Environment Variables

Setting Up a LAN Cache

Select Registry

The SyncAdmin tool can be executed from either Windows or UNIX. When you make selections and then click **OK**, your selections are written to a registry file.

Click on the image for more information about each field.



Change 'Site' settings

If you select this option, SyncAdmin stores registry settings in the site registry file. All DesignSync clients and SyncServers read the site registry file. Use the site registry file to store common site-wide settings. The site registry is stored in the `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` file where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`.

Change 'Project' settings

As a LAN administrator, you can use this option to create a new registry file or write to an existing one. As a project leader, you can use this field to specify the project registry file (`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`). If you specify a registry file with this field, DesignSync reads this registry file after reading the site registry file, but before reading the user registry file.

If the enterprise registry file (`EntRegistry.reg`) exists in the `<SYNC_ENT_CUSTOM>/config` field, DesignSync reads this registry file before reading the site registry file, the project registry file, and the user registry file.

Change 'User' settings

If you select this option, SyncAdmin reads and displays registry settings from the following registry files scanned in the order listed:

- The DesignSync registry file: `<SYNC_DIR>/share/SyncRegistry.reg`
- The enterprise registry file:
`<SYNC_ENT_CUSTOM>/config/EntRegistry.reg` only if you have set the `<SYNC_ENT_CUSTOM>` environment variable and the `EntRegistry.reg` file exists.
- The site registry file: `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` where `<SYNC_SITE_CNFG_DIR>` resolves to `<SYNC_SITE_CUSTOM>/config` which, in turn, resolves to `<SYNC_CUSTOM_DIR>/site/config`
- The project registry file:
`<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg` only if you have set the `<SYNC_PROJECT_CFGDIR>` environment variable and have created a `ProjectRegistry.reg` file.
- The user registry file: `<SYNC_USER_CFGDIR>/UserRegistry.reg` where `<SYNC_USER_CFGDIR>` defaults to `<HOME>/synchronicity` on Unix and `%AppData%\Synchronicity` on Windows.

If any of the displayed values are changed, then those changes are stored back in the `UserRegistry.reg` file.

OK

Clicking on the **OK** button accepts the registry file(s) information you selected and brings up the SyncAdmin tool.

Select File (Windows only)

Clicking **Select File** allows you to specify a registry file that is not shown for selection. For example, if you are the installation owner, and the owner of a SyncServer configured from that installation, you may want to specify the server's `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/PortRegistry.reg` file. Server specific settings can also be applied from ProjectSync's Administer Server.

Cancel

Clicking on the **Cancel** button closes the SyncAdmin tool.

Help

Clicking on the **Help** button displays the help for the SyncAdmin tool

Related Topics

Registry Files

General

General Options

The **General** options pane in the **Synchronicity Administrator** window allows a:

- LAN administrator to modify the system-wide settings chosen by the system administrator during the installation of the DesignSync software.
- DesignSync user to override a subset of settings that were specified during installation of DesignSync software or by a LAN administrator using SyncAdmin.

These settings define the way DesignSync clients (the DesignSync graphical interface, syncd as used by dss and stcl, dssc, and stclc) operate when users check in and check out project files.

When you have made your selections and you click **OK** or **Apply**, settings are written to the appropriate registry files and inherited by all users on your LAN.

Click on the image for more information about each field.

Check out read only when not locking
 Show collection size in bytes
 Warn me when an object is excluded
 Map operations on collection members to owner

Command Reference Help Mode: Module-based mode ▼

Minimum comment length:

Default cache directory: Browse...

HTML browser:

File editor:

Use File editor for comment entry

Check out read only when not locking

This option controls whether checking out files (checkout without lock) places read/write or read-only copies in your work area. This option also affects check-in operations when you select the **Keep an unlocked copy** option in DesignSync. The `mkmod` command's `-checkin` operation leaves behind unlocked copies, so obeys the **Check out read only when not locking** setting.

Show collection size in bytes

For collection objects, the Size column displays the total number of member files in the collection. You can change it to display the size in bytes.

Warn me when an object is excluded

If you have specified files or expressions in an exclude filter, DesignSync will notify you when you perform an operation that excludes the selected files. If you do not want to display the message, then uncheck the **Warn me...** box.

Enable CR/LF processing (Windows Only)

This option is only available on Windows. By default, a SyncServer substitutes CR/LF for `^M` characters when a text file is checked in from a client running Windows. It is possible for a UNIX client to check the files out again without further cleanup. When a

file is checked out to a Windows client, the SyncServer substitutes ^M characters for CR/LF.

As a System Administrator you may have the following scenario at your site:

1. Windows-specific files are checked in by a Windows client, from a mounted disk on a UNIX file system.
2. Data checked by Windows users in step 1 is mirrored to a directory on a UNIX file system. (For more information, see [Mirroring Overview](#).)
3. The files in the mirror directory are then operated on by Windows tools.

Turn the Enable CR/LF Processing button off to retain the ^M characters found in files. (For more information, see [Managing Revision Control of Files Between Windows and UNIX](#).)

Map operations on collection members to owner

DesignSync does not perform revision control operations individually on collection members. Instead, DesignSync operates on the collection itself. By default, if you select a collection member and attempt to apply a revision control operation, DesignSync skips the object and warns that the object is not versionable. If you select the **Map operations on collection members to owner** button and DesignSync attempts to operate on a collection member during a revision control operation, DesignSync determines the member's owner collection and operates on the collection as a whole. The **Map operation on collection members to owner** setting applies to the following revision control commands: `ci`, `co`, `tag`, `cancel`, `unlock`, and `retire`.

Command Reference Help Mode

DesignSync features the ability to tailor the ENOVIA Synchronicity Command Reference to your site usage by selecting a working methodology. When you select a working methodology, the Command Reference displayed when you select help within the command interface is tailored to method you selected.

- All modes - links to the All version of the ENOVIA Synchronicity Command Reference. This version contains all commands and options for DesignSync.
- Module-based mode - links to the Module version of the ENOVIA Synchronicity Command Reference. This version contains only the commands and options for DesignSync modules usage.
- File-Based mode - links to the File version of the ENOVIA Synchronicity Command Reference. This version contains only the commands and options for DesignSync file/vault-based usage.

Note: This does not affect the help you see when links within the help system are linked. The help links always link to the All version.

Minimum comment length

As an administrator, you can require DesignSync users to enter a comment of a given length when checking in project files, creating new module versions, or creating module branches. This feature is useful when you want to force the methodology of requiring check-in comments.

To require a check-in comment, type or use the scroll bar to set a comment length between 1-999 characters.

To remove a check-in comment requirement, set the field to 0.

The comment-length requirement does not include leading and trailing white space, or the portion of your check-in comment that comes from the Revision Log. (See *DesignSync Data Manager User's Guide: Revision Control Properties.*)

Default cache directory

During the installation of your DesignSync software, the system administrator defines a default cache directory. On UNIX, this value is automatically seeded in the **Default cache directory** field in the **Synchronicity Administrator** window. All users in the installation inherit this default cache directory. The default cache can be used to share read-only files in a UNIX environment. On both Windows and UNIX, the DesignSync software also uses the cache to store temporary files.

As a LAN administrator, you can change the cache directory that will be used to share files within your own LAN. To change the cache directory, click in the text box and enter the pathname to a directory that is accessible from your network. You can use the **Browse** button to navigate through your local directory structure. (For more information, see *DesignSync Data Manager User's Guide: What is a LAN Cache?*)

Notes:

The following restrictions apply to the cache directory:

- Cache directories cannot be a subdirectory of another cache directory.
- All users of the cache must have read and write permissions to the cache directory. (For more information, see *Setting Permissions on the Cache.*)
- The **Default cache directory** field is editable in "site" mode, but read-only in "user" mode.

HTML Browser

DesignSync applications use an HTML browser for purposes such as displaying data sheets and online help. On UNIX platforms during the installation of DesignSync software, the system administrator defines the installation-wide default HTML browser.

All users inherit this defined browser. On Windows platforms, each individual system's default browser is used.

As a LAN administrator, you can change the default browser for UNIX users on your LAN by clicking in the HTML browser text box and entering the name of a different browser. All users on your LAN will inherit the new default browser.

Note:

DesignSync provides a script called nsremote, located in the \$SYNC_DIR/bin directory.

This script calls the default browser with the -remote option so that browser requests connect to an active process if one exists instead of always starting a new process. Consider specifying nsremote as your HTML browser. The default browser for nsremote is Firefox.

You can modify the nsremote script to change certain behaviors, such as whether each browser request opens a new browser window (the default) or reuses an existing window.

See the comments in the nsremote script for instructions.

File editor

During the installation of DesignSync software, the system administrator defines an installation-wide default file editor. All DesignSync users inherit this default. When you double click on a file in the DesignSync GUI, the file will automatically open in the default file editor. In a UNIX environment, the default editor is automatically seeded in the **File editor** field in the **Synchronicity Administrator** window.

As a LAN administrator, you can change the default file editor for all DesignSync users on your LAN. To do so, click in the **File editor** text box and enter the pathname to the executable of a different file editor. All users on your LAN will inherit the new default file editor.

Individual DesignSync users can also change the default file editor by executing the SyncAdmin tool and entering an editor of their own choice.

If the path to your file editor contains spaces, enclose the path in quotation marks. However, do not include arguments within the quotation marks. Examples of valid file editor entries include:

```
"C:\Program Files\Windows NT\Accessories\wordpad.exe"
```

```
"/usr/cad tools/xterm" -e vi
```

Use File Editor for Comment Entry

When specifying a comment for DesignSync commands that use an interactive comment interface, such as ci, co, rollback, sitr submit, sitr release, and url setprop, you

can specify using the defined editor for comments instead of entering your comment interactively at the command line.

Note: If you plan to enter multibyte characters, use a editor that supports UTF-8 characters.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (<SYNC_USER_CFGDIR>/UserRegistry.reg, which defaults to <HOME>/ .synchronicity/UserRegistry.reg) on UNIX or the Windows registry. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (<SYNC_USER_CFGDIR>/UserRegistry.reg, which defaults to <HOME>/ .synchronicity/UserRegistry.reg) on UNIX. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin online help is displayed.

Related Topics

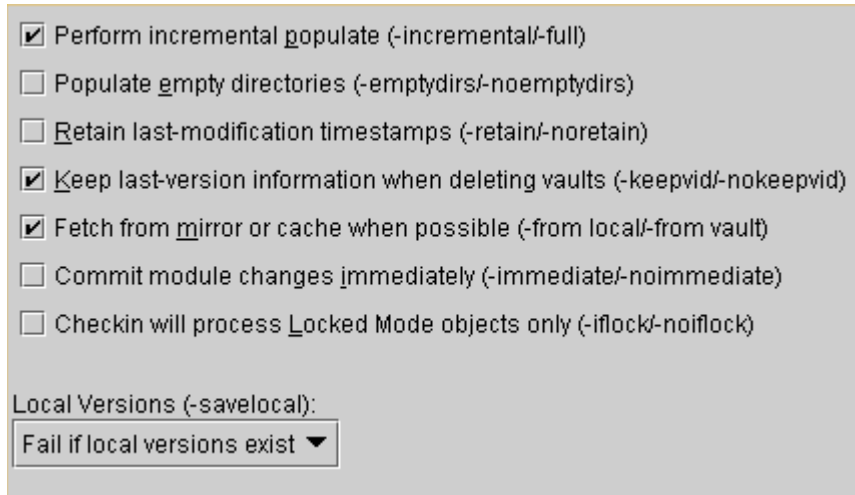
[SyncAdmin Tool Overview](#)

Command Defaults

Using the **Command Defaults** options pane of SyncAdmin, a system administrator can set the default behavior for DesignSync commands within the GUI interfaces..

Note: DSCC does not use the default set in SyncAdmin. Those defaults are set within the user interface separately. The DSDFII interface uses some of the options here to set

the underlying behaviors of the DSDFII system. The documentation for the specific options below notes which options are used by DSDFII.



The screenshot shows a list of seven options for the populate command, each with a checkbox and a label:
- Perform incremental populate (-incremental/-full)
- Populate empty directories (-emptydirs/-noemptydirs)
- Retain last-modification timestamps (-retain/-noretain)
- Keep last-version information when deleting vaults (-keepvid/-nokeepvid)
- Fetch from mirror or cache when possible (-from local/-from vault)
- Commit module changes immediately (-immediate/-noimmediate)
- Checkin will process Locked Mode objects only (-iflock/-noiflock)
Below the list is a section labeled "Local Versions (-savelocal):" with a dropdown menu currently set to "Fail if local versions exist".

Note: When using the GUI interface, the command defaults system is disabled. For commands run from a command line interface, even within the GUI, default values saved via the command line defaults system override default behavior specified by SyncAdmin.

Perform incremental populate (-incremental/-full)

This option determines whether to perform incremental populates or to always perform full populates. An incremental populate traverses only those folders whose corresponding vault folders have been modified. A full populate traverses the entire vault folder hierarchy.

If you are updating mirrors, use the incremental option. When you specify the full option, mirror updates may take a long time to complete.

Note: This setting is used by the DSDFII interface.

Populate empty directories (-emptydirs/-noemptydirs)

This option determines whether empty directories are retained or removed when populating a directory. By default, this setting is not enabled; therefore, the populate operation removes empty directories.

Note: Users can override the default setting by selecting **Populate empty directories** on the Populate dialog. The **-emptydirs** or **-noemptydirs** command line options for the command **populate** can override a given default SyncAdmin setting.

Retain last-modification timestamps (-retain/-noretain)

This option determines whether the operation retains the last-modified timestamp of the object or uses the time that the operation took place. The cancel, check-in, check-out, and populate operations obey this setting. By default, this option is not enabled; therefore, the timestamp of the local object is the time of the operation.

Notes:

- Users can override the default setting by selecting **Retain last modification timestamps** on the Populate dialog. The **-retain** or **-noretain** options for the command **populate** can override a given default SyncAdmin setting.
- This setting is used by the DSDFII interface.

Keep last-version information when deleting vaults (-keepvid/-nokeepvid)

This option retains the vault information of a object after you delete the object.

Notes:

- Users can override the default setting by selecting **Keep latest version information when deleting vaults** on the Populate dialog. The **-keepvid** or **-nokeepvid** options for the command **populate** can override a given default SyncAdmin setting.
- This setting is used by the DSDFII interface.

Fetch from mirror or cache when possible (-from local/-from vault)

This option specifies whether the object is fetched from the vault, or copied from the site's mirror or cache directories. This option enables a performance optimization specific to the 'co -lock', 'co -get', 'populate -lock', and 'populate -get' commands.

Commit module changes immediately (-immediate/-noimmediate)

This option controls how module remove or rename operations in the workspace are processed by default. When unchecked, **-noimmediate**, DesignSync queues remove and rename operations in the workspace until the next module checkin operation (Default). This reduces the number of module versions created, and minimizes confusion when a large number of files are processed. When checked, **-immediate**, DesignSync immediately performs the remove or rename operation.

Users can override the default setting in the:

- GUI by checking or unchecking the **Apply changes locally first; commit with next checkin** option for the remove or rename operation.
- command line by specifying the **-immediate** or **-noimmediate** option for the command.

Note: This setting is used by the DSDFII interface.

Commit will process Locked Mode objects only (-iflock/-noiflock)

This option controls the default behavior of object checkin. When unchecked, -noiflock, DesignSync processes the entire workspace looking for modified files (Default). This prevents checkin from missing any files that were modified, but not checked out of source control with a lock. When checked, -iflock, DesignSync uses a list in the metadata of all the module structural changes and all the files that are locked in the workspace and checks in only the objects listed.

Users can override the default setting in the:

- GUI by checking or unchecking the **Only process locked objects** option for the checkin operation.
- command line by specifying the **-iflock** or **-noiflock** option for the command.

Local Versions (-savelocal)

When a collection object is checked out or populated to a user's workspace, the check-out or populate operation first removes local versions of the object from the workspace. Then the operation fetches the object being checked out or populated (with the local version number it had at the time of checkin).

The **Local Versions** option specifies the action that the check-out or populate operation takes with modified local versions in a user's workspace (other than the current, or highest numbered, local version). (DesignSync considers a local version to be modified if it contains modified members or if it is not the local version originally fetched from the vault when the collection object was checked out or populated to your workspace.)

Select a value:

- **Save local versions.** If a user's workspace contains a local version other than the local version being fetched, the check-out or populate operation saves the local version for later retrieval. For information on retrieving local versions that were saved, see the ENOVIA Synchronicity Command Reference: **localversion** command.
- **Delete local versions.** If a user's workspace contains a local version other than the local version being fetched, the check-out or populate operation deletes the local version from your workspace.
- **Fail if local versions exist.** If a user's workspace contains an object with a local version number equal to or higher than the local version being fetched, the check-out or populate operation fails. This is the default action.

Note: If a user's workspace contains a local version with local version number lower than the local version being fetched and if that local version is not in the vault, the check-out or populate operation saves the local version. This behavior occurs even when you specify **Fail if local versions exist**.

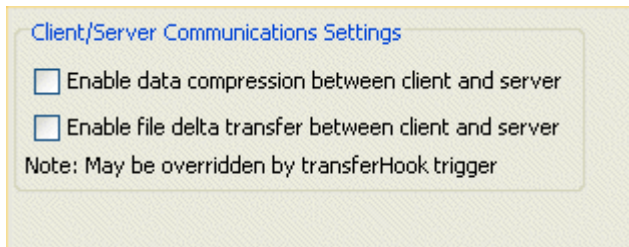
Related Topics

DesignSync Data Manager User's Guide: Command Line Defaults System

Communications

Using the **Communications** options pane of SyncAdmin, a system administrator can:

- Enable data compression.
- Enable file delta transfer between the client and the server.



Enable data compression between client and server

DesignSync has the ability to compress data before it is transferred between the client and SyncServer. In general, compressing files improves the speed of data transfers by reducing the file size. File compression also increases the security of your data transfers. As a LAN administrator, you can decide how file compression will be handled for files being transferred on your LAN by selecting or deselecting **Enable data compression between client and server**.

Note:

If your data consists of large binary files and your network is fast, it may take more time to compress the data than the amount of time that you save by transferring the smaller files.

Enable file delta transfer between client and server

File delta transfer compression is a compression mechanism that relies on determining and transferring the file differences between the file version in the workspace and the one on the server. On the receiving end the original file is restored by applying the difference to a copy of the base file, rather than transferring the entire file. In cases

where differences are small, or on a slower network, this mechanism can result in a drastic reduction of the transfer size, resulting in enhanced performance.

Delta transfer can be used on most DesignSync file types including modules, module members, collection objects, and file-based objects.

When enabled, checkin, checkout, and populate attempt to use delta transfer during operation.

The direct fetch performance optimization must be turned off in order for delta transfer to be used.

When disabled, delta transfer compression is not used for any operations.

Notes:

- For checkin to use delta transfer, a copy of the file must exist in the file cache, for example, if `populate` or `checkout` were used with the `-share` mode. This means that Windows operating systems will typically not use delta transfer.
- Populate uses the vault to extract the 2 versions to calculate the delta and then sends over the delta to the client. However, the client workspace must contain an unmodified version of the file being fetched.
- This setting can be overridden by using the `transferHook` trigger. For information on using the `transferHook` trigger or the `transferDeltaHook`, see [Tuning Communications between Client and Server](#). For general information on defining a trigger, see [Add or Edit a Trigger](#).

Related Topics:

[Data Compression](#)

[Delta Transfer Compression](#)

[Tuning Communications between Client and Server](#)

Default Fetch State Options

In DesignSync, project files are checked in and checked out by various members of a project team. Based on your project team's design methodology, you may want all team members to check in and check out objects with the same default fetch state.

As a user, you can override the LAN default by specifying an object's state each time you check in, check out, populate, or cancel design objects. (See [DesignSync Data Manager User's Guide: Object States](#) for additional information.)

To define the default fetch state for all users on the LAN, choose from the following options available under **Default fetch state**:



Note: For commands run from a command line interface, default **-keep** or **-get** options saved via the command line defaults system overrides the default fetch state specified by SyncAdmin.

- Unlocked copies

This option leaves unlocked copies of files in your local work area.

- References to versions

When a file is checked in from your local work area, DesignSync leaves a reference (or pointer) to the version of the object that is in the vault. These files are only visible to DesignSync

Note: Do not select this option when working with Cadence or Synopsys data. In the reference state, an object has only DesignSync metadata in the workspace and no corresponding files exist on disk. Since neither Cadence nor Synopsys tools read the DesignSync metadata, referenced objects will not be recognized by those tools.

- Links to cache (available only to Administrator)

If your project employs a file cache to share files, this option creates links from your work area to objects in the cache.

- Links to mirror (available only to Administrator)

If a mirror is defined for your project data, this option creates links from your work area to objects in a mirror directory.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

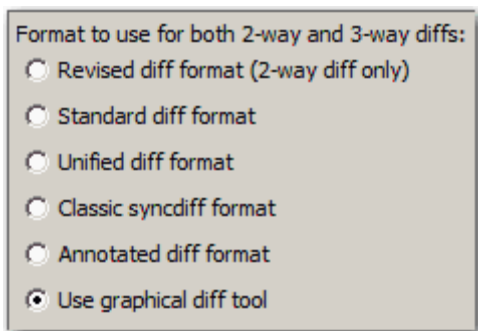
When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

DesignSync Data Manager User's Guide: Command Line Defaults System

Diff Format Options

Using the **Diff Format** options pane of SyncAdmin, you can select the display option format for all of the common diff operations. This also defines whether to invoke the built-in graphical diff viewers when running diff operations from within the DesignSync GUI. The graphical diff viewers are used by default. Note that a saved Display Options value in the Advanced Diff dialog does not override SyncAdmin's Diff Format setting. The saved Display Options value in the Advanced Diff dialog only controls that field for subsequent invocations of Advanced Diff.



- **Revised diff format** displays an annotated file showing added, deleted, and updated lines with markers and text colors. Color indicators can be specified with the Customize Diff Options page. The Revised diff format is only used for 2-way diffs, such as **Show Local Modifications** and **Compare Original to Latest** operations. Revised diff format is also used by the Advanced Diff's **Compare 2 Files/Versions** when two unique files are selected, as opposed

to two versions of the same file. If Revised diff format is selected as the default and a 3-way diff is performed, such as by **Compare to Latest**, the **Annotated diff format** is used instead.

- **Standard diff format** displays only lines that have been added, removed, or changed. The results appear as they would if you used the UNIX diff command. The Standard diff format is not applicable to 3-way diffs. If the Standard diff format is the default Diff Format for 2-way diffs, 3-way diffs will use the Classic syncdiff format.
- **Unified diff format** displays the changes in the context where the diff occurs in the file. The Unified diff format is not applicable to 3-way diffs. If the Unified diff format is the default Diff Format for 2-way diffs, 3-way diffs will use the Classic syncdiff format.
- **Classic syncdiff format** displays the diff as it was provided in earlier releases of DesignSync.
- **Annotated diff format** displays both lines that are the same in both files and lines that have changed. Lines that have been added, removed, or changed are indicated and color-highlighted. The annotations are displayed in the Diff View.
- **Use Graphical diff Tool** displays the results of the diff in DesignSync's built-in graphical Diff Viewer. (Default) For information on defining a custom GUI tool, see Diff Format.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

Advanced Diff

Diff Options

Exclude Lists

The Exclude Lists pane lets you specify directories and files that should always be excluded from revision-control operations that support exclude filters (the **-exclude** option for DesignSync command-line commands and the **Exclude Filter** field for GUI operations). The mkmod command does not have a **-exclude** option, but does utilize the **Exclude Lists**, for the mkmod command's `-checkin` option.

Note: Using the command line defaults system to save a default value for the **-exclude** option pertains only to the **-exclude** command line option. **Exclude Lists** are unaffected by the command line defaults system.

The global exclude lists apply to revision control operations such as `ci`, `co`, `populate`, `tag`, and `mkbranch`. The global exclude lists do not apply by default to commands that list objects, such as `ls`, `vhistory` (**Tools=>Reports=>Version History**), `compare` (**Tools=>Reports=>Compare**), and `contents` (**Tools=>Reports=>Contents**). For these types of listing commands, exclude objects in the global exclude lists by specifying a null string ("") using the **-exclude** option or the **Exclude Filter** field. To exclude objects from the List View, select **View=>Hide Excluded Objects**.

The Exclude Lists pane has two different views:

- Site and Project Exclude Lists pane (accessible through SyncAdmin)
- User Exclude Lists pane (accessible using DesignSync (**Tools=>Options=>General=>Exclude Lists**))

Objects Excluded by DesignSync Tools by Default

By default, DesignSync excludes the following objects :

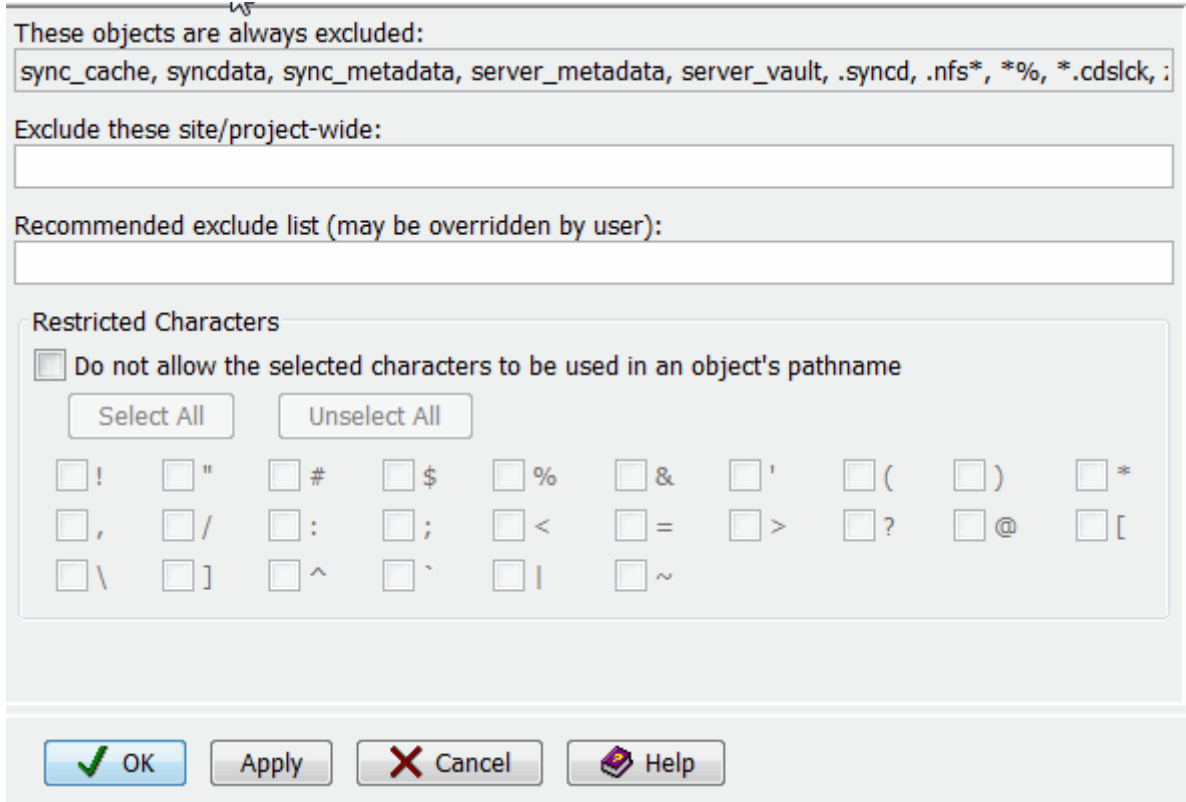
```
sync_cache, syncdata, sync_metadata, client_vault,  
server_metadata, server_vault, .syncd, *.cdslck, zpcellSCRATCH,  
zcleSCRATCH*, .nfs*, *%
```

These objects display in the **These object are always excluded** field. This field is not modifiable.

Site and Project Exclude Lists

The Exclude Lists pane shown below displays in SyncAdmin. The fields in the Exclude Lists pane allow wildcards. Separate items with commas.

Click on the image for more information about each field.



These objects are always excluded:
 sync_cache, syncdata, sync_metadata, server_metadata, server_vault, .syncd, .nfs*, *%, *.cdslck, ;

Exclude these site/project-wide:

Recommended exclude list (may be overridden by user):

Restricted Characters

Do not allow the selected characters to be used in an object's pathname

Select All Unselect All

! " # \$ % & ' () *
 , / : ; < = > ? @ [
 \] ^ ` | ~

OK Apply Cancel Help

These objects are always excluded

This field displays the objects DesignSync excludes by default. This field is not modifiable.

Exclude these site/project-wide

If you are a project leader with Administrator privileges, you can exclude objects on a site-wide scope for revision-control operations. Enter the file types you want to exclude in this field.

Recommended exclude list

Enter the files types you want to set as the default user-defined exclude list. Users can override this list individually by selecting **Tools=>Options=>General=>Exclude Lists** in DesignSync and entering values in the **Exclude these objects** field.

Restricted Characters

The site administrator can restrict the characters available to use in object and object path names. All the characters that can have special meanings can be restricted from use. To restrict characters, click the checkbox next to "Do not allow the selected

characters to be used in an object's pathname" and select any characters you want to restrict.

IMPORTANT:

- Administrators whose users use DSDFII should not restrict the use of the # character.

User Exclude Lists

The Exclude Lists pane shown below displays from DesignSync if you select **Tools=>Options=>General=>Exclude Lists**. The fields in the Exclude Lists pane allow wildcards. Separate items with commas.

Click on the image for more information about each field.

These objects are always excluded:
sync_cache, syncdata, sync_metadata, server_metadata, server_vault, .syncd, .nfs*, *%, *.cdslck, ;

These objects are excluded site/project wide:

Exclude these objects:

Restricted Characters

Do not allow the selected characters to be used in an object's pathname

Select All Unselect All

! " # \$ % & ' () *
 , / : ; < = > ? @ [
 \] ^ ` | ~

OK Apply Cancel Help

These objects are always excluded

This field displays the objects DesignSync excludes by default. This field is not modifiable.

These objects are excluded site/project wide

This field displays the objects your project leader or Administrator has excluded using SyncAdmin. This field is not modifiable when the Exclude Lists pane is in user mode. If you are a project leader or Administrator and have permissions to modify the project or site registry settings, you can edit this field by selecting **General=>Exclude Lists** from SyncAdmin.

Exclude these objects

Enter the objects you want to exclude from revision-control operations. The values you enter override the recommended defaults set by your project leader or Administrator.

Restricted Characters

The client user can restrict the characters available to use in object and object path names. All the characters that can have special meanings can be restricted from use. To restrict characters, click the checkbox next to "Do not allow the selected characters to be used in an object's pathname" and select any characters you want to restrict.

IMPORTANT:

- DSDFII users should not restrict the use of the # character.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (<SYNC_USER_CFGDIR>/UserRegistry.reg, which defaults to <HOME>/synchronicity/UserRegistry.reg) on UNIX or the Windows registry. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (<SYNC_USER_CFGDIR>/UserRegistry.reg, which defaults to <HOME>/synchronicity/UserRegistry.reg) on UNIX, and %AppData%\Synchronicity on Windows. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin online help is displayed.

Related Topics

DesignSync Data Manager User's Guide: Command Line Defaults System

Exclude Restricted Characters (RestrictedCharacters and EnforceRestrictedCharacters)

Exclude Lists (SiteFilter and Filter)

Logging Options

The **Logging** options pane lets you:

- Enable logging upon startup
- Enable logging of command results
- Indicate where log files are to be stored

As an administrator, you can change the logging settings for all users on a LAN or for all users of a particular project, depending on whether you have SyncAdmin modify the site registry or a project registry. See *Executing SyncAdmin from UNIX* for information about specifying a site or project registry to modify.

You can use environment variables to standardize the location of your team's log files. For example, you can specify `$SYNC_USER_CFGDIR/synclogs` as the log file directory so that each user's log files are stored in the same relative location, in this case `$HOME/.synchronicity/synclogs`. You can also choose to store all users' log files in a central location by setting the log directory to `<fileserver>/logs/$USER`, for example. See *Using Environment Variables* for more information.

Users can change these logging settings within DesignSync by selecting **Tools=>Options=>Logging** or typing the `log` command. The changes users make to the logging settings are stored in the user registry, which has precedence over the settings of the project and site registries.

Click on the image for more information about each field.

The screenshot shows a configuration window with the following elements:

- Two checked checkboxes:
 - Logging should automatically begin at startup
 - Record both commands and command results
- A section titled "Put log files in this directory:" containing a text field with the path `/home/tjames/test` and a "Browse..." button.
- A section titled "Initial log file" containing two radio buttons:
 - Create a unique log file name
 - Always log to this file:
- A text field below the radio buttons containing the filename `dss_session.log`.

Logging should automatically begin at startup.

If you select this check box, DesignSync automatically begins logging users' sessions when they invoke DesignSync. The log file records commands, and optionally, command results in DesignSync, stcl, stclc, dss, dssc. Logging is enabled by default.

Record both commands and command results.

If you select this check box, DesignSync writes out both the commands users execute and the command results as displayed in DesignSync's output window. If you do not select this option, only commands are recorded in users' log files. DesignSync records both commands and command results by default.

Put log files in this directory

Specify where you want users' DesignSync log files to be placed. If you do not specify a directory in this field, then DesignSync puts the log file in each user's home directory (as defined by `$HOME` on Unix or the user profile on Windows). If you clear this field and click OK or Apply, the user's home directory is automatically reselected as the default log directory.

Note: Users can override the default setting by selecting **Put log files in this directory** on the Populate dialog. The **-defaultdir** options for the command **log** can override a given default SyncAdmin setting.

Initial log file

Specify if you want to:

- create a new and unique log file for each session

If you select "Create a unique log file name," DesignSync automatically creates a log name for you with the following format: **dss_<date>_<time>.log**. For example, **dss_07061998_175103.log** is a log file created on July 6, 1998 at 5:51 PM.

To prevent large numbers of **dss_<date>_<time>.log** files from collecting in your log directory, DesignSync automatically deletes log files that are older than 2 days. If more than 20 log files remain, DesignSync deletes all log files older than 1 hour. If you want to retain a log file indefinitely, you must move or rename the file.

- use the same log file name for each session.

If you select to "Always log to this file," enter the file name you want to use for the log file in the field. If you select this option, DesignSync always create the log file with the name you specify (or accept) in this edit box. You will be prompted to remind you that each time it creates the file, it overwrites the older file of that name.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (`$SYNC_USER_CFGDIR/UserRegistry.reg`, which defaults to `$HOME/.synchronicity/UserRegistry.reg`) on UNIX or the Windows registry. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (`$SYNC_USER_CFGDIR/UserRegistry.reg`, which defaults to `$HOME/.synchronicity/UserRegistry.reg`) on UNIX. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Modules Options

Using the **Modules** options pane of SyncAdmin, a system administrator can:

- Specify how to handle legacy modules without hierarchical references.

- Specify how many versions of a module to display, when browsing a server.

Change traversal mode with static selector on top level module

Fetch legacy modules with no hierarchical references as modules

Activate legacy hcm command set

Maximum number of versions displayed:

Retry on module checkin failure

Number of checkin retry attempts:

Checkin retry time interval:

Note: For commands run from a command line interface, default values saved via the command line defaults system override default behavior specified by SyncAdmin.

Change traversal mode with static selector on top level module

Specifies how the module hierarchy is processed by DesignSync operations when using the hrefmode "Normal," with a static selector on the selected module.

When this mode is disabled (default), the operation traversing the hierarchy resolves the href selectors of the selected module, switching to static mode for the remaining hierarchical references if a static selector (such as a version tag) is reached on a submodule.

When this mode is enabled, the operation traversing the hierarchy resolves the selector of the selected module. If the selector is static, all hierarchical references resolve statically from that point. If the selector is dynamic, the traversal continues in normal mode.

For more information on understanding href mode processing, see the *ENOVIA Synchronicity DesignSync Data Manager User's Guide: Module Hierarchy*.

Fetch legacy modules with no hierarchical references as modules

Specifies how legacy modules that do not have any hierarchical references are represented in the local workspace, when initially fetched by the **populate** command with the **-target** option. The default behavior is to represent such modules as modules. If this option is unselected, the fetched module will be represented as a folder, without associated module information. This option only applies to data that is initially fetched by the **populate** command with the **-target** option.

If hierarchical references are later added to a module that was initially fetched as a folder, users must re-run the **populate** command with both the **-target** and **-recursive** options. That will update the workspace information, to represent the hierarchical module in the workspace as a module.

Activate legacy hcm command set

Specifies whether to enable the legacy HCM command set. When legacy HCM is enabled, only legacy module commands use the hcm prefix. "Modern" module commands use the command, with no prefix. For information about using legacy hcm, see the *ENOVIA Synchronicity DesignSync HCM User's Guide*. By default, this is disabled.

Maximum number of versions displayed

How many versions of a module to display, when browsing a module on a server. By default, the last 50 versions on a module branch are shown.

Retry on Module Checkin Failure

When retry is enabled for a checkin, and the Module Checkin Retry on Failure Trigger Hook is not enabled, these settings determine how many retry attempts will be made and the time interval is between the retries.

By default, DesignSync attempts 60 retries.

By default the interval between retry attempts is 900 seconds (15 minutes).

These settings can also be controlled by using the corresponding registry keys, Checkin Error Retry Attempts (ModuleFailureRetryAttempts) and Checkin Error Retry Interval (ModuleFailureRetryInterval).

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

DesignSync Data Manager User's Guide: What is a Module Cache?

Procedure for Setting Up a Module Cache

DesignSync Data Manager User's Guide: Command Line Defaults System

ENOVIA Synchronicity Command Reference: populate Command

Module Cache Tab

Using the **Module Cache** options pane of SyncAdmin, a system administrator can set the default module cache paths and mode for fetching from a module cache. You can define a setting for the default module cache path or mode so that the setting applies to all users at a site or users working on a project. Individual users can also define these settings for their own use.

Default module cache paths:

Allow non-existing module cache paths

Allow linking to submodules in the module cache

Verify module cache has no overlapping modules

Default module cache mode

Link to modules in the module cache

Copy modules from the module cache

Fetch modules from the server

Default module cache paths

Identifies the path to one or more module caches. The **populate** and **mcache** operations use this path to locate the module cache when a user does not specify it with the **-mcachepaths** option.

In this field, specify a list of space-separated path names. Path names can be absolute or relative.

Note: To specify a path that includes spaces, surround the path containing the spaces with curly braces. For example: `/dir1/cache {/dir2/path name}`.

To override the default module cache paths setting, specify the **populate** command with the **-mcachepaths** option.

Allow non-existing module cache paths

Specifies the behavior of **populate** when it encounters an invalid module cache path. By default, when **populate** encounters an invalid module cache path, it stops processing the command and returns an error. If this option is selected, processing skips the object that experienced the error and continues processing the command.

Allow linking to submodules in the mcache

Specifies whether submodules are linked in the **mcache**.

When not enabled, submodule links are not enabled into the **mcache**, so users can only link to the top level module populated in the **mcache**. (Default)

When enabled, links to the hierarchical references are stored in the **mcache**, so that a user can link to a submodule from the top-level module cache.

Verify Module cache has no overlapping modules

Specifies whether to verify on module cache creation that the module cache will not contain overlapping modules. Modules in the module cache cannot contain overlapping modules as this interferes with proper processing. By default, this option is not enabled, meaning that the admin must be aware of the issue and not set up overlapping modules in the module cache.

Default module cache mode

Identifies the method (mode) that the **populate** operation uses to fetch modules when users do not specify the **-mcachemode** option. Click the mode you want. Available modes are:

- **Link to modules in the module cache.** For each module it finds in the module cache, the **populate** operation sets up a symbolic link from your

work area to the base directory of the module in the module cache. **Note:** This mode is the default behavior on UNIX platforms only. On Windows platforms, the default behavior is copy.

- **Copy modules from the module cache.** For each legacy module it finds in the module cache, the **populate** operation copies the module to your work area.

Notes:

- This mode is the default behavior on Windows platforms.
- This mode is ignored for non-legacy modules, which are fetched from the server.
- **Fetch modules from the server.** Causes the **populate** operation to fetch modules from the server.

To override the default module cache mode setting, specify the **populate** command with the **-mcachemode** option.

Performance Options

Using the **Performance** options pane of SyncAdmin, a system administrator can optimize DesignSync's performance by:

- Setting the number of client threads DesignSync uses

By default, DesignSync uses multiple threads to speed up check-in, check-out, and populate operations. In this way, while one client thread is waiting for a server operation to complete, another client thread can be performing client-side operations.

To increase DesignSync performance, you can use the SyncAdmin Performance tab to set the number of threads that DesignSync uses in addition to the main thread. You can also choose to turn off multiple-thread processing.

- Controlling communication time outs

You can safeguard against client/server communication time outs by setting the maximum time for the period of inactivity between the client and server.

- Controlling the number of retries to a server in maintenance mode

When the DesignSync server is performing a backup operation or a data import or export, the server automatically switches to read-only mode to protect the integrity of data.

You can use the SyncAdmin settings to control the number of attempts to retry sending data modification requests to the server, automatically resending the requests after a specified interval.

- Enabling data compression between client and server

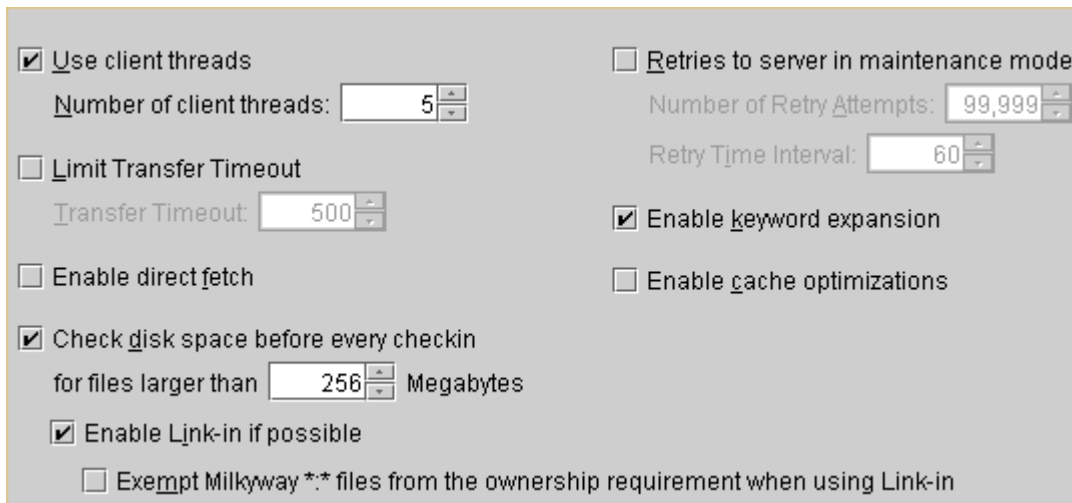
By default, data compression is turned off to optimize transfer speed.

- Turning off keyword expansion

To optimize DesignSync's performance, you can turn off keyword expansion if you do not use keywords in your files. By default, DesignSync expands all keywords that are used in your files.

- Enabling cache optimizations
- Fetching files directly into a user's work area
- Checking for available disk space before checking in files greater than a specified size
- Enabling performance optimizations by creating links when checking in large files

Click on the image for more information about each field.



The screenshot shows the SyncAdmin settings interface with the following options and values:

- Use client threads
Number of client threads: 5
- Retries to server in maintenance mode
Number of Retry Attempts: 99,999
- Limit Transfer Timeout
Transfer Timeout: 500
- Retry Time Interval: 60
- Enable direct fetch
- Enable keyword expansion
- Enable cache optimizations
- Check disk space before every checkin
for files larger than 256 Megabytes
- Enable Link-in if possible
- Exempt Milkyway *.* files from the ownership requirement when using Link-in

Use Client Threads

This setting controls whether DesignSync uses multiple threads (in addition to the main thread) to carry out client operations. By default, this setting is turned on, specifying multiple threads. (You can specify the number of additional threads in the **Number of Client Threads** field.) If you turn off this setting, DesignSync uses only the main thread for client operations. This setting applies to all DesignSync clients at your site if you are using SyncAdmin to modify the site registry.

Number of Client Threads

When the **Use Client Threads** setting is turned on, this field lets you select the number of client threads DesignSync uses in addition to the main thread to carry out client operations. Specify a number from 1-10. The default number is 5.

Limit Transfer Timeout

This setting controls the length of time DesignSync clients wait for their server connection to be restored before timing out. By default, this setting is off, as DesignSync's behavior is to never time out. This behavior may result in a client's hanging if its server connection is lost. When the server connection is restored, DesignSync operations resume automatically.

Transfer Timeout

When the Limit Transfer Timeout setting is turned on, this field lets you set the maximum time for the period of inactivity between the client and server. The default value is 500 seconds. This amount of time represents the maximum time between sending/receiving two TCP/IP packets. Specify a number from 1 - 500, but be aware that too low a number may result in DesignSync exiting unnecessarily.

Note:

If you are using a Windows client and you turn on Limit Transfer Timeout, you need to also edit the ConnectTimeout registry setting (see the Registry Settings for Client/Server Communication for details).

For Windows clients, the ConnectTimeout registry setting takes precedence over the Limit Transfer Timeout value.

If you are using a UNIX client, you do not have to change the ConnectTimeout registry setting.

Retries to Server in Maintenance Mode

This setting controls the processing of DesignSync requests when the DesignSync server is in read-only mode. The DesignSync server automatically switches to read-only mode when performing a backup operation or a data import or export to protect the integrity of data. When this setting is on, DesignSync retries the request at the specified Retry Time Interval for the specified Number of Retry Attempts. By default, this setting is off.

If the server has not returned to a write state after the specified number of retries, the modification requests fail and return errors to indicate that the server may be experiencing non-maintenance related problems.

Number of Retry Attempts

When the Retries to Server in Maintenance Mode setting is turned on, this field lets you set the number of retry attempts. The default value is 99,999 attempts.

Retry Time Interval

When the Retries to Server in Maintenance Mode setting is turned on, this field lets you set the wait time between retry attempts. The default value is 60 seconds.

Enable Keyword Expansion

DesignSync expands all keywords that are used in your files. By default this setting is on. Turn off keyword expansion if you do not use keywords in your files.

When you turn off keyword expansion, the "key expansion" and "retain timestamp" options are ignored when you check in and check out files under revision control. Use this optimization only if you do not use keys in your files.

Note: The check-out and check-in operations detect binary files and Cadence collections and do not expand keywords when operating on these objects, even if the **Enable Keyword Expansion** setting is on.

Enable cache optimizations

Use this setting to disable cache reference counting. When this is disabled, you will need to use `cachescrubber` to remove any version in the cache that is not being used. (See [Cleaning a Cache](#) for details.)

You cannot enable this setting on Windows clients.

Enable direct fetch

Use this setting to optimize DesignSync performance by fetching objects directly into users' work areas.

By default, DesignSync writes a fetched file to the `.SYNC` metadata directory under a temporary name. When the entire file is successfully transferred from the vault, DesignSync renames the file and saves it in the user's work area. This approach ensures that no corrupted information is written into the work area. In the event of a network or disk problem during the transfer, the fetch is rolled back and no data is written to the work area.

If you opt to fetch files directly, DesignSync transfers the files into a work area without copying and renaming them. Thus populate, check-out, and other fetch operations

perform more quickly. However, if your system is unstable, you risk data corruption in users' work areas when they fetch files directly.

Note: This mode is automatically disabled for Populate with merge operations.

Check disk space

Use this setting when you need to check-in large files. When this option is enabled, DesignSync checks whether the vault has sufficient disk space for a large file and fails quickly if the space is insufficient. In the scrolling field, you can specify the minimum size in megabytes that defines a large file.

You cannot enable this setting on Windows clients.

When you enable this option, DesignSync checks the following requirements before each check-in:

- Whether the vault has sufficient disk space for a large file.
- Whether the workspace is on the same file system as the vault. (This check is needed when you enable link-in of large files.)

If this check succeeds and you have enabled link-in of large files, DesignSync attempts to use hard links when checking in large files.

See Setting Up Link-In for details on how to set up link-in of large files.

Enable Link-in

Use this setting when you have configured DesignSync to take advantage of the link-in performance optimization that controls the check-in of large files. When this option is enabled, DesignSync creates hard links to large files that a user checks in. The owner of the file is changed from the user to syncmgr when the hard link is created.

You cannot enable this setting on Windows clients.

When you enable this setting, **Check disk space** also must be enabled.

Your system administrator must set up your server to handle link-in. See Setting Up Link-In for details on how to set up link-in of large files.

Exempt Milkyway *.* files

IMPORTANT: While the Synopsys MilkyWay integration options are still visible, the integration has been deprecated.

You cannot enable this setting on Windows clients.

See Setting Up Link-In for details on link-in of large files.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Linking Large Files](#)

[Multithreading Optimization](#)

[Turning Off Keyword Expansion](#)

[Limit Transfer Timeout](#)

[Maintenance Retry Attempts](#)

[Maintenance Retry Interval](#)

Startup Options

You can have DesignSync run a startup script when you invoke any DesignSync client. Use the Startup options pane to specify a startup script.



Run this startup script:

dsinit.dss

The script can contain DesignSync and Tcl commands. By default, the script is named `dsinit.dss`, but you can choose any name and path for the script. DesignSync interprets the specified script file as a dss script unless the file has a `.tcl` extension, in which case DesignSync interprets it as a stcl script. If you do not specify a path for the script, DesignSync searches for the script in these directories in the following order:

1. `$SYNC_USER_CFGDIR` (resolves to `<HOME>/synchronicity` by default on Unix and `%AppData%\Synchronicity` on Windows)
2. `$SYNC_SITE_CUSTOM` (resolves to `<SYNC_CUSTOM_DIR>/site` by default)
3. `$SYNC_ENT_CUSTOM` (resolves to `<SYNC_CUSTOM_DIR>/enterprise` by default)

DesignSync searches for the script in the `$SYNC_SITE_CUSTOM` directory only if the `$SYNC_USER_CFGDIR` directory does not contain the script.

Examples of tasks you might include in a startup script are:

- `scd` to a particular directory
- Perform an incremental populate (so you always have the latest versions of files)

Your system administrator or project leader can also prepare startup scripts for DesignSync to source at startup. In this way, your site or project team can share the same aliases. See Autoloading Tcl Procedures for more information.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

See the following topics in DesignSync Data Manager User's Guide for more information:

DesignSync Data Manager User's Guide: DesignSync Command-Line Shells

DesignSync Data Manager User's Guide: Running Scripts

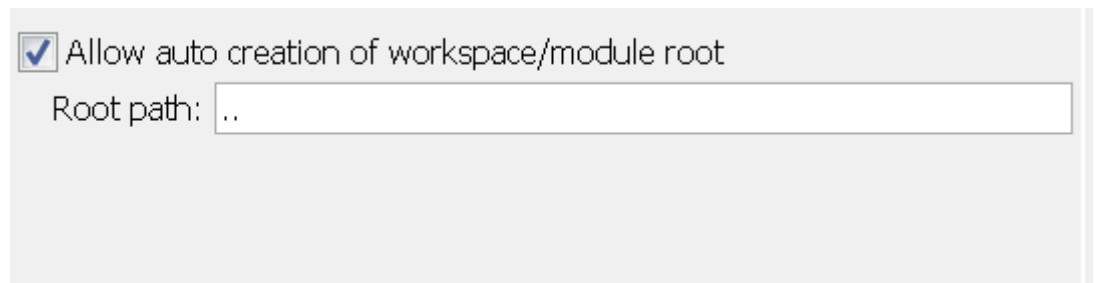
DesignSync Data Manager User's Guide: Creating DesignSync Scripts

Autoloading Tcl Procedures

Using Environment Variables

Workspaces

Using the **Workspaces** options pane of SyncAdmin, a system administrator can optimize the behavior of DesignSync workspaces.



The screenshot shows a configuration pane with a checked checkbox labeled "Allow auto creation of workspace/module root". Below the checkbox is a text input field labeled "Root path:" containing the value "..".

Allow auto creation of workspace/module root

When checked, enables auto-creation of workspace root at specified location. The default value is ".." which is one level above the base directory of the module or project (file-based) DesignSync objects. The workspace root is used to establish context in the workspace when data is populated into a workspace (by mkmod for newly created modules, populate for modules or files-based data from the server, and co, and setvault for file-based data.). This context allows for "smart" processing of new and existing data. Among other benefits, the metadata allows you to track if the information in the workspace has been moved outside of DesignSync.

When not checked, the workspace root is not created when a module or vault is associated with the workspace.

Note for modules workspaces:

If there is no existing root directory in the specified directory for the new module or above that directory, the creation operation fails.

For example: If you want to create a module in c:\workspaces\engineering\chip\, any of these directories can be defined as your module root. If auto-creation is allowed, and there is no defined module root, the module root will be created, by default, at the "engineering" level. If auto-creation is not allowed, a module root directory must exist at one of the following directories: workspaces, engineering, or chip in order to create the new module in the workspace.

Site Options

Site Options

The **Site Options** pane in the **Synchronicity Administrator** window allows you to set a variety of site-wide options.

When you have made your selections and you click **OK** or **Apply**, settings are written to the appropriate registry files and inherited by all users on your LAN.

Note: You may have to reset your server for the options to be activated.

Click on the image for more information about each field.

Server Settings

Alarm trigger maintenance timeout: Minutes

Limit server process size

Maximum process size: Mbytes

Enable disk quota checking

Enable Access Control Administrator

Access control deny logging:

Access control deny log rotation size: Mbytes

Honor 'Aliases' keyword expansion

Allow auto-merging when creating a new module version on the server

Allow older clients to access non-cacheable objects

Temporary upload server workspace

Browse...

Mirror Write-through

Prohibit mirror write-through

Allow mirror write-through

Alarm Trigger Maintenance Timeout

Alarm triggers can be used to perform background maintenance tasks such as, automatic Enterprise Design Object synchronization and automatic email notifications when fields on defect notes have been inactive for more than a specified time.

SyncAdmin lets you change the alarm trigger frequency. Enter the number of minutes, hours or days in the field. Click to choose Minutes, Hours, or Days. The default frequency is every 15 minutes.

For information on configuring Enterprise Design Servers, see Enterprise Servers. For more information on Enterprise Design Synchronization, see Enterprise Design Synchronization Queue.

For more information on using the alarm trigger to monitor defect notes, see *ProjectSync User's Guide: Setting Up Email Reminders for Defects*.

This option can also be set using the Registry Key: Alarm Trigger Frequency (MaintenanceTimeout).

Limit Server Process Size

You can choose to limit the server process size by clicking the check box and selecting the maximum size in MB.

Enable disk quota checking

Enable disk quota checking is disabled by default. When enabled, DesignSync provides advanced warning when reaching the soft quota limit, allowing you to adjust the quota or data storage location appropriately.

Enable Access Administrator

Access Administrator is disabled by default. When enabled, Access Administrator menus appear in the DesignSync web interface, allowing you define access controls using a graphical interface.

Note: Before you activate the Access Administrator, you should create a user profile with the name username as the server owner (usually syncmgr). After Access Administrator has been enabled and the server reset to activate the changes on the server, log in to DesignSync through the web-interface using the same login as the server owner.

Access Control deny logging

Access Control deny logging is disabled by default. When enabled, DesignSync logs information about denied command access. You can select from the following options:

- **Disable** - Do not enable "deny" logging for Access Control.
- **Enable** - Log the denied action, the user performing the action, and any parameters specified with the action.
- **Enable verbose** - In addition to information captured by the Enable setting, log the reason the access was denied.

Access control deny log rotation size

When the Access Control system denies access to a command and the Access Control deny logging is enabled, you can specify the maximum size of the log. After the maximum file size is reached, the file is automatically cleared when the next access control request is denied. By default the maximum log size is 1 MB.

Honor 'Aliases' Keyword Expansion

The **Honor 'Aliases' Keyword Expansion** is enabled by default. It allows the expansion of the \$Aliases: V6R2019x-Final,V6R2020xFD01 \$ keyword, which expands to the tag names. The expansion of the \$Aliases: V6R2019x-Final,V6R2020xFD01 \$ keyword can be individually disabled to improve performance if you are not using the

Aliases keyword. This expansion is disabled on the server, and impacts the server vaults.

Note: This is a separate setting than the client-side Performance option, **Enable Keyword Expansion**.

Allow auto-merging when creating a new module version on the server

Auto-merging is allowed by default. To ensure that users' checkins are always from the Latest version of a module, deselect this option. This functionality applies to any operation that creates a new module version.

This option applies to servers using the DesignSync installation from which SyncAdmin is run. Auto-merging is allowed or disallowed for all module data on a server.

Allow older clients to access non-cacheable Objects

Pre-V6R2016x clients may attempt to cache objects even if they are intended to be excluded from caching. By default, if the server receives a request from an older client, it will comply with the request, even if that permits overriding the non-cacheable directive. To protect the information from unauthorized caching by older clients, click the checkbox to enable this option on the server. A user who attempts to populate objects into a cache from an older client will receive an appropriate error message.

This option can also be controlled through the Allow Non-Secure Cache Populate (AllowNonSecureCachePopulate) registry key.

Temporary upload server workspace

A location on the server which is used as the temporary directory when a compressed IP is uploaded to DesignSync. The server directory must be accessible by the users performing the upload operation and contain enough space to hold the uncompressed file while the server calculates the changes in the compressed file. For more information on uploading compressed IP, see Upload Archive.

This option can also be controlled through the Temporary Directory for Upload registry key.

Mirror Write-through

DesignSync performs a mirror write-through for any fetch state (not just `-mirror`) when a mirror is set on the local workspace using the `setmirror` command. As System Administrator, you can change the behavior of the mirror write-through by:

- Prohibiting mirror write-through
- Allowing mirror write-through

For more information about mirrors, see [Mirroring Overview](#).

OK

When you click on the **OK** button, your changes are written to the registry file you selected when you started SyncAdmin.

After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, your changes are written to the registry file you selected when you started SyncAdmin.

After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin online help is displayed.

Related Topics

[DesignSync Data Manager User's Guide: Auto-Merging](#)

Client Triggers

Client Triggers Options

Triggers are watchpoints that cause an action to occur automatically in response to some other action, such as checking in a file. DesignSync triggers can be defined to execute a Tcl script whenever a revision-control operation occurs. For example, you could create a trigger that compresses files when they are checked in and uncompresses them again when the files are checked out. Each revision-control operation generates a series of events and each event has its own set of associated event properties. (See [Events Overview](#) and [Event Properties](#) for more details.)

The **Client Triggers** options pane in the **Synchronicity Administrator** window allows you, as a LAN administrator, to create client-side triggers. Once you add a trigger, it is listed at the top of the **Synchronicity Administrator** window along with a definition of

its action and properties. This list is updated as trigger information is added and/or deleted using the SyncAdmin tool.

- To add a trigger, click **Add Trigger**.
- To edit an existing trigger, highlight the trigger in the window and click **Edit Trigger**.
- To delete a trigger, highlight the trigger in the window and click **Delete**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (`$HOME/.synchronicity/UserRegistry.reg`) on UNIX or the Windows registry. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (`$HOME/.synchronicity/UserRegistry.reg`) on UNIX. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click the **Cancel** button, SyncAdmin exits. If you have not clicked the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

The Property List Window

Event Overview

Events Properties

Supported Trigger Commands

Add or Edit a Trigger

To create a trigger, after clicking **Add Trigger**, fill in the fields in the Add Trigger pane. To edit an existing trigger after clicking **Edit Trigger**, edit the fields on the Edit Trigger pane.

Click on the image for more information about each field.

Trigger Name

In this text box, you enter a unique name for your trigger. Typically you would name the trigger to indicate the type of action that the trigger will take. This field cannot be left blank if you want to add a trigger.

Action Type

In this section, you define what will get executed when your trigger fires. The following choices are available:

- **Tcl File** This option indicates that each time the trigger fires, a defined Tcl file is loaded and executed. Use this option when you want to edit the file and have your changes take effect as soon as the trigger fires again.
- **Tcl Command** With this option you can enter single Tcl commands or a Tcl script directly in the **Synchronicity Administrator** window that will get executed when your trigger fires.
- **System Command** With this option, you can enter a system command line directly in the **Synchronicity Administrator** window that will get executed when your trigger fires. For example, you may want to enter specific UNIX or

DesignSync commands, or a particular program that you want to get executed each time your trigger fires.

For more information on, see [Supported Trigger Commands](#).

- **Tcl Stored Procedure** With this option selected, the defined procedure is loaded from the named file as soon as you hit the **Apply** button. Then, when the trigger fires, the stored procedure is executed. If you make any edits to the file that your stored procedure was loaded from, you must redefine the trigger in order to update the stored procedure. This option is the same as the **Tcl Commands** option except that the Tcl code comes from a file instead of being typed into the window.

Enabled

This check box is a toggle switch to enable or disable the trigger that you are defining. You can also enable or disable a predefined trigger by selecting a trigger from the displayed list and clicking in this box.

Tcl File Name/ Tcl Commands/ System Command/ Tcl File Name

The name of this field changes based on the **Action Type** that you selected. The following parameters apply:

- When **Tcl File** is the selected action, you can enter the name of the file that you want to be executed. You can also use the **Browse** button to browse your local file structure.

Note: The trigger searches only the current folder for the specified Tcl file. If you want to use a script in another folder, you must give the full file system path. Triggers do not recognize environment variables in your path specifications. For example, `/home/tmmf/scripts/reportCheckouts.tcl`.

- When **Tcl Command** is the selected action, you can type the Tcl scripting language in the text box. For example:

```
{  
  
set fd [open [glob ~/checkin.log] a];  
  
puts $fd "$objURL\n";  
  
close $fd  
  
}
```

Note:

You can include any valid Tcl commands, including autoloaded stcl procedures in the **Tcl Commands** field. See the **Autoloading stcl Procedures** topic in the System Administration folder of DesignSync Help for more information.

When **System Command** is the selected action, you can type in the system command that you want to execute. For example, in the case of a Unix system, you can enter `cd` followed by an `ls` command.

When **Tcl Stored Procedure** is the selected action, you can enter the name of the script that you want to be stored. You can also use the **Browse** button to browse your local file structure.

To use event property names in the Tcl and System commands, type the event property name preceded by a dollar sign (\$). See the Tcl Commands example. The dollar sign is the syntax for a Tcl variable. Before DesignSync executes the command, it replaces the variable with the value of the event property.

Trigger will fire when

For each revision-control operation, such as checking in, you can set triggers that fire only under specified conditions. For example, you can create a trigger that checks whether a user is permitted to check in certain types of files and aborts the check-in of a file if the user does not have permission.

This field allows you to define under what conditions you want the trigger to fire. When you click on the **Modify** button, it automatically displays the Property List window. This window allows you to select the conditions under which you want the trigger to fire.

Trigger will not fire when

For each revision-control operation, such as checking in, you can set triggers that do not fire under specified conditions. For example, you may not want a trigger to fire when the DesignSync command is tag or when certain users checks out a file.

This field allows you to define under what conditions you do not want the trigger to fire. When you click on the **Modify** button, it automatically displays the Property List window. This window allows you to select the conditions under which you do not want the trigger to fire.

Note: For either the **Trigger will fire when** or the **Trigger will not fire when** fields, SyncAdmin does not allow you to specify an event property with an empty string (has an expression of {}, "", or ' ') as its Expression List. For example, if you define a trigger to fire when the comment property type has {} as its Expression List, SyncAdmin

creates and saves the trigger, but the trigger definition specifies Always for **Trigger will fire when**.

To create a trigger that fires based on the presence or absence of a comment, use the **isComment** event property. See Client Trigger Examples for an example of this type of trigger.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. If you were not asked to select a registry file, your changes are written to your user registry file (`$HOME/.synchronicity/UserRegistry.reg`) on UNIX or Windows. After saving any changes, SyncAdmin exits.

Cancel

When you click the **Cancel** button, SyncAdmin exits. If you have not clicked the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Trigger Property List

By default, each trigger fires in response to every event. The **Trigger Property List** window provides a way for you to specify event properties that determine when your trigger will fire or will not fire.

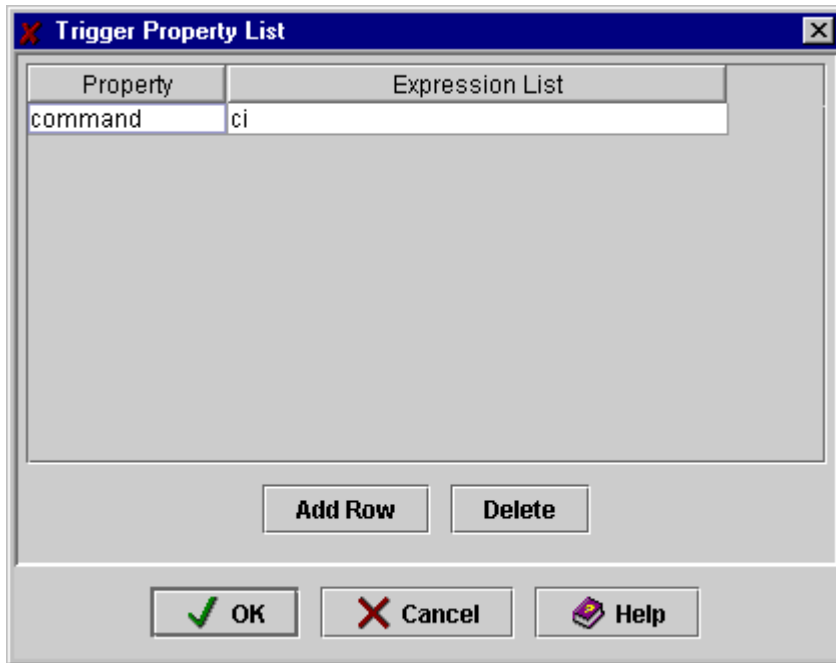
The **Trigger Property List** window works in conjunction with two fields on the **Add/Edit Triggers** dialog: the Trigger will fire when: field and the Trigger will not fire when: field. When you click **Modify** for either field, SyncAdmin displays the Property List window.

To limit the set of events that cause a trigger to fire, you can specify **Trigger will fire when** (on the Client Triggers tab) and then use the **Trigger Property List** window to specify value expressions for event properties. When an event occurs, DesignSync compares the event's properties with the expressions you specified in the **Trigger Property List** window. If they match, DesignSync fires the trigger. You can use the **Trigger Property List** window in the same manner with the **Trigger will not fire when** field to specify event properties that keep the trigger from firing.

- To create a new trigger property, click **Add Row** and edit the Property and Expression List fields.
- To edit an existing property, highlight the row and change the Property and Expression List fields.

- To delete a property, highlight the row and click **Delete**.

Click on the image for more information about each field.



Property

This field allows you to select from a pull-down menu of available event properties. Click on the arrow to display the menu. (See Event Properties for a list of event properties and their descriptions.)

Expression List

In this field, type the expression for DesignSync to compare to the value of the event property.

Guidelines:

- Some properties can take more than one expression. Separate multiple expressions with spaces. For example, if you select **command** from the Property pull-down menu, in the **Expression List** field you can enter: **ci co**.
- If you selected **type** from the Property pull-down menu, make sure that the event type name in the **Expression List** field has correct spelling and case. (See the Event Types list.) For example, if you type **preobject** in the Expression List field instead of **preObject**, no match is made to the type property and therefore the trigger does not fire.

- You can use wildcards in expressions. For example if you want your trigger to fire only for events of type **toVaultFilter** and **fromVaultFilter**, you would select **type** from the **Property** field. Then in the **Expression List** field, you can enter the expression ***Filter**.

OK

When you click on the **OK** button, the values entered are written to the **Client Triggers** window and the **Property List** window is closed.

Cancel

When you click on the **Cancel** button, the **Property List** window is closed without saving any changes.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

Client Triggers Options

Events Overview

Event Properties

Client Trigger Examples

Trigger to disable user george from checking in file top.v

This trigger fires when a user with username george performs a checkin of the file named `top.v`. The trigger executes Tcl commands that return an error, which aborts the checkin and displays an error message.

1. In the **Client Trigger** pane, click **Add Trigger**. The **Add Trigger** screen displays.
2. In the **Trigger Name** field, type a name, for example: **disableGeorgeTop**.
3. Make sure **Enabled** checkbox is checked.
4. For **Action Type**, choose **Tcl Command**.
5. In the **Tcl Commands** field, type:

```
puts "You have been denied this action by a trigger.\n"
error ""
```

6. For **Trigger will fire when**, click **Modify** to display the Property List window.

7. Click **Add Row**.
8. In the **Property** field, select **objPath**. In the **Expression List** field, type: `"*/top.v"`.
9. Click **Add Row**.
10. In the **Property** field, select **command**. In the **Expression List** field, type `ci`.
11. Click **Add Row**.
12. In the **Property** field, select **user**. In the **Expression List** field, type `george`.
13. Click **OK**.
14. At the **Add Trigger** screen, click **OK**.
15. At the **Client Trigger** pane, click **Apply**.

Trigger to validate files (with lint) in a directory before checkin

This trigger uses the preFolder event to get a list of files in the directory before performing a check-in operation. (The preFolder event has the objList property, which contains the filenames of all the files in the directory.) The trigger causes the system command lint to act on the files provided in the objList environment variable.

1. In the **Client Trigger** pane, click **Add Trigger**. The **Add Trigger** screen displays.
2. In the **Trigger Name** field, type a name, for example: `validationcheck`.
3. Make sure **Enabled** checkbox is checked.
4. For **Action Type**, choose **System Command**.
5. In the **System Command** field, type:

```
lint $objList
```

6. For **Trigger will fire when**, click **Modify** to display the Trigger Property List window.
7. Click **Add Row**.
8. In the **Property** field, select **type**. In the **Expression List** field, type `preFolder`.
9. Click **Add Row**.
10. In the **Property** field, select **command**. In the **Expression List** field, type `ci`.
11. Click **OK**.
12. At the **Add Trigger** screen, click **OK**.
13. At the **Client Triggers** pane, click **Apply**.

Trigger to allow only locked checkouts

This trigger uses the preCommand event type to make sure users check out files with a lock. The trigger fires when a user performs a check-out operation and has not specified

the lock option. The trigger executes Tcl commands that return an error, which aborts the checkout and displays an error message.

1. In the **Client Trigger** pane, click **Add Trigger**. The **Add Trigger** screen displays.
2. In the **Name** field, type a name, for example: **AllowOnlyCoLock**.
3. Make sure **Enabled** checkbox is checked.
4. For **Action Type**, choose **Tcl Commands**.
5. In the **Tcl Commands** field, type:

```
puts "You are only allowed to check out objects for  
write using the Lock switch.\n"  
error ""
```

6. For **Trigger will fire when**, click **Modify** to display the **Trigger Property List** window.
7. Click **Add Row**.
8. In the **Property** field, select **type**. In the **Expression List** field, type **preCommand**.
9. Click **Add Row**.
10. In the **Property** field, select **command**. In the **Expression List** field, type **co**.
11. Click **Add Row**.
12. In the **Property** field, select **isLock**. In the **Expression List** field, type: **0**.
13. Click **Add Row**.
14. In the **Property** field, select **isGet**. In the **Expression List** field, type **1**.
15. Click **OK**.
16. At the **Add Trigger** screen, click **OK**.
17. At the **Client Triggers** pane, click **Apply**.

Trigger to disable all commands

This trigger fires when a user enters a revision control command. The trigger executes Tcl commands that return an error, which aborts the command and displays an error message.

1. In the **Client Trigger** pane, click **Add Trigger**. The **Add Trigger** screen displays.
2. In the **Name** field, type a name, for example: **disableAll**.
3. Make sure **Enabled** checkbox is checked.
4. For **Action Type**, choose **Tcl Commands**.
5. In the **Tcl Commands** field, type:

```
puts "Sorry\n"  
error ""
```

6. For **Trigger will fire when**, click **Modify** to display the **Trigger Property List** window.
7. Click **Add Row**.
8. In the **Property** field, select **type**. In the **Expression List** field, type **preCommand**.
9. Click **OK**.
10. At the **Add Trigger** screen, click **OK**.
11. At the **Client Triggers** pane, click **Apply**.

Trigger to allow only checkins that have a comment

This trigger uses the **preCommand** event type to make sure users provide a comment when they check in files. The trigger fires when a user performs a check-in operation and has not specified a comment (when the **isComment** property has a value of 0). The trigger executes Tcl commands that return an error, which aborts the checkin and displays an error message.

1. In the **Client Trigger** pane, click **Add Trigger**. The **Add Trigger** screen displays.
2. In the **Name** field, type a name, for example: **requireCiComment**.
3. Make sure **Enabled** checkbox is checked.
4. For **Action Type**, choose **Tcl Commands**.
5. In the **Tcl Commands** field, type:

```
puts "You are allowed to check in objects only when you
have added a comment.\n"
error ""
```

6. For **Trigger will fire when**, click **Modify** to display the **Trigger Property List** window.
7. Click **Add Row**.
8. In the **Property** field, select **type**. In the **Expression List** field, type **preCommand**.
9. Click **Add Row**.
10. In the **Property** field, select **command**. In the **Expression List** field, type **ci**.
11. Click **Add Row**.
12. In the **Property** field, select **isComment**. In the **Expression List** field, type: **0**.
13. Click **OK**.
14. At the **Add Trigger** screen, click **OK**.
15. At the **Client Triggers** pane, click **Apply**.

Related Topics

The Client Triggers Tab

Property List Window

Events Overview

Event Properties

Events Overview

Each revision-control operation, such as a check-in, generates a series of **events**. You can use an event to activate a trigger, which in turn causes some other action to take place. In creating trigger scripts, you can use predefined events that DesignSync provides or you can create your own events to control the actions of your triggers.

Predefined Event Types

Several different types of events are generated by revision control operations. The specific events associated with an operation depend on the operation. In general, each command has a **preCommand** event before it begins and a **postCommand** event after it completes. In between, most commands have a **preObject** event before the operation on an object and a **postObject** event after the operation on an object. In addition, special filter events (**toVaultFilter** and **fromVaultFilter**) let you alter the contents of files that are moved into or out of a revision control vault.

Types of Events Generated by Each Revision Control Operation:

	pre Command	pre Folder	pre Object	to VaultFilter	from VaultFilter	post Object	post Command
Cancel	X		X			X	X
Check in	X	X	X	X	X	X	X
Check out	X		X		X	X	X
Delete File	X		X			X	X
Delete Folder	X		X			X	X
Delete Vault	X		X			X	X
Delete Version	X		X			X	X
Populate	X		X		X	X	X
Retire	X		X			X	X
Tag	X	X	X			X	X
Unlock	X		X			X	X

Event Type Descriptions

This Event Type	Takes Place...
-----------------	----------------

preCommand	<p>After the command's arguments have been parsed, but before any objects are operated on.</p> <p>If a preCommand trigger returns an error, the entire command is aborted.</p>
preObject	<p>Before the operation on each object.</p> <p>If a preObject trigger returns an error, the operation on the object is aborted.</p>
preFolder	<p>Before DesignSync operates on a directory.</p> <p>If a preFolder trigger returns an error, the operation on the directory is aborted.</p>
toVaultFilter	<p>Before an object is checked into the vault.</p> <p>If a toVaultFilter trigger returns an error, the check in of the object is aborted.</p> <p>A trigger for a toVaultFilter event reads the contents of the file named by the fromFile property and creates its output in a file named by the toFile property.</p>
fromVaultFilter	<p>Before an object is fetched to the local directory from the vault. This event can occur either during check out or during check in with -lock, -keep, and so on.</p> <p>If a fromVaultFilter trigger returns an error, the fetch of the object is aborted.</p> <p>A trigger for a fromVaultFilter event reads the contents of the file named by the fromFile property and creates its output in a file named by the toFile property.</p>
postObject	<p>After the operation on each object.</p>
postCommand	<p>After the command's entire operation is completed.</p>

Each event is described by a number of **properties** that vary according to the revision-control operation. The properties associated with an event depend on the type of event. For example, the **toVaultFilter** event includes the **fromFile** property, whose value is the name of the file where the trigger reads its input. For a list of all event properties, see Event Properties.

About Filter Events

For most event types, you cannot affect how the revision-control operation is performed. However, the **toVaultFilter** and **fromVaultFilter** events let you specify actions to be performed on objects going to or from the vault.

The **toVaultFilter** event occurs after the **preObject** event. A **toVaultFilter** trigger performs a two-step operation. First the trigger reads data from the object you want to check in, the **fromFile**. Then the trigger writes data to a temporary file, the **toFile**. The contents of the **toFile** are then checked into the vault. This two-step process lets you perform filtering before the object is sent to the vault.

Similarly, the **fromVaultFilter** event occurs when an object is fetched from the vault. A **fromVaultFilter** trigger also performs a two-step operation. First the trigger reads data from the object you want to check out, the **fromFile**. Then the trigger writes data to a temporary file, the **toFile**. The contents of the **toFile** are then checked into your working directory. This two-step process lets you perform filtering before the object is sent to the working directory.

When you have more than one **toVaultFilter** or **fromVaultFilter**, each subsequent filter reads its input from its predecessor. However, the filters are not executed in any determined order. If you want your filters to run in a particular sequence, define a single trigger that performs the filter operations in the required order.

Both **toFile** and **fromFile** are event properties. Because you may want to specify more than one filter trigger, make sure your triggers read data from the file named by the **fromFile** property. You should not read data from the **objPath** property because the file it names may not yet exist.

You also can create your own custom events to use in Tcl scripts. For more information, see creating events.

Event Types and the Client Triggers for Tag Operations

Client triggers for tag operations activate trigger events based on the number of objects specified for tagging.

When tagging a large number of objects, DesignSync groups all **preObject** events and activates them, then groups all **postObject** events and activates them. For example:

1. **preCommand**
2. All **preObject** events:
 - **preObject** (object 1)
 - **preObject** (object 2)
 - **preObject** (object 3)
 - **preObject** (object4)
3. All **postObject** events:
 - **postObject**(object 1)
 - **postObject** (object 2)
 - **postObject** (object 3)
 - **postObject** (object 4)
4. **postCommand**

When tagging a small number of objects, DesignSync activates preObject and postObject events for each object in succession. For example:

1. preCommand
2. preObject (object 1)
3. postObject (object 1)
4. preObject (object 2)
5. postObject (object 2)
6. postCommand

Related Topics

Property List Window

Event Properties

Event Properties

The following table describes all the predefined event properties. The properties associated with an event depend on the type of event and the type of operation that fires the trigger. The event properties listed in the table are available from the **Property** pull-down menu in the **Property List** window.

For the list of commands that supports triggers, see Trigger Commands.

Note: The Boolean properties listed in this table (**isKeep**, **isShare**, and so on) do not necessarily correspond directly with the flags (**-keep**, **-share**) given on the command line. Instead, these Boolean values indicate how the command operates. For example, if you check in a file using the **-lock** option, both the **isLock** and **isKeep** property values will be 1 (true) because checking in with a lock also keeps a copy of the file in the local directory.

Event Property	Description
clientIP	The IP address of the client. (This variable is set on the server, not the client.)
command	The name of the DesignSync command that generated the event. The value can be any of the supported trigger commands. For example: <pre>trigger create Email -require command "ci co"</pre> <p>Note: Populate commands set this value to co.</p>
comment	The check-in or check-out comment. For example: <pre>comment Ready for release</pre>

	To determine if a comment was specified for a check-in or check-out operation, use the isComment event property.
errorCount	The number of objects for which the operation failed. For example, if you are checking in a number of files, errorCount reports the number of files that failed to check in. For example: <code>errorCount 3</code>
fetchState	The state of the object in the work area. Possible values are: Lock, Copy, Mirror, Cache, Reference, or NotFetched For example, an unlocked copy would return: <code>fetchState Copy</code>
fromFile	The local file system path of the file from which the trigger reads its input. For example: <code>fromFile /home/tmmf/Projects/Sportster/code/samp.asm</code>
groupID	The UNIX groupID of the process. For example: <code>groupID 100</code>
hcmTarget	The target URL of the module configuration on which the system operates.
host	The name of the machine where the trigger executes. For example: <code>host zeus</code>
isBranch	Indicates whether the branch option is in effect (for example, in checking in a file, you specify the -branch option plus a branch name argument). If yes, the value is 1; if no, the value is 0. For example: <code>isBranch 1</code> Notes: <ul style="list-style-type: none"> • The isBranch property is set only if the -branch option is used on the command line. • If the -branch option is specified on the command line, its argument is the value for the selector property for preCommand and postCommand events.
isComment	Indicates whether there is a check-in or check-out comment specified. If there is a comment, the value is 1; if no comment, the value is 0. For example: <code>isComment 1</code>
isDelete	Indicates whether the delete option is in effect (for example, deleting

	<p>a tag from the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isDelete 0</pre>
isForce	<p>Indicates whether the force option is in effect (for example forcing a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isForce 0</pre>
isGet	<p>Indicates whether the get option is in effect (for example, checking out an unlocked file). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isGet 1</pre>
isKeep	<p>Indicates whether the keep option is in effect (for example, keeping a local copy after a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isKeep 1</pre> <p>Note: Checking in an object with the -lock option causes both the isLock and isKeep properties to have a value of 1, even though you did not specify the -keep option with the ci command. The isKeep property has this value because in DesignSync, the ci -lock command implies that a local copy of the file should be kept in the work area (-keep).</p>
isLock	<p>Indicates whether the lock option is in effect (for example, keeping a locked copy after a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isLock 1</pre> <p>Note: Checking in an object with the -lock option causes both the isLock and isKeep properties to have a value of 1, even though you did not specify the -keep option with the ci command. The isKeep property has this value because in DesignSync, the ci -lock command implies that a local copy of the file should be kept in the work area (-keep).</p>
isMerge	<p>Indicates whether the merge option is in effect (for example, fetching a file from the vault and merging it with a locally modified copy of the same file). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isMerge 0</pre>
isMirror	<p>Indicates whether the mirror option is in effect (for example, keeping a link to the file in the mirror directory after check-in). If yes, the value is 1; if no, the value is 0. For example:</p>

	<code>isMirror 0</code>
isNew	<p>Indicates whether the new option is in effect (for example, checking in a new file to the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isNew 0</code></p>
isOverlay	<p>Indicates whether the overlay option is in effect (for example, in checking out a file, you specify the -overlay option plus a selector argument). If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isOverlay 1</code></p> <p>Notes:</p> <ul style="list-style-type: none"> • The isOverlay property is set only if the -overlay option is used on the command line. • If the -overlay option is specified on the command line, its argument is the value for the selector property for preCommand and postCommand events.
isRecursive	<p>Indicates whether the recursive option is in effect (for example, checking in a folder and all its subfolders). If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isRecursive 0</code></p> <p>Because all commands issued from the GUI are recursive, this value will always be 1 when you are using the GUI.</p>
isReference	<p>Indicates whether reference option is in effect (for example, checking out a file with reference option enabled to avoid copying over the entire file to the local workspace). If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isReference 1</code></p>
isReplace	<p>Indicates whether the tag is being replaced. If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isReplace 0</code></p>
isRetain	<p>Indicates whether the retain option is in effect (for example, retaining the last-modified timestamp of the file rather than the check-in time). If yes, the value is 1; if no, the value is 0. For example:</p> <p><code>isRetain 0</code></p>
isServer	<p>Indicates whether the trigger is fired by the server or by a client. The value is 1 when fired by the server and 0 when fired by a client. For</p>

	<p>example:</p> <pre>isServer 0</pre>
isShare	<p>Indicates whether the share option is in effect (for example, keeping a link to the file in the shared cache directory after check-in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isShare 0</pre>
isSkip	<p>Indicates whether the skip option is in effect (for example, checking in a version that is not derived from the latest version in the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isSkip 0</pre>
isVersion	<p>Indicates whether a version was specified for the revision control operation (with the <code>-version</code> option). If a version was specified, the value is 1; if no version was specified, the value is 0. For example, if you enter:</p> <pre>dss> co -version 1.1 sample9</pre> <p>Then <code>preCommand</code> and <code>postCommand</code> events show:</p> <pre>isVersion 1</pre>
keys	<p>Indicates how revision-control keywords in a file are treated during revision-control operations. The values are:</p> <pre>kkv -- keep keyword values</pre> <pre>kk -- keep keywords</pre> <pre>kv -- keep values</pre> <pre>ko -- keep output</pre> <p>(For more information about revision control keywords, see DesignSync Data Manager User's Guide: Revision Control keywords.)</p>
objList	<p>A list of objects (files and subdirectories) in the directory on which the system operates. For example:</p> <pre>objList alu decoder stack_pointer top.f top.gv top.v</pre>
objPath	<p>The file system path of a file, directory, or link that is operated on by the system. For example:</p>

	<pre>objPath /home/tmmf/Projects/Sportster/code/samp.asm</pre>
objURL	<p>The URL of an object that is operated on by the system. For example:</p> <pre>objURL file:///home/tmmf/Projects/Sportster/code/samp.asm</pre> <p>For the preCommand event for the populate operation, this property is the directory on which the populate is invoked.</p>
okCount	<p>The number of objects for which the operation succeeded. For example, if you are checking in a number of files, okCount reports the number of files that were checked in successfully. For example:</p> <pre>okCount 15</pre>
osName	<p>The name of the operating system the trigger is executing on. On UNIX, this property has the same value as the uname -s command. On Windows, possible values are WIN32_WINDOWS and WIN32_NT. A UNIX example is:</p> <pre>osName SunOS</pre>
osVersion	<p>The version of the operating system the trigger is fired on. On UNIX, this property has the same value as the uname -r command. On Windows, this property gets the major and minor version numbers--for NT, for example, 4.0. A UNIX example is:</p> <pre>osVersion 5.5.1</pre>
parentPID	<p>The process ID of the process that launches the trigger. (This process may not be the same process that executes the trigger.) For example:</p> <pre>parentPID 113</pre>
port	<p>The port used by the server processing the trigger. (This variable is set on the server, not on the client.)</p>
requestURL	<p>The URL from the HTTP header of the request. (This variable is set on the server, not on the client.)</p>
selector	<p>The selector for the branch on which the object resides. For example:</p> <pre>selector Trunk</pre> <p>For all preObject events, this property is the selector that the revision control operation uses. For example, for check-out events, the property is the selector that the check-out operation uses to determine which version to check out. You manipulate this property</p>

	<p>by using the setselector command or specifying co -version.</p> <p>For postObject check-out events, this property is the version number of the version retrieved.</p> <p>For preCommand and postCommand events, this property is the value specified as an argument to the -branch, -overlay, or -version option for the revision control operation. The value of the selector property is empty if these options are not specified.</p> <p>Notes:</p> <ul style="list-style-type: none"> • If the check-out command did not specify a version, the default value of selector is Trunk. • If you use the check-out command with the -version option, the value of selector is the value you specified for the -version option.
statusOK	<p>Indicates whether the operation on an object was successful. For example, if you are checking in a file, statusOK reports whether the file was checked in successfully. A value of 1 indicates success; 0 indicates an error. For example:</p> <pre>statusOK 1</pre>
statusText	<p>Any text that explains the value of the statusOK property. If there is no status text, the value is an empty string. For example:</p> <pre>statusText Success - New version 1.2</pre>
tag	<p>The name of the tag. For example:</p> <pre>tag readyForRelease</pre>
toFile	<p>The local file system path of the file that a filter trigger writes data to. For example:</p> <pre>toFile /var/tmp/synBAAa002qq</pre>
trigArgs	<p>Any arguments that have been passed from the command line to the trigger. For example:</p> <pre>trigArgs Golden</pre> <p>To pass parameters from the command to your trigger, use the -trigarg option with the DesignSync command. For example, this DesignSync tag command gives the value golden to the trigArgs property:</p> <pre>dss> tag Golden *.v -trigarg "golden"</pre>

trigger	The name of the trigger being executed. For example: <code>trigger listFiles</code>
type	The type of event. For example: <code>type postObject</code>
user	The name of the user who created the event. For example: <code>user jackie</code>
vaultURL	The URL of the vault of a particular object. For example: <code>vaultURL sync://srvr1.ABCo.com:30090/Projects/ALU</code> For the preCommand trigger for populate, this property is the URL of the vault folder that corresponds to the directory to be populated. For the fromVaultFilter trigger, this property is the vault URL of the file on which the fromVaultFilter trigger is operating. For example: <code>vaultURL sync://srvr1.ABCo.com:30090/Projects/ALU/test.v;</code>
version	The version of the object being operated on. For example: <code>version Trunk</code> For preObject check-out events, this information is used to determine which version to check out. For postObject check-out events, this property is the version retrieved. If the check-out command did not specify what version to get, the value of version is Trunk .

Access to Event Properties from Scripts and the Command Line

Event properties are available to both Tcl and non-Tcl trigger scripts as environment variables.

To access event properties from...	Use...
Tcl commands and scripts	The event property name preceded by dollar sign (\$). (The dollar sign is the syntax for a Tcl variable.) For example: \$objPath.
Scripts in languages other than Tcl (such as Perl)	The event property name preceded by SYNC_ . For example, SYNC_trigArgs.

Displaying the Event Properties for a Revision Control Operation

You can list the event properties of an operation by running the **triggerInfo.tcl** script, located in `<SYNC_DIR>/share/examples/triggers`. To run the script, create a simple trigger that fires for the revision-control operation for which you want to know the associated events and event properties. For example, to list all the properties of the `preObject` event for the `co` command, create a trigger like the following one:

1. In SyncAdmin, choose the Client Triggers tab.
2. In the **Name** field, type a name, for example: `checkouttriggerInfo`.
3. For **Action Type**, choose **Tcl File**.
4. In the Tcl File Name field, type the path to the `triggerInfo.tcl` file, for example:

```
/home/jackie/syncinc/share/examples/triggers/triggerInfo
.tcl
```

5. For **Trigger will fire when**, click **Modify** to display the Property List window.
6. In the **Property** field, select **command**. In the **Expression List** field, type `co`. Click **Add/Edit** to add the property to the trigger definition.
7. In the **Property** field, select **type**. In the Expression List field, type `courier, courier; font-weight: bold;" face=courier>preObject`. Click **Add/Edit**; then click **OK**.

Then check out a file. You will get a list like the following in your command shell window:

```
# co preObject
# command    co
# comment
# fetchedState    Copy
# host    hera
# isForce    0
# isGet    0
# isLock    1
# isMerge    0
# isMirror    0
# isRetain    0
```

DesignSync Data Manager Administrator's Guide

```
# isServer    0
# isShare    0
# keys       ko
# objPath    c:\Projects\Sportster\top\top.v
# objURL     file:///c:/Projects/Sportster/top/top.v
# osName     WIN32_NT
# osVersion  4.0
# parentPID  253
# selector   Trunk:Latest
# trigger    triggerInfo
# type       preObject
# user       mmf
# version    Trunk:Latest
```

Related Topics

[Property List Window](#)

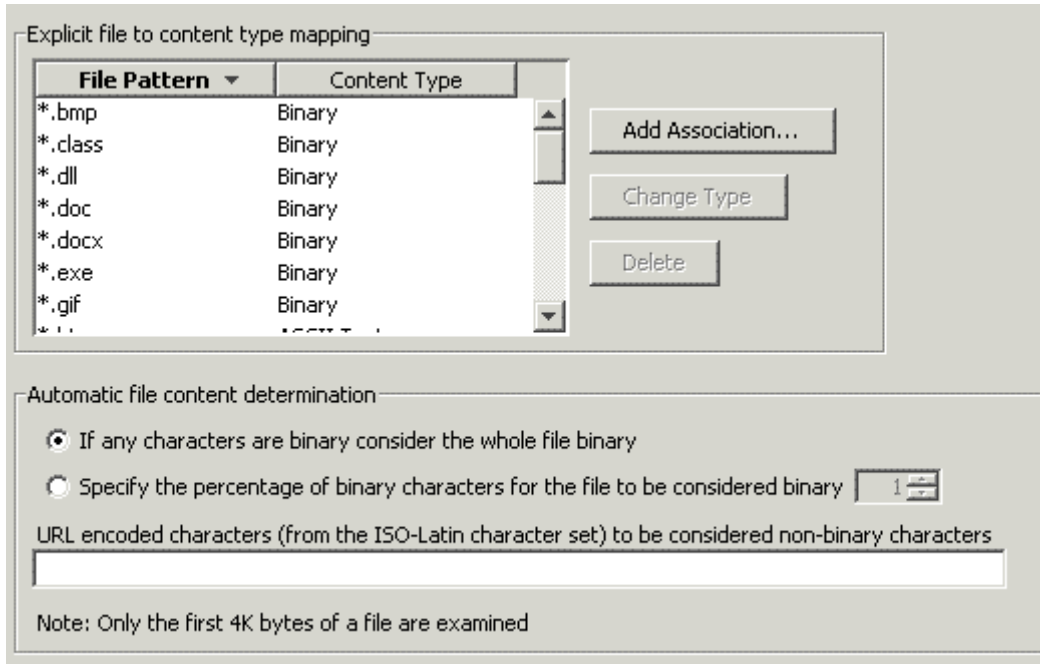
[Events Overview](#)

[Client Trigger Examples](#)

File Content

File Content

The File Content options determine whether objects are created as text or binary in DesignSync.



Explicit file to content type mapping

Stores a list of file extensions or patterns with explicit content type definitions. The definitions stored here are used instead of DesignSync's smart file type determination. This saves processing time for known binary file types or patterns.

- **File Pattern** - Shows a list of the defined file extensions. Click on the title bar to sort the list by file pattern in ascending or descending order.
- **Content Type** - Shows the defined content type for the file extension. Options are ASCII and Binary. When the type is ASCII, DesignSync allows version comparisons using the compare utilities. When the type is Binary, DesignSync does not allow version comparisons. Click on the title bar to sort the list by content type.

Add Association

To add a file content type, click **Add Association**.

Change Type

Changes the file content type. Select the association you want to change and click the Change Type button to toggle the association between ASCII and Binary.

Delete

Deletes the specified file content type. It does not prompt for confirmation.

Automatic File Content Determination

You can customize the smart determination for a file by specifying what percentage of non-ASCII characters are required to consider the file binary.

- **If any characters are binary, consider the whole file binary** - This option process the first 4K of the file and if it encounters any binary characters, considers the entire file binary, and sets that as the file type. (Default)
- **Specify the percentage of binary characters for the file to be considered binary** - This option processes the first 4K of the file an if the percentage of binary characters equals or is greater than the value set in this option, considers the entire file binary and sets that as the file type.

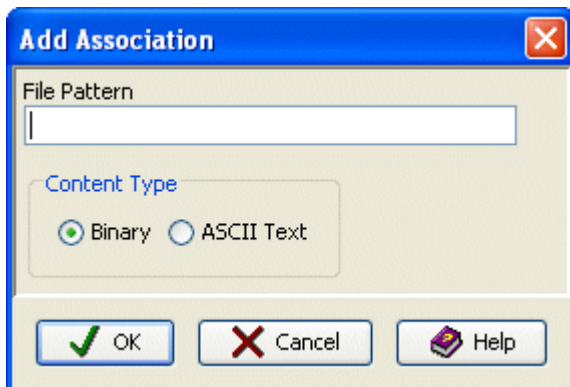
URL encoded characters to consider

Specify any binary characters that should be considered text for purposes of determining file type. Enter the character as a URL encoded string, for example, to consider the copyright character (©) an ASCII character, enter "%A9."

Add Association

The most common file associations are pre-defined in DesignSync, but you can add additional definitions as needed.

To add a File Association, on the **File Contents** options pane, click **Add Association** and fill in the fields in the **Add Association** pane.



File Pattern

Type the desired pattern to match. In most cases, you will want this to be a file extension, but it is not required. Use wildcards to indicate variable input. For example, to set all .chm files to binary, enter *.chm in the field.

Content Type

Select the appropriate type for the file pattern.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Links Options

The Link Options page allows you to manage how links are created and managed by the system.

You can use links in DesignSync in the following ways:

- Following symbolic links during checkin operations to check in a file or folder target of the link.
- Managing symbolic links in DesignSync as a version-controlled object.
- Sharing object copies, which can be done in the following ways:
 - a symbolic link to a module cache. For information about module caches, see Procedure for Setting Up a Module Cache.
 - a symbolic link to a file in a mirror
 - a symbolic link to a file in a file cache
 - a hard link to a file in a file cache

Symbolic links managed or followed by DesignSync

DesignSync, by default, handles symbolic links to files by dereferencing the link and operating on the object pointed to by the link. For example, DesignSync checks in a file pointed to by a link, rather than the link itself. Depending on your check-in mode, DesignSync leaves behind the file, a link to the cache or mirror, or a reference. When the file (from the dereferenced link) is checked back out, the file is retrieved. (This treatment of links is similar to `tar -h` on UNIX.)

DesignSync, by default, handles links to folders as though the link were the folder itself. The link appears in the Tree View and you can follow the link to the folder pointed to by the link. You can check in the contents of the folder by performing a recursive checkin

(the default behavior from the GUI or by specifying `-recursive` from the command line). If you try to check in a link to a folder without the recursive option, DesignSync tries to check in the folder itself, which fails because DesignSync doesn't revision-control folders.

DesignSync also allows you to manage symbolic links so the link itself is checked in; not the file or folder it is pointing to. The link itself becomes a versionable object. Upon checkout, the link is created in the workspace. Recursive operations do not follow the link.

Effective use of links relies on the proper use and management of links by the team sharing the vault. For example, if a symbolic link is defined using an absolute path, that path needs to be accessible by all team members to be meaningful. Links defined with relative paths are typically more useful.

Symbolic links for sharing data in a file cache or mirror

When the mirror or share fetch state is used, a symbolic link is created in the workspace linking to the file in the mirror or file cache. The file in the mirror or cache can be shared among many workspaces.

If a symbolic link is managed as a link within DesignSync, then even if fetching the symbolic link with the share option, the symbolic link is created in the workspace and the object is left as a copy, as if the **Unlocked Copy** fetch state (`-get` option) was used. DesignSync does not create symbolic links to the cache for managed symbolic links. If a collection object has some members as managed symbolic links and some members as files, then the object is left in the cache state and the managed symbolic links are created in the workspace and the file copies are created as symlinks to the cache. When all members of a collection are managed symbolic links, the object is left in the copy state.

The handling of managed symlinks when using the mirror fetch state is configurable.

You have the option to leave a link to the mirror or recreate the managed link in the workspace. See the Symbolic Link Handling for details on fetching with the mirror fetch state and managed symbolic links.

Hard Links for sharing data in file cache

When a symbolic link is created, it uses a system inode. When working with large numbers of files, workspaces, and caches, you may find that your system does not have enough inodes to support symbolic links to the cache. Hard links provide the same advantage as symbolic links, disk space savings, but use a single inode per file, rather than an inode for each link to a file; this can significantly reduce the amount of inodes required for caching, leaving those system resources free for other uses. Hard links may also use less disk space than symbolic links, depending on the full path length of files in the cache.

In order to use hard links on your system, you must verify the following:

- Both file cache and workspace are on the same disk file partition.
- Both file cache and workspace have the same mount point.
Note: When automount is used, for example, with home directories, the mount points may be different even though the workspace and cache are on the same file partition. You may need to consult your system administrator to find out how the directories are mounted.
- Hard links must be enabled using the SyncAdmin panel described below, or the registry key.

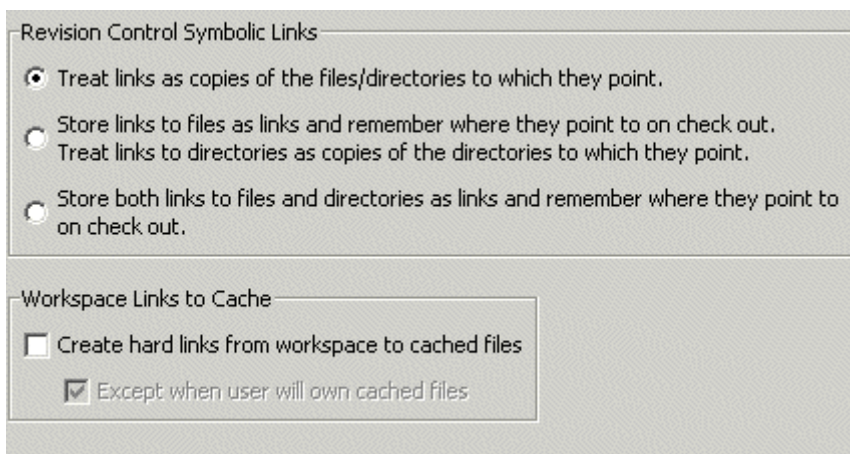
Note: With SUID enabled, the owner of the cache is always the designated system administration account, usually `syncmgr`, and all files in the cache are owned by this account. When SUID is not enabled, the owner of the cache files is the user who fetched the version into the cache. Because the cache hard links look like standard DesignSync file objects, the owner may be confused and attempt to modify the cache files inadvertently. To minimize confusion and make it obvious that the shared objects in the workspace are not meant to be modified, DesignSync provides an option to use symbolic links when the owner of the file in the cache is the same as the user running the share operation.

Defining link options in SyncAdmin

The **Links** options pane in the **Synchronicity Administrator** window allows you, as a LAN administrator, to specify how DesignSync should handle links in your LAN.

Making changes in the **Links** tab will force the consistent treatment of symbolic and hard links across the project team. When selections are made, settings are written to the appropriate registry files and inherited by all users on your LAN.

Click on the image for more information about each field.



Treat links as copies of the file/directory to which they point.

With this option selected, if a link to a file or directory is checked in, the link is dereferenced and the file(s) that the link points to is operated on. When the link is checked back out, the file(s) that the link points to will be displayed. This option represents the default DesignSync behavior with regards to links.

Store links to files as links and remember where they point to on check out. Treat links to directories as copies of the directories to which they point.

With this option selected, if you check in or check out a link to a file, the link itself gets managed and the file pointed to by the link is not affected. When you check in a link to a directory, the link is dereferenced and the files in the directory that the link points to are checked in.

Store both links to files and directories as links and remember where they point to on check out

With this option selected, the first time you check in a link to a file or directory, the link itself is placed under revision control. The link remains under revision control and the file(s) that the link is pointed to are not affected.

Create hard links from workspace to cached files

With this option selected, DesignSync will create hard links to file caches whenever appropriate.

Except when user will own cache

This option becomes selectable when the **Create hard links from workspace to cached files** option is selected. When this option is selected, a user who owns the files in the cache will not be able to create hard links to the file in the cache; the link to the file will be created as a symbolic link. This is a recommended setting that reduces confusion for the workspace owner, since it is difficult to distinguish between a hard link to a cache and a local file in the workspace.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

How DesignSync Handles Symbolic Links

Projects

What Is a Project?

A DesignSync project is an infrastructure designed to support a real-world design project. A project infrastructure includes:

- The vault location that is used to store project files.
- The cache and mirror directories that will be used for sharing project files (UNIX only).
- The design methodology (such as lock versus merge model, or check-in comments required versus not required) that is to be enforced for the project.

Project attributes are defined by the project leader or system administrator and are inherited by all project members. Once a project leader or system administrator defines a "Public" project, all users that have access to that project will inherit the project's attributes. The term "Public Projects" is used to define those projects that can be accessed by multiple users in DesignSync. (For more information on projects, see DesignSync Data Manager User's Guide: What is a Project?.)

Using SyncAdmin, or by defining the project in a TCL script, your LAN administrator can define or modify the following project attributes:

- A project name and description for each project that your team will be working on
- The name and location of the vault that stores your project files
- The name and location of the cache directory that is being used on your LAN

Note: Currently, the **Cache Directory** is only used in a UNIX environment. Although the LAN administrator must define a cache directory, it has no effect in a Windows environment.

Related Topics

Projects Options

Defining a Public Project

How DesignSync Determines the Correct Project Cache

Projects Options

A DesignSync project is an infrastructure designed to support a real-world design project. The **Projects** options pane in the **Synchronicity Administrator** window allows a LAN administrator to define projects that will be used by the members of your development team. As a project is defined, it is listed at the top of the **Synchronicity Administrator** window along with any of its configuration parameters. This list is updated as information is added and/or deleted using the SyncAdmin tool.

- To add a project, click **Add Project**.
- To edit an existing project, highlight the project in the window and click **Edit Project**.
- To delete an existing project, highlight the project in the window and click **Delete**.

Important: If your system uses a large number of projects and project cache definitions, you may want to, instead of defining projects in SyncAdmin, define them in a TCL script which loads definitions on an as needed basis, rather than loading all project definitions at client start time. For more information see

Related Topics

What is a Project?

Defining a Public Project

Project Registry Keys (ProjectCacheTclScript) and (Projects)

Add or Edit a Project

To create a project, after clicking **Add Project**, fill in the fields in the Add Project pane.

To edit an existing project, after clicking **Edit Project**, edit the fields on the Edit Project pane.

Click on the image for more information about each field.

Project Name

The name you enter in the **Project Name** field will appear in the Tree View of DesignSync under **Public Projects**. The following naming conventions apply to project names:

- The name can consist of up to 40 alphanumeric characters.
- No spaces are allowed.
- Underscores, periods, and dashes are allowed.

Once a project is defined, DesignSync users can join a Public Project by selecting it in the DesignSync GUI or by using DesignSync's Workspace Wizard.

Description

The description of the project that you enter in the **Description** field will appear in the List View of DesignSync when the project is selected from the list of **Public Projects**. The description can consist of up to 80 alphanumeric characters but is not required to define a new project.

Vault URL

To allow your work to be shared by others using DesignSync, you need to define the remote vault folder that will contain your revision-controlled project files. This vault folder represents the top level of the SyncServer from which all data is managed. Once files are placed in the vault, other members of your design team can access them. The URL that you enter can be that of any valid DesignSync server (SyncServer) in the world and is not restricted to a SyncServer installed in your local area network.

Important: You can define a module category as a public project. If you do, the project must be defined with a trailing slash (/) after the category name.

To specify the **Vault URL**, use the format appropriate for your SyncServer:

```
sync://<host>:<port>/<subfolder_for_project> (or  
syncs://<host>:<port>/<subfolder_for_project>)
```

or for module categories:

```
sync://<host>:<port>/Modules/<category_name>/ (or  
syncs://<host>:<port>/Modules/<category_name>/)
```

Where:

<host> is the computer where the SyncServer is running.

Important: Specify the host name exactly as it appears in the ServerName property of the `httpd.conf` file located in the directory:

```
SYNC_CUSTOM_DIR/servers/<host>/<port>/conf. (This ServerName property is  
derived from information specified when the system administrator configured the server  
during DesignSync installation.)
```

<port> is the port used to communicate with that SyncServer. Specify the port (Cleartext or SSL) that users specify when they use the `setvault` operation to associate their workspaces with the SyncServer.

<subfolder_for_project> is the subfolder, including the path on the server, where you want to store the project files.

For example, suppose you want to create a Module called `Design1` on a server called `granite`. The `ServerName` property in the `httpd.conf` file is:

```
ServerName granite.ABCo.com:30046. You know that users specify that same  
port (33046) when they use the setvault operation.
```

So for Vault URL, you would specify:

```
sync://granite.ABCo.com:30046/Modules/Design1
```

If you wanted to specify the `Tools` category on the `tools` server as a public project, you would specify the Vault URL as follows:

```
sync://tools.ABCco.com:30046/Modules/Tools/
```

Cache

During the installation of DesignSync software, the system administrator defines a default cache area. A cache is necessary for multiple users to share project files. In a UNIX environment, the default cache is typically `<SYNC_DIR>/../sync_cache`.

When you define a project using the SyncAdmin tool, you can associate a specific cache with that project.

Because caching is implemented using UNIX links, its use is limited to a UNIX environment. Although not used as a means of sharing DesignSync files on Windows platforms, DesignSync does use the cache for other housekeeping purposes. Therefore, you must define the **Cache** when defining a project.

In a UNIX environment, if a user specifies the "share" option when checking files into DesignSync, the cache is updated with new versions of files. Links to cached files on UNIX are static. The link to a particular version in the cache directory remains unchanged until you request an update using the "check-out" or "populate" operations. Because multiple versions of a file can exist in the cache, the amount of disk used is potentially larger than the disk usage for mirrors. Versions in a cache that are no longer linked to by a user are automatically purged from the cache (See Mirrors Versus LAN Caches in DesignSync Help for more information on the use of the cache directory.)

When DesignSync starts, it compares the vault folder against the project cache, and matches the cache/vault folder association to the longest cache entry it can find. For more details, see How DesignSync Determines the Correct Project Cache.

Using the SyncAdmin tool, a LAN administrator can associate a specific cache to a specific project. The following restrictions apply to the **Cache** field.

- The directory name must be valid. (If the directory does not exist, the system will create it for you.)
- The directory must exist on your local area network.
- The directory must be accessible to other users on your LAN.

Note: If the directory that you entered in **Cache** field does not exist, the system will create it for you.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Defining a Public Project

To define a Public Project, follow the steps below:

1. In the **Projects Options** pane, click **Add Project**.
2. Click in the **Name** text-box and enter the name of the "Public" project that you want to define.
3. Click in the **Description** field and type in a brief description of the project being defined. (Optional)
4. Click in the Vault URL text-box and enter a valid server-side vault URL.
5. Click in the Cache Directory text-box and enter a valid cache directory.

Note: If necessary, click on the **Browse** button to navigate your local area network. Select a valid directory and click on the **OK** button to accept your selection.

Related Topics

What is a Project?

Projects Options

Revision Control (RC) Notes

Revision Control (RC) Notes Options

The **Revision Control Notes** options pane allows you, the DesignSync tools administrator, to enable DesignSync to generate RevisionControl notes during revision-control operations such as check in, check out, and populate. By default, RevisionControl notes are disabled.

When you enable RevisionControl notes, DesignSync generates one note for each revision control command even if the command affects multiple files, such as a recursive checkin.

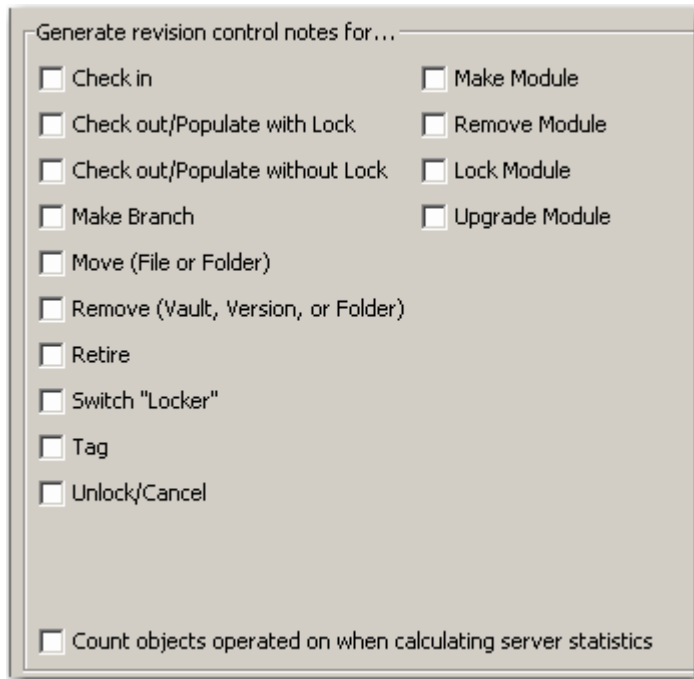
DesignSync does not attach notes to objects. However, an Objects field on the RevisionControl note lists the objects that have been changed by the revision-control operation and the resulting e-mail notification includes the list. (For information on setting up this e-mail notification, see Configuring Email Notifications).

RevisionControl notes generated for a check-in operation include both the comment specified for the check-in operation and the comment specified when the object was checked out. These comments are also included in the automatic e-mail notification of the RevisionControl note.

Notes:

- When you check-in an object with the -reference option and only the state of the object has changed, DesignSync does not add the object to the Objects field on the RevisionControl note.
- DesignSync note generation is available for remote vaults only. You cannot enable note generation for local (client) vaults.
- Generation of revision-control notes is disabled by default because the operations for which notes are generated is a system administration preference.

Click on the image for more information about each field.



Commands

Check In

When an object is checked in, a snapshot of a design object is stored in a vault. The new snapshot is called a version. As a LAN administrator, you can choose if you want a RevisionControl note to be generated for this DesignSync operation. (See DesignSync Data Manager User's Guide: Checking in Design Data for more information.)

Check Out or Populate with lock

When an object is checked out by a check-out or populate operation, the object is available in your local work area. You can check out a locked or unlocked copy of the object, a link to a cache or mirror copy of the object, or a locked or unlocked reference to the object.

When an object is checked out or populated with a lock, only the user who has the lock can check in a newer version of the object on that branch. You should use this option when your project team uses the locking model (as opposed to the merging model) and you intend to make changes to an object. To remove the lock use the 'checkin or cancel command.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when users check out or populate objects with a lock. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: Checking out Design Data and Populating Your Work Area.*)

Check Out or Populate without lock

When an object is checked out by a check-out or populate operation, the object is available in your local work area. You can check out a locked or unlocked copy of the object, a link to a cache or mirror copy of the object, or a locked or unlocked reference to the object.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when users check out or populate objects without a lock.

Note: RevisionControl notes are not generated when checking out unlocked references, or links to the cache or mirror. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: Checking out Design Data and Populating Your Work Area.*)

Make Branch

Separate branches may be created for projects that require multiple lines of development (parallel development).

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a new branch is created. (For more information, see *DesignSync Data Manager User's Guide: Parallel (Multi-Branch) Development.*)

Move (File or Folder)

You can use the `mvfile` command to move or rename a specified local file or collection object (for example, a Cadence cell view. You also can use the `mvfolder` command to move or rename a specified local folder.

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a file, collection object, or folder is moved or renamed. See the `mvfile` and `mvfolder` commands descriptions in the *ENOVIA Synchronicity Command Reference* for more information.

Remove (Vault, Version, or Folder)

You can delete vaults, versions in vaults, and folders, for non-module data. As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a vault, version, or folder is removed.

See the `rmvault` and `rmfolder` command descriptions in the *ENOVIA Synchronicity Command Reference* for information on removing vaults and folders. For information on removing versions, see *DesignSync Data Manager User's Guide: Deleting Versions from a Vault*.

Retire

You can retire design objects when they become obsolete. As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when design objects are retired.

For more information, see *DesignSync Data Manager User's Guide: Retiring Design Data*.

Switch "Locker"

An administrator with permission can use the `switchlocker` command to change the lock owner of a design object. (By default, access controls deny this command to all users.)

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when the `switchlocker` command is used to change a lock owner. See the `switchlocker` command description in the *ENOVIA Synchronicity Command Reference* for more information.

Tag

A tag is an identifier that you attach to a group of design object versions that work together as a configuration. You can then perform operations on the group of objects as a whole. For example, you might tag all versions that went into a release "Rel2_0" so that at a later date another user could fetch all versions tagged "Rel2_0" thereby recreating that release.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when design objects are tagged. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: What is a Design Configuration and Tagging Versions and Branches*.)

Unlock/Cancel

You can unlock objects that have been locked by you or other team members. You can also cancel a lock of objects that you have locked in your own workspace. (For more information, see *DesignSync Data Manager User's Guide: Unlocking Server Data*.)

As a LAN administrator, you can choose to have a RevisionControl note generated when users unlock design objects.

Make Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module is created on the server.

For more information, see *DesignSync Data Manager User's Guide: Creating a Module*.

Remove Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module is deleted from the server.

For more information, see *DesignSync Data Manager User's Guide: Deleting a Module*.

Lock Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module branch is locked.

For more information, see *DesignSync Data Manager User's Guide: Locking Module Data*.

Upgrade Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a legacy module is upgraded.

For more information, see *DesignSync Data Manager User's Guide: Importing Legacy Modules*.

Count objects operated on when calculating server statistics

DesignSync generates a single note for each revision control command, however many objects are operated upon.

As a LAN administrator, you can choose whether server statistic reports are generated based on the number of notes or the number of objects recorded within the notes.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. (In most cases, this is the site registry file.) After saving any changes, SyncAdmin exits.

Note: There may be situations where you want to implement server-specific settings. See [Enabling/Overriding RevisionControl Note Generation for Specific Servers](#) for information.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. (In most cases, this is the site registry file.) After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Enabling Note Generation](#)

[Registry Files](#)

Enabling Note Generation

As DesignSync administrator, you can enable DesignSync to generate a RevisionControl note each time a specified revision control operation takes place. You can then view and query these notes. You can also use the generation of a RevisionControl note to trigger some other action, such as running a Tcl script.

Note: For more information on RevisionControl notes and their uses, see the [RevisionControl Notes Overview](#).

To enable note generation for an entire site, use SyncAdmin to modify the site settings:

1. On the UNIX server machine, start the SyncAdmin tool:

```
%SyncAdmin &
```


2. Select **Change Site Settings**. Then click **OK**.
3. Select **Site Options=>RC Notes**.
4. Turn on the RevisionControl note generation for the command you want and click **OK**. (See the RevisionControl Notes Options for information on each command.)
5. Restart any SyncServers that share the same SYNC_DIR installation directory to have the settings take effect.
6. Any clients that access the server need to be restarted as well, so they recognize the change in the server's RevisionControl note setting.

Notes:

There may be situations where you want to implement server-specific settings. This is possible if your servers are on Unix machines. For more information, see [Enabling/Overriding RevisionControl Note Generation for Specific Servers](#).

Related Topics

[RevisionControl Notes Options](#)

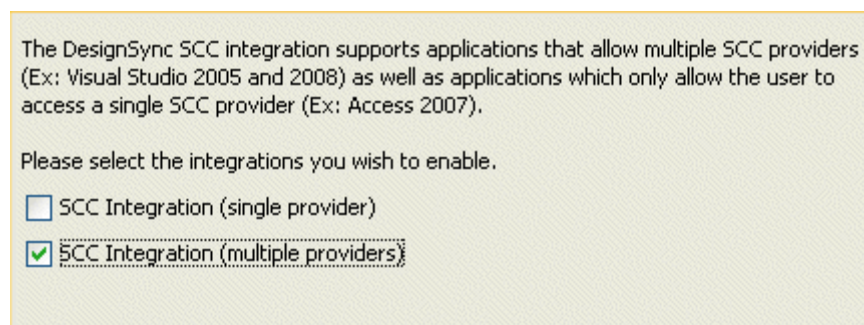
SCC Provider

Within DesignSync, you have the ability to perform revision control operations through the Windows SCC (Source Code Control) engine. Windows has two possible SCC engines that you can use. One of the SCC engines allows you to optionally specify multiple SCC providers, for example Microsoft Visual Studio 2005 and 2008. The other SCC engine only allows you to use a single SCC provider, for example Microsoft Access 2007.

Note: If you specify the single-provider SCC integration, DesignSync will be only enabled SCC application on your system.

Both versions can be installed on your system if you use applications that require different SCC integrations.

Click on the image for more information about each field.



SCC Integration (single provider)

Enables DesignSync Source Code Control integration for applications allowing only a single SCC provider. For example, Access 2007.

SCC Integration (multiple providers)

Enable DesignSync Source Code Control integration for applications allowing multiple SCC providers. For example, Visual Studio 2008 and 2010.

3DPassport

This feature is reserved for future development.

Third Party Integration Options

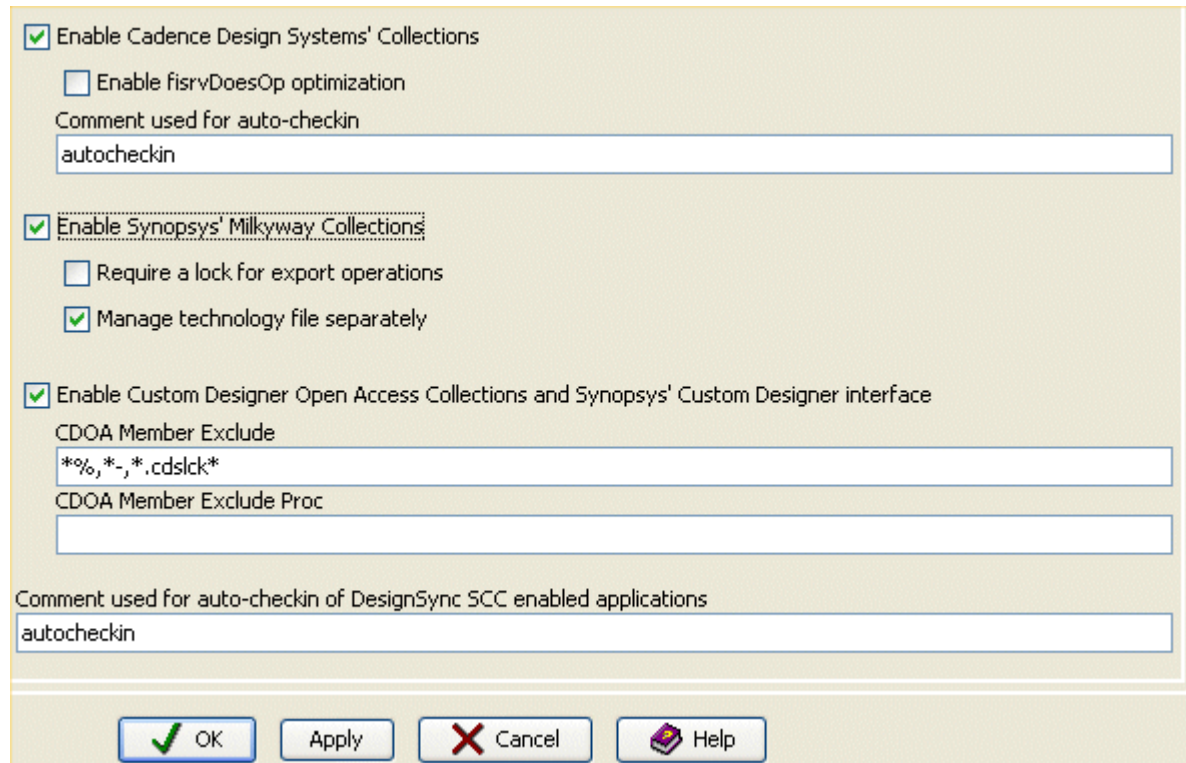
Within DesignSync, you have the ability to revision control third-party design data. This design data is handled as a collection. Collections in DesignSync are groups of files that together define a design object. For example, a schematic might consist of dozens or even hundreds of files within any number of folders. Each collection is operated on as a single revision controllable object while your design tools manage the object at the file level. For more information, see DesignSync Data Manager User's Guide: Collections Overview.

DesignSync supports Cadence Design Systems and Synopsys Custom Compiler cellviews as collection objects. The cellview collection is part of DesignSync's general support for Cadence and Synopsys libraries.

IMPORTANT: While the Synopsys MilkyWay integration options are still visible, the integration has been deprecated.

By default, the recognition of collections is disabled for Cadence and Synopsys cellviews; however, the system administrator can enable Cadence or Synopsys recognition during the installation of DesignSync software. The **Third Party Integration** options pane in the **Synchronicity Administrator** window also allows you, as a LAN administrator to enable or disable the use of Cadence and Synopsys collections on your LAN. SyncAdmin writes the selections you make in this pane to the appropriate registry file you chose upon invoking SyncAdmin.

Click on the image for more information about each field.



Enable Cadence Design Systems' Collections

Enable fisrvDoesOp optimization

Comment used for auto-checkin
autocheckin

Enable Synopsys' Milkyway Collections

Require a lock for export operations

Manage technology file separately

Enable Custom Designer Open Access Collections and Synopsys' Custom Designer interface

CDOA Member Exclude
%,-,*.cdslck*

CDOA Member Exclude Proc

Comment used for auto-checkin of DesignSync SCC enabled applications
autocheckin

Enable Cadence Design System's Collections

The recognition of Cadence objects is only available in a UNIX environment. Support for Cadence objects is determined during DesignSync client installation and requires a DesignSync DFII license. If you did not have a DesignSync DFII license when the software was initially installed, but you later acquire one, you can use this option to enable Cadence object recognition. (For more information about DesignSync DFII, see DesignSync Data Manager User's Guide: Cadence Design Objects Overview.)

Notes:

- Choosing this option after not selecting DesignSync DFII support at installation time enables Cadence object recognition within DesignSync.
- If you have both Cadence collection recognition and Synopsys Custom Designer collection enabled, Cadence collection recognition takes precedence and new objects will be presumed as Cadence collection options.

Enable fisrvDoesOp optimization

This option enables or disables the fisrvDoesOp optimization.

The fisrvDoesOp optimization lets operations through the Cadence GDM system execute data management operations such as checkins and checkouts directly through the libgdmSync library and the syncfisrvpp executable rather than spawning a separate

stcl process. Executing such operations in this way can improve performance and reliability.

Note: It is recommended that this option remain enabled.

Comment used for auto-checkin

This comment is used by the Cadence system as a comment for automatic, silent, checkins that take no user input. By default the comment is "autocheckin."

Note: The comment length is not restricted to the length of the box, but you cannot specify a return or line feed character (CR/LF).

Enable Synopsys' MilkyWay Collections

This option enables or disables DesignSync recognition of MilkyWay objects. The option is selected if, during client installation, you enabled Synopsys MilkyWay Tools and Data.

DesignSync support for Synopsys Milkyway objects requires a DesignSync MW license. If you did not have a DesignSync MW license when the software was initially installed, but you later acquire one, you can use this option to enable Synopsys Milkyway object recognition.

Note: The recognition of Synopsys objects is available only in a UNIX environment.

Require a lock for export operations

This option enables or disables the checkout with a lock of these files before an export operation (export library information or the export step of the setup library operation):

- The technology file (`tech.tf`)
- The list of reference libraries (`refs.txt`)
- The Other library information shadow object (`shadow.sync.mw.lib`)

If this option is enabled, DS MW locks these files before performing an export. Then if the checkout with a lock fails, the export operation fails. A project team or site would use this setting if they follow a strict locking model for files. By default, this option is disabled and DS MW does not lock the files before performing an export operation.

Manage technology file separately

This option enables or disables the management of the technology file (`tech/tech.tf`) as a separate file. If the option is enabled (checked), DS MW manages the technology file as a separate file and not part of the Other library information. This is the default setting. If the option is disabled, DS MW exports and imports the technology file as part of the Other library information.

Enable Custom Designer Open Access Collections and Synopsys Custom Designer interface

This option enables or disables DesignSync recognition of CDOA collection objects. The option is selected if, during client installation, you enabled Synopsys Custom Designer tools and Data

DesignSync support for CDOA objects requires a DesignSync CD license. If you did not have a DesignSync CD license when the software was initially installed, but you later acquire one, you can use this option to enable Synopsys CDOA object recognition.

Note: The recognition of Synopsys objects is available only in a UNIX environment.

CDOA Member Exclude

Enter the objects you want to exclude from revision-control operations, separated by commas. By default, the following masks are excluded from checkins.

```
*%,*-,*.cdslck*
```

CDOA Member Exclude Proc

Indicates the name of a TCL procedure that can be called with each potential member of a CDOA collection. The procedure must return a value of 1 if the file should be included in the collection or 0 if it should not.

Comment used for auto-checkin of DesignSync SCC enabled applications

This comment is used by the DesignSync SCC plug-in (DSVS) as a comment for automatic, silent, checkins that take no user input. By default the comment is "autocheckin."

Note: The comment length is not restricted to the length of the box, but you cannot specify a return or line feed character (CR/LF). For more information about DesignSync DSVS, see DesignSync Data Manager VS User's Guide. DSVS is enabled within SyncAdmin. For more information, see **SCC Provider**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made on one of the tabs.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

GUI Customization - States

Vault Types

Vault Type Options

As a Synchronicity Administrator, you can support multiple vault types at your site. This enables you to manage different data types, and convert data from one vault type to another vault type. DesignSync currently supports three vault types, RCE Vault, Copy Vault, and Smart Vault. (Copy Vault and Smart Vault have an optional Zipped format). (For more information, see About Vault Types).

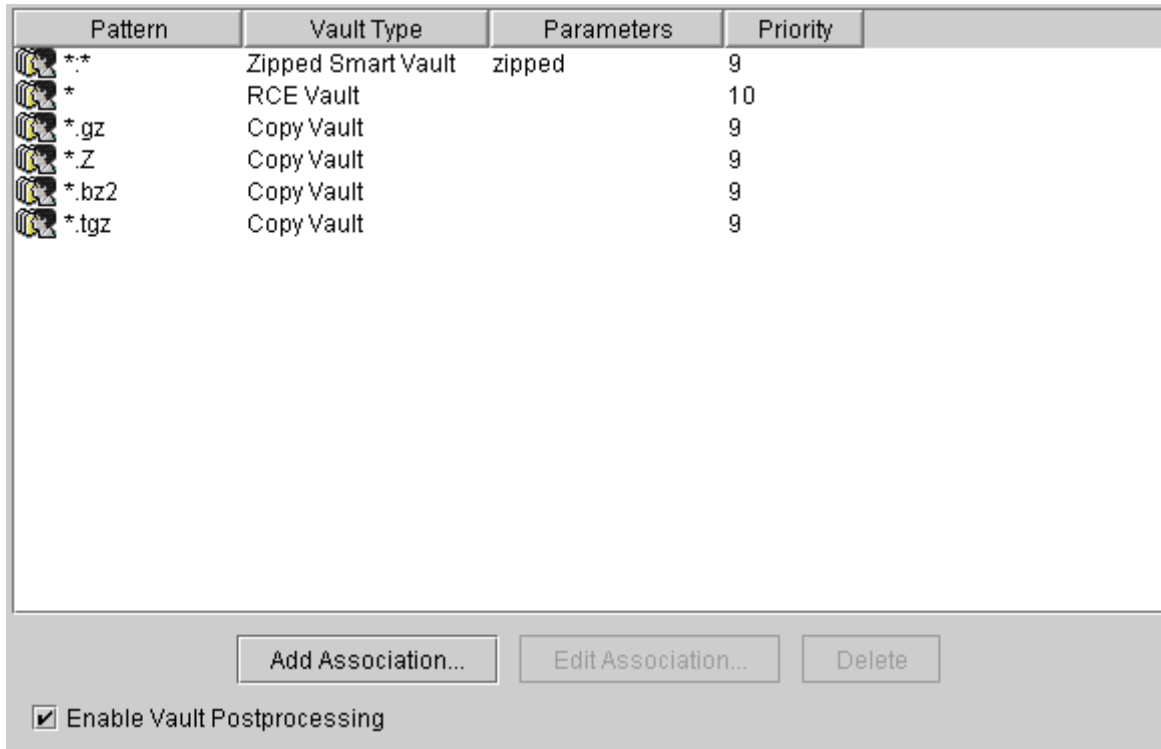
The information about vault types is read, in the following order, from these files: `PortRegistry.reg`, `SiteRegistry.reg`, `EntRegistry.reg`, and finally from the software. The default vault type entries are:

- RCE Vault with a pattern `*`, and a priority of 10.
- Copy Vault with a pattern of `*.gz`, `*.tgz`, `*.Z`, and `*.bz2`, and a priority of 9
- Zipped Smart Vault with a pattern of `*:*` and a priority of 9, for Milkyway member files. (Deprecated.)

Note:

SyncAdmin writes to the site-wide `SiteRegistry.reg` file, which is read by all servers configured from the site installation. To set vault type mappings for a specific server, use the DesignSync Web UI **Admin Menu | Server | Administer Server | Vault Types** options pane.

To support multiple vault types successfully, you need to define the vault types at your site. You manage vaults at your site by assigning these settings: **Pattern**, **Vault Type**, **Parameters**, and **Priority** on the **Vault Types** tab. SyncAdmin records the settings in the `SiteRegistry.reg` file.



- To add a vault type, click **Add Association**.
- To edit an existing vault type setting, highlight the trigger in the window and click **Edit Association**.
- To delete a vault type, highlight the trigger in the window and click **Delete**.
- If you define a Smart Vault and want to enable post check-in optimization, click the checkbox labelled **Enable Vault Postprocessing**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Adding or Edit Vault Type](#)

[Setting Vault Types](#)

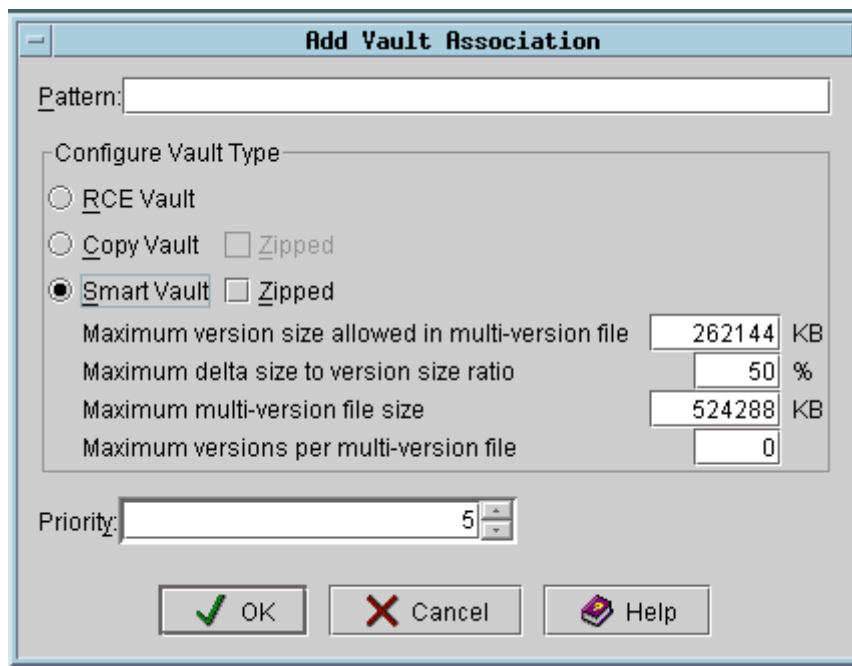
[About Vault Types](#)

Add or Edit Vault Association

To add a vault type setting, on the **Vault Types** options pane, click **Add Association** and fill in the fields in the **Add Vault Association** pane.

To edit an existing vault type setting, on the **Vault Types** options pane, click **Edit Association**, and edit the fields on the **Edit Vault Association** pane.

Click on the image for more information about each field.



Add Vault Association

Pattern:

Configure Vault Type

RCE Vault

Copy Vault Zipped

Smart Vault Zipped

Maximum version size allowed in multi-version file KB

Maximum delta size to version size ratio %

Maximum multi-version file size KB

Maximum versions per multi-version file

Priority:

Pattern

The **Pattern** is a non-verified string that DesignSync uses to assign objects to their vault types. In adding or editing a vault type, you must specify a pattern.

To assign an object to a vault type, DesignSync compares the objects's entire server URL path to a pattern. (The order in which patterns are compared to the object URL depends on the Priority value for the pattern.) The Pattern value is used in the following ways:

- If the value of **Pattern** is `*`, DesignSync maps all objects to the assigned vault type. The **Priority** value for this pattern is always 10.
- If the value of **Pattern** is `*.tgz`, DesignSync maps any file name that contains `.tgz` to the assigned vault type.
- Because the DesignSync compares the objects's entire server URL path, you can use parts of the URL path other than the file name to assign a vault type. For example, to assign objects associated with the Logic Design project to the Zipped vault type, you would specify the Pattern `*/Projects/ LogicDesign/*` and select Zipped as the Vault Type.

Note: When you work with Cadence collection objects you must specify a vault type pattern for the member files. If, for example, you specify a pattern match of `*.sync.cds`, the pattern match is ignored because there is no such object on the disk. However, if you change the default `"*"` pattern match from RCE Vault to Copy Vault, then `*.sync.cds.cpv` vault files are created.

Configure Vault Type

Click the selector arrow to choose between the RCE Vault (`.rca` file extension), Copy Vault (`.cpv` file extension), or Smart Vault. Copy Vault and Smart Vault also provide an optional feature that allows you to compress files with specified extensions to save disk space (Zipped). For more information, see About Vault Types.

Smart Vault Options

If you select Smart Vault, you have the option to set the maximum size of additional parameters:

- **Maximum version size allowed in multi-version file**
Enter the maximum file size in KB. Enter 0 to indicate no maximum size.
- **Maximum delta size to version size ratio**
Enter a percentage between 1 and 100. To turn this parameter off, set it to 0.
- **Maximum multi-version file size**
Enter a file size in KB. Enter 0 to indicate no maximum size.

- **Maximum versions per multi-version file**
Enter the number of versions you wish to allow for each multi-version file.
Enter 0 to indicate no maximum number.

Pattern Priority

The **Priority** value determines the order in which patterns are compared. You can specify a value from 1 (highest priority) to 9 (lowest). For example:

You set vault-type,V1, to have a **Priority** of 5 and a **Pattern** of '*.tgz'.

You set vault-type,V2, to have a **Priority** of 3 and a **Pattern** of '*.tgz'.

In this case, a file, `module.tgz` will match both V1 and V2, but V2 takes precedence over V1, as V2's priority is numerically smaller than V1's.

Note: When you use a **Pattern** of '*', SyncAdmin enters the default priority of 10.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Vault Types](#)

[Setting Vault Types](#)

[About Vault Types](#)

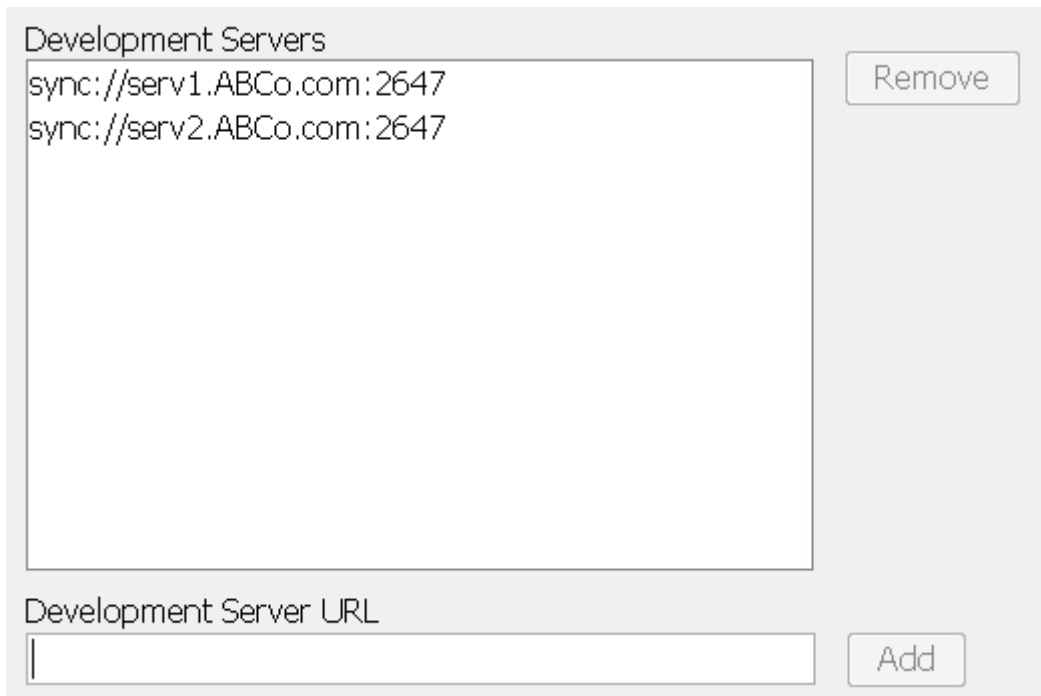
Development Servers

The Development Servers options page allows you to define DesignSync development area servers and view the list of development servers. In order to deploy DesignSync

Development areas to your clients, you must first have defined one or more development servers.

For more information on using DesignSync Developments, see the *Enterprise DesignSync Administration User's Guide*.

Click on the image for more information about each field.



The screenshot shows a configuration window titled "Development Servers". It features a list box containing two entries: "sync://serv1.ABCo.com:2647" and "sync://serv2.ABCo.com:2647". To the right of the list box is a "Remove" button. Below the list box is a text input field labeled "Development Server URL" and an "Add" button.

Development Servers

Displays an alphabetical list of the development servers defined for use on this client. If no development servers are defined, this list is empty. You can select an item in the list to remove it. You cannot modify a defined development server. If a development server changes name, add a new definition and remove the old one.

Remove

Removes the selected development server location from the available Development Server list.

Development Server URL

Enter the SyncServer URL for the server hosting the shared developments. After typing the name of the server click the Add button. The URL must be entered in the form:

```
sync[s]://<server>[:<port>]
```

IMPORTANT: The URL is checked for syntactical accuracy, but is not validated against any specified server. This allows you to add servers that are not currently running or are on a remote site, if you are defining definitions to deploy enterprise-wide.

Add

Adds the Development Server URL to the list of Development Servers.

Related Topics

Enterprise Servers

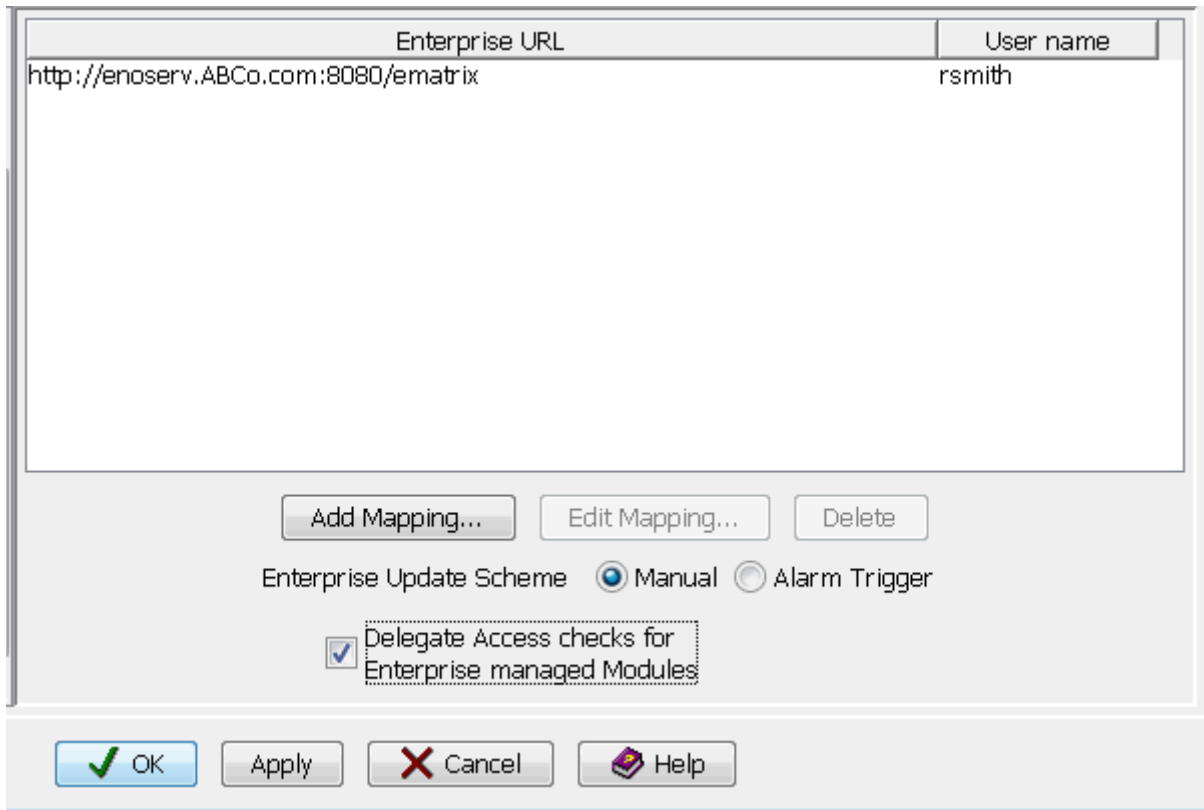
Enterprise DesignSync Administration User's Guide: Overview of Enterprise DesignSync Administration

Enterprise Servers

Enterprise Servers

The Enterprise Servers options page lists the defined Enterprise servers associated with DesignSync objects. In order to deploy DesignSync Development areas to your clients, you must first have defined one or more development servers.

Click on the image for more information about each field.



List of Enterprise Servers

Shows the list of Enterprise Design server mappings and their associated properties:

- Enterprise System URL
- Username to login on the Enterprise system.
- Priority order in which the mapping is used.
- Username to login on the Enterprise system.

Add/Edit Mapping

Launches either the Add Mapping/Edit Mapping dialog.

Delete

Delete the selected mapping. You will be prompted to confirm the deletion.

Enterprise update scheme

When updates are made to modules in the DesignSync system, DesignSync can automatically synchronize those changes with the corresponding Enterprise Design Objects based on the setting you select:

- Alarm trigger - Uses the alarm trigger mechanism to determine when to check for queued updates to send to the Enterprise server.
- Manual - Disables automatic synchronization. When automatic synchronization is enabled, you can use one of several methods to update the Enterprise Design Objects. For more information on manual synchronization methods, see Enterprise Design Synchronization Queue. (Default)

For more information on using DesignSync Developments, see the *Enterprise DesignSync Administration User's Guide*.

Delegate Access Checks for Enterprise Managed Modules

When DesignSync is used as part of an enterprise system, some DesignSync and Enterprise Design objects are maintained, in synchronization, across both platforms. In order to control access to design elements, DesignSync allows you to map DesignSync permissions to ENOVIA permissions to allow appropriate access to design objects. When Delegate Access Checks for Enterprise Managed Models is enabled, modules which are Platform connected have all the access checked delegated to the connected Platform.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin and the dialog box closes..

Apply

When you click on the Apply button, SyncAdmin saves the changes, and leaves the dialog box open.

Cancel

When you click on the **Cancel** button, SyncAdmin exits this page, without saving any changes you have made.

Help

When you click on the **Help** button, SyncAdmin launches a web browser and opens this topic..

Related Topics

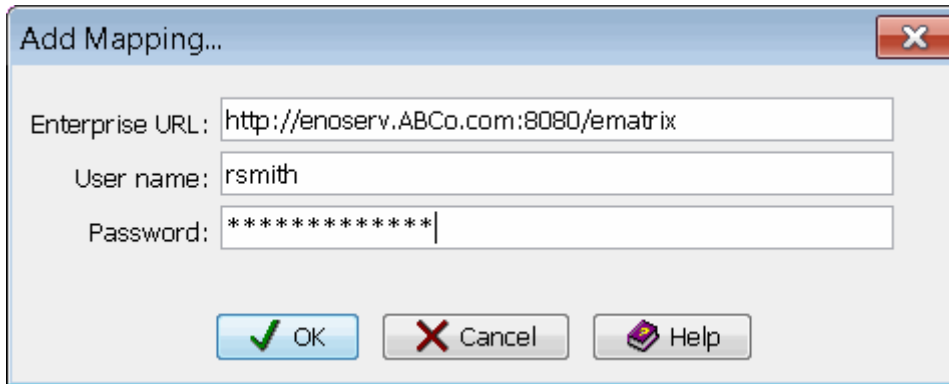
Site Options

Enterprise DesignSync Administration User's Guide: Enterprise Design Synchronization

Add/Edit Mapping

The Enterprise Servers options page allows you to define the Enterprise servers associated with module URLs in DesignSync,

Click on the image for more information about each field.



Enterprise URL

The Enterprise URL for the project associated with the DesignSync module, including the server name, port name, and top level directory location, for example:

```
http://enoserv:8080/ematrix
```

User name

Username on the ENOVIA server. If the Username is not provided, the user may receive an error when attempting to connect through the ENOVIA client.

Password

Password for the specified ENOVIA username.

Command Buttons

Standard dialog buttons.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin.

Cancel

When you click on the **Cancel** button, SyncAdmin exits this page, without saving any changes you have made.

Help

When you click on the **Help** button, SyncAdmin launches a web browser and opens this topic..

GUI Options

GUI Options

The **GUI Options** pane allows you to set the following options on the DesignSync graphical user interface.

Click on the image for more information about each field.

The screenshot shows a window titled 'GUI Options' with the following settings:

- Synchronize graphical and command-line interfaces
- Always refresh on tree selection
- Show data sheet in frame
- Confirm use of Populate "Force overwrite of local modifications" option

At the bottom, there are two spinners:

- Show stale icon after how many minutes: 0
- Automatically refresh after how many minutes: 0

Synchronize graphical and command-line interfaces controls whether selection and current directory are synchronized between the tree and list views and the command bar.

If this option is selected, a selection in the tree view results in the execution of an **scd** command, and selections in the list view result in the execution of **select** and **unselect** commands.

Similarly, executing **cd**, **select**, and **unselect** commands in the command bar result in selections being made in or removed from the tree and list views.

Always refresh on tree selection refreshes the List View anytime you select a folder in the Tree View. This ensures that the information displayed in the List View is always up-to-date. However, it takes time to update the information. If this option is not enabled, there is no automatic refresh; subsequent visits to a folder cause a quick re-display of the List View, but the data is stale and could be incorrect. See the **Automatically refresh after how many minutes** option for an alternate approach.

Show data sheet in frame determines whether data sheets are shown in the View Pane. If this is not selected, data sheets are always displayed in a browser.

Note: Choosing to display data sheets in a browser disables embedded URL links to vaults, branches, and versions.

Confirm use of Populate "Force overwrite of local modifications" option determines whether a confirmation dialog is displayed when the **force** option is selected in the populate dialog. This can prevent the accidental overwriting of files in your work area.

Show stale icon after how many minutes indicates how many minutes to wait before displaying a stale data icon, indicating that the information for that object may be obsolete and should be refreshed. If 0 is specified, the stale data icon is never shown.

Automatically refresh after how many minutes indicates how many minutes to wait before automatically refreshing objects. If 0 is specified, objects are never automatically refreshed. This option is only recommended when a fast connection to the server is available.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Initial Folder Options

The **Initial folder** options pane determines whether a folder is initially selected in the tree view. The choices are:

The screenshot shows a dialog box titled "Initial folder". It contains four radio button options: "Current directory", "Home directory", "None", and "This directory:". The "This directory:" option is selected. Below the options is a text input field containing the path "/home/tjames/test" and a "Browse..." button.

- Initially select the **Current directory** in which DesignSync was started. (This option is not available on Windows.)
- Initially select the **Home directory**.
- Do not initially select a folder (**None**).
- Choose **This folder** by clicking the **Browse** button and selecting a folder.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

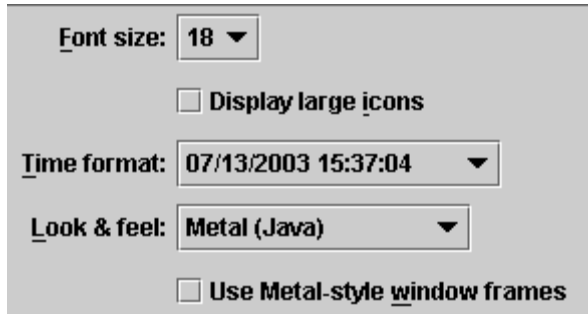
When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Display Options

The **Display** options pane is used to customize the way data is displayed by the GUI. The following options are provided:



The screenshot shows a grey panel with the following controls:

- Font size:** A dropdown menu set to '18'.
- Display large icons:** An unchecked checkbox.
- Time format:** A dropdown menu showing '07/13/2003 15:37:04'.
- Look & feel:** A dropdown menu set to 'Metal (Java)'.
- Use Metal-style window frames:** An unchecked checkbox.

- **Font size** determines the size of the font used in the Tree View and List View.
- **Display large icons** controls whether large or small icons are used in the Tree View and List View.
- **Time format** determines the time format used for dates displayed in the GUI. Choose a format from the drop-down menu.
- **Look and feel** allows you to select a look and feel for the GUI. This change will not take effect until DesignSync is restarted.

Note: Some Look and Feel choices may result in unexpected behavior in text entry fields, such as data you enter displaying in bold.

- **Use Metal-style window frames** replaces the default window manager frames with frames that match the metal look and feel. This option is available only when **Metal** is selected for **Look & Feel**, and some window managers may not support metal-style frames. When using a remote display, metal-style frames may give better performance.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

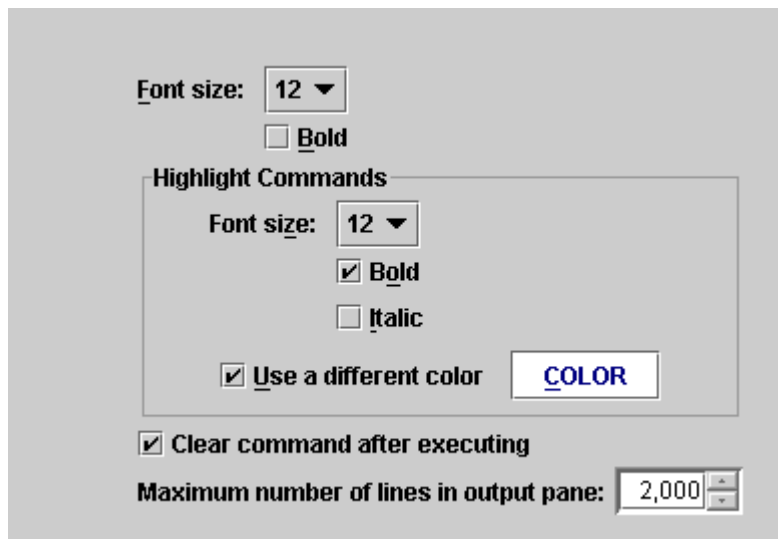
When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Command Bar Options

The **Command Bar** options pane allows you to change the appearance of the Output Pane. The Command Bar pane is only available when you select **Tools=>Options=>GUI Options=>Command Bar** from the DesignSync GUI. (The Command Bar pane is not available when you invoke a standalone SyncAdmin session from the command line or from the Start menu.)



The following options are available:

- **Font size** determines the font size used to display the output of commands.
- Check the **Bold** box to indicate whether commands should be in bold.
- **Highlight Commands**
 - **Font size** determines the font size used to display commands that were typed into the command bar or executed as a result of a menu choice.
 - Check the **Bold** and **Italic** boxes to indicate whether commands should be highlighted in bold or italic.
 - **Use a different color** indicates that commands should be displayed in a different color. Click the **COLOR** button to display the Color Selection pane.
- **Clear command after executing** clears the command from command bar display after the command executes.

- **Maximum number of lines in output pane** helps control the memory usage of DesignSync. The default limit is 2000 lines. The limit affects visible lines only; it does not affect lines hidden by verbosity setting.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

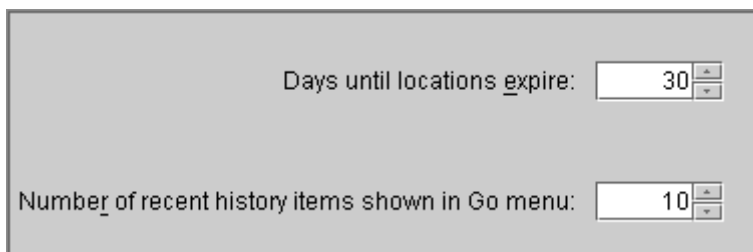
When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

History Options

The **History** options pane allows you to set the following properties:



The screenshot shows a dialog box with a light gray background. It contains two settings, each with a text label, a numeric input field, and a small up/down arrow icon to the right of the field. The first setting is 'Days until locations expire:' with the value '30'. The second setting is 'Number of recent history items shown in Go menu:' with the value '10'.

Days until locations expire

Specifies the number of days before locations are removed from the Location/Last Visited list (as displayed in the History dialog); the default is 30. **Note:** To clear the list of locations each time the DesignSync GUI starts up, specify a value of 0 for this option.

Number of recent history items shown in the Go menu

Specifies the number of locations displayed in the recent history list at the bottom of the **Go** menu. The default is 10.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the Apply button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the Apply button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Columns Options

The **Columns** options pane allows you to customize the columns that appear in the list view. You can display this pane by selecting the **Tools=>Options=>GUI Options=>Columns** menu choice, or by clicking the right mouse button on the list view header and selecting **Show/Hide Columns** from the context menu.



This pane lists all of the columns that can be displayed in the list view. By checking and unchecking the boxes, you can choose which columns are displayed and which are hidden. For example, by unchecking the **Size** column check box, you can remove the Size column from the list view.

If this page is displayed by selecting **Show/Hide Columns** from the list view header's context menu, only the columns appropriate for the objects current being displayed are listed. If the page is displayed by selecting **Tools=>Options=>GUI Options=>Columns**, all columns for all object types are displayed.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics















Customizing the List View

Customize Diff Options

The **Customize Diff** options pane allows you to customize the appearance of annotated diffs.

Use black background

Tab stop width:

State	Sym	Text Color	BG Color	Bold	Italic	Uline	Strike
For multi-window Diff viewers/editors:							
Added							
Deleted							
Changed							
Conflict							
Resolved Conflict							
For 2-way Diffs (single window):							
Line added	+		(none)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Line deleted	-		(none)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Changed to	>		(none)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Changed from	<		(none)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
For 3-way Diffs (single window):							
Added to B	+B	(none)		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Added to C	+C	(none)		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Added to both	+=	(none)		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Deleted from B	-B			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

The **Use black background** checkbox causes the diff view to appear as white text on a black background, instead of the default black text on a white background.

The customize diff page also contains a table that allows you to change background colors and font attributes in the diff views.

For multi-window Diff viewers/editors

These settings are used when the multiple window Diff Viewer or the Merge Conflict Editor are used. The Diff Viewer and Conflict Editor are invoked when **Use graphical**

diff tool is selected as the display option. The settings only affect the background color. Font attributes cannot be changed.

For 2-way Diffs (single window) and For 3 way Diffs (single window)

These settings are used when a single window diff format is used. The single window diff formats are Revised Diff (2-way only), Standard Diff, Unified Diff and Annotated Diff. Each line in the table contains the following settings:

- A **state** of a line, such as "line added."
- The **symbol** that indicates this state in the textual diff result, such as ">".

The remaining columns in the table can be edited by clicking on them or by selecting them and pressing **F2**.

- The **foreground color** to use for lines of that state.
- The **background color** to use for lines of that state.
- Checkboxes indicating that lines of that state should be displayed in **bold**, **italic**, **underlined**, or **strikethrough**.

Notes:

- If you select both underline and strikethrough, only underline displays.
- Font attributes, other than text color, are not applied in Revised Diff format.
- In a single window 3-way Diff, **A** represents the base file.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

Common Diff Operations

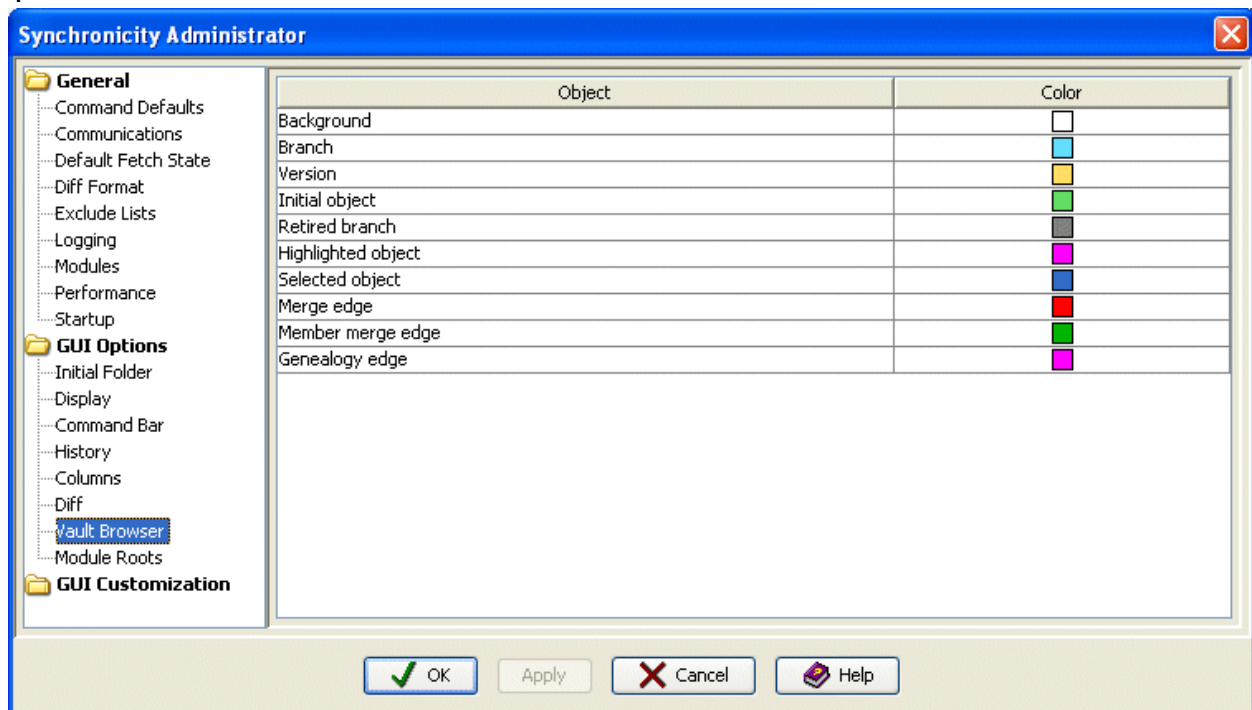
Advanced Diff Options

diff Command

Vault Browser

The **Vault Browser** options pane allows you to customize the appearance of the vault browser display.

T



Related Topics

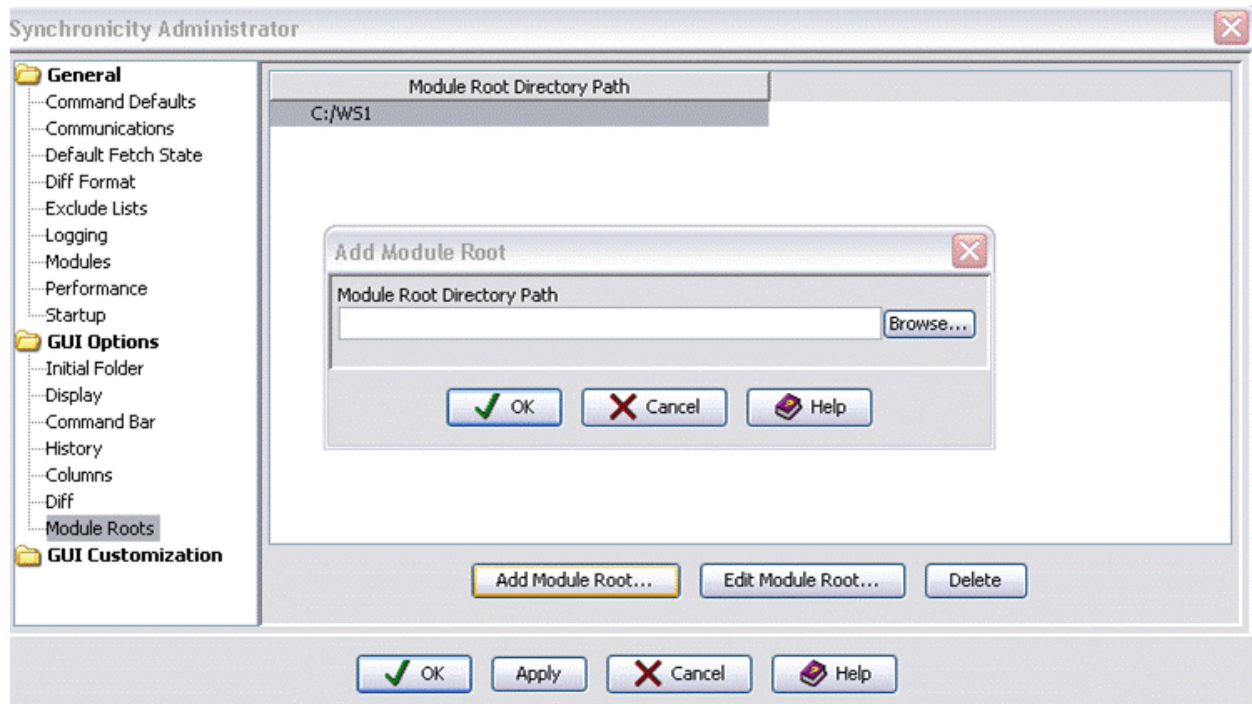
ENOVIA Synchronicity DesignSync Data Manager User's Guide: Vault Browser Overview

Module Roots

Using the **Module Roots** options pane of SyncAdmin, a user can create, modify, or delete entries from the list of workspace roots to include in the Modules view browser

within the DesignSync GUI interface. This panel is only visible when SyncAdmin is launched in User mode.

When data with a workspace root added to DesignSync (by creating a module, populating a module, setting a file-based workspace root, setting a vault for files-based data, or populating or checking out files-based data) a reference to the data is included automatically in the Module Root view.



Add a Module Root

To add the workspace root that is included in the Modules view from session to session, click Add Module Root. The Add Module Root pane is displayed.

Edit a Module Root

To edit the selected workspace root path, either by typing it directly or by browsing and selecting it from the file system, click Edit Module Root. The Edit Module Root pane is displayed.

Delete a Module Root

To remove a workspace root from the DesignSync GUI Modules view and from the list, select it from the list and click Delete. A confirmation box is displayed; click OK to delete the module root.

Making a Change to the Display of Module Roots

Once you have entered the change you want to make to the workspace root, click the appropriate button to direct DesignSync to take an action.

OK

When you click OK, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin closes.

Apply

When you click Apply, the changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the Apply button is disabled until changes are made in the pane.

Cancel

When you click Cancel, SyncAdmin exits. If you have not clicked Apply prior to clicking Cancel, any changes you have made are not saved.

Help

When you click the Help, the SyncAdmin help is displayed.

See Also

Added module roots (ModuleRoots)

Auto-creation of workspace root directory (AllowAutoRootCreation)

Workspace root path (DefaultAutoRootPath)

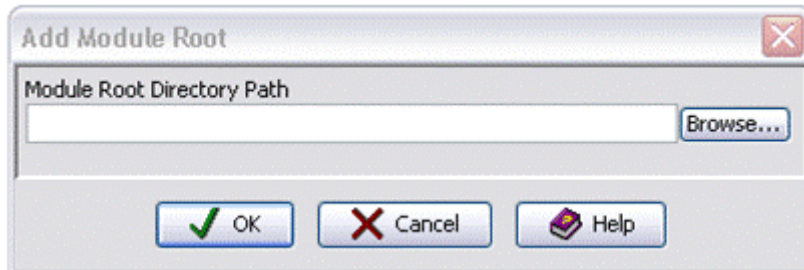
Adding a Module Root

The Module view is headed by the Module roots container. DesignSync adds a reference in the Modules view:

- Each time you select a module instance in the Tree view.
- When you select a client-side module base directory from the Bookmarks menu.
- When you enter a url in the Location field to navigate to a client-side module base directory.
- When you select a file-based vault within a workspace root directory,

Each reference remains in the Modules view until you log out of the GUI.

To include a reference to a module or file-based folder for display in the Modules view from session to session, click the icon in the Modules view and select **Modules => Add Initial Module Root**.



Module Root Directory Path

You can type the path name to the workspace root directory or click browse to locate and select the workspace root directory from the file system.

Browse

When you click Browse, the Select Module Roots Directory dialog box is displayed. Use this to navigate to the location of the workspace root you want to include in the Modules view.

OK

When you click OK, the workspace root directory path is displayed in the SyncAdmin window, and the workspace root is included in the Modules view from session to session.

Cancel

When you click Cancel, the Add Module Root dialog box closes and no changes are changes.

Help

When you click the Help, the help page is displayed.

Removing the Reference

To remove the reference to the module or file-based folder from those included in the Modules view the next time you log on, click the module instance icon or the file-based

reference and select **Modules => Remove Initial Module Root**. or from the Module Roots panel, select the reference and press **Delete**.

Related Topics

Added module roots (ModuleRoots)

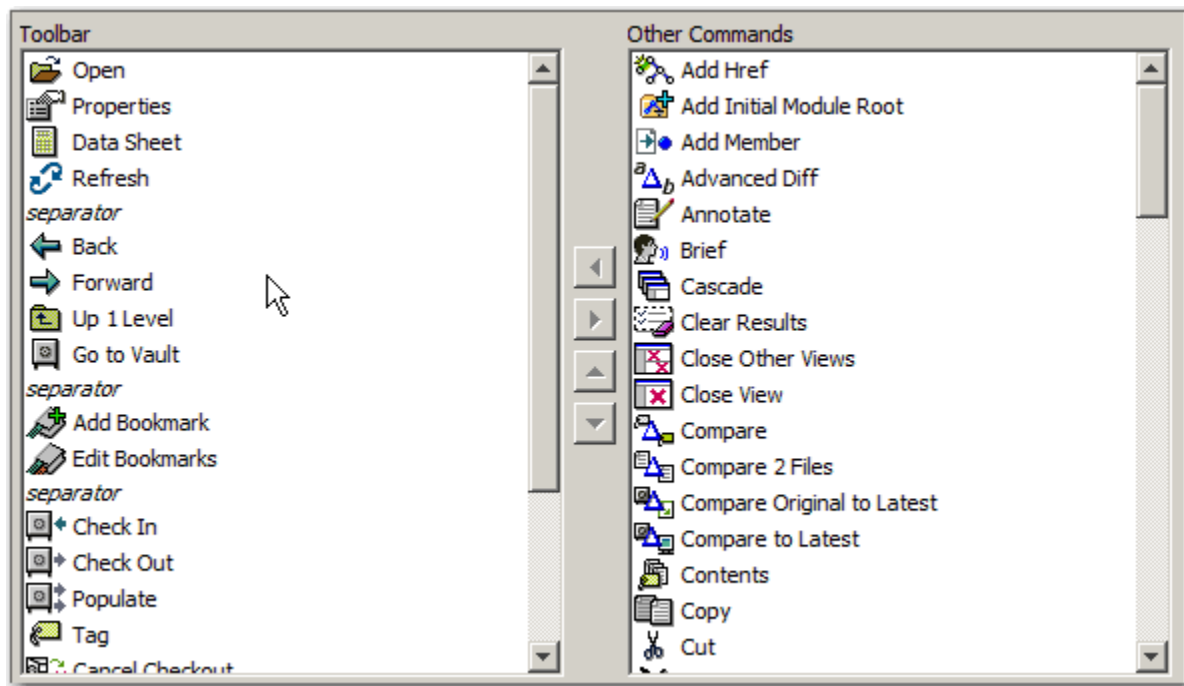
Auto-creation of workspace root directory (AllowAutoRootCreation)

Workspace root path (DefaultAutoRootPath)

GUI Customization

Main Toolbar

The **Main Toolbar** options pane allows you to choose which operations are displayed in the main toolbar. The Main Toolbar pane is only available when you select **Tools=>Options=>GUI Customization=>Main Toolbar** from the DesignSync GUI. (The toolbar is not available when you invoke a standalone SyncAdmin session from the command line or from the Start menu.)



The Main Toolbar pane contains two lists of operations. The list on the left shows the operations currently included in the main toolbar. The items are listed in the order in which they appear on the main toolbar. The list on the right shows operations that are not included in the main toolbar but can be added. These items are listed in alphabetical order.

All commands are available for both the main toolbar and the module toolbar. To customize the module toolbar, see GUI Customization: Module Toolbar.

You can customize the main toolbar in the following ways:

- To rearrange the main toolbar, click on an operation in the left list, and use the up and down arrow keys to move it.
- To add a button to the main toolbar, click on an operation in the right list and press the left arrow button. If you have selected an item in the left list, the new operation will appear directly below that item.
- To add a separator to the main toolbar, click on the **separator** entry located at the very bottom of the right list, and press the left arrow button.
- To remove a button from the main toolbar, click on an operation in the left list, and press the right arrow button.

Holding down the ALT key and pressing an arrow key has the same effect as pressing the corresponding arrow button.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

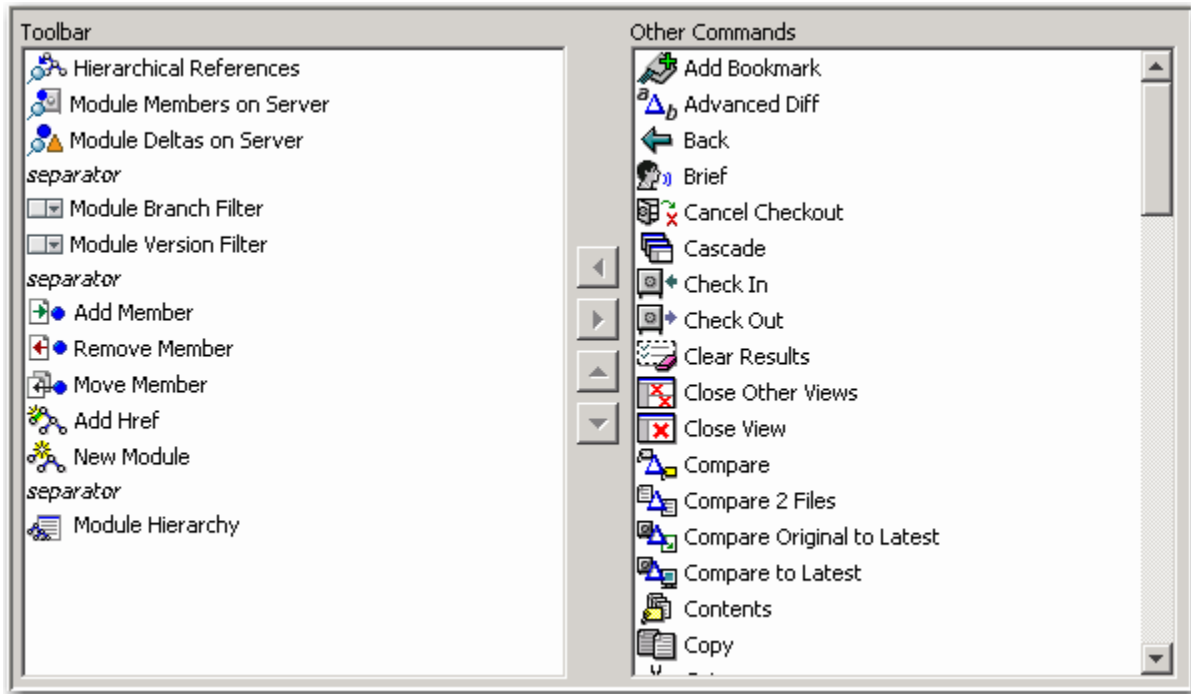
Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Module Toolbar

The **Module Toolbar** options pane allows you to choose which operations are displayed in the module toolbar. The Module Toolbar pane is only available when you select **Tools=>Options=>GUI Customization=>Module Toolbar** from the DesignSync

GUI. (The toolbar is not available when you invoke a standalone SyncAdmin session from the command line or from the Start menu.)



The Module Toolbar pane contains two lists of operations. The list on the left shows the operations currently included in the module toolbar. The items are listed in the order in which they appear on the module toolbar. The list on the right shows operations that are not included in the module toolbar but can be added. These items are listed in alphabetical order.

All commands are available for both the Module Toolbar and the Main Toolbar. To customize the main toolbar, see GUI Customization: Main Toolbar.

You can customize the module toolbar in the following ways:

- To rearrange the module toolbar, click on an operation in the left list, and use the up and down arrow keys to move it.
- To add a button to the module toolbar, click on an operation in the right list and press the left arrow button. If you have selected an item in the left list, the new operation will appear directly below that item.
- To add a separator to the module toolbar, click on the **separator** entry located at the very bottom of the right list, and press the left arrow button.
- To remove a button from the module toolbar, click on an operation in the left list, and press the right arrow button.

Holding down the ALT key and pressing an arrow key has the same effect as pressing the corresponding arrow button.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

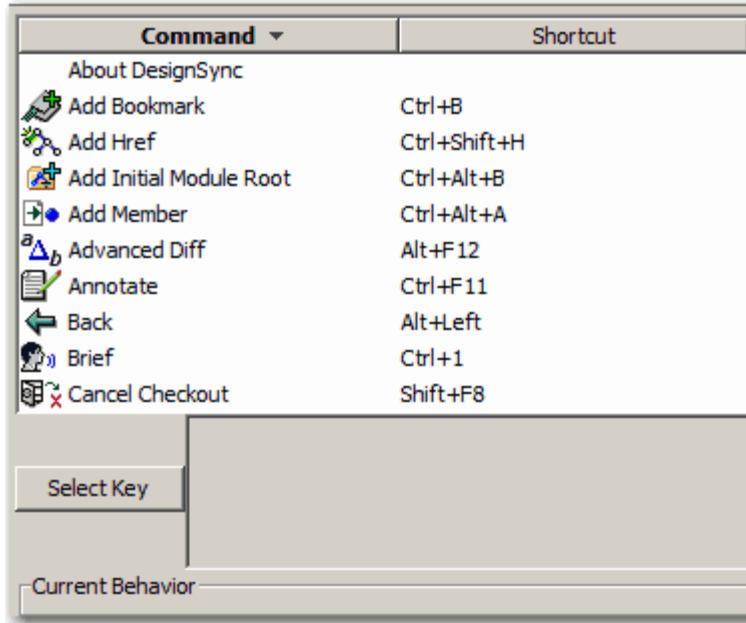
Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Keyboard Options

The **Keyboard Options** pane allows you to change the accelerator keys assigned to menu entries. In addition, you may assign keystrokes to bookmarks, and change the special keystroke operations.

Customize Keys contains a table listing all menu operations, bookmarks, and special keystroke operations, and they keystrokes currently assigned to each. You can sort by operation or by keystroke by clicking on the appropriate column header.



To assign a keystroke:

1. In the table, select the operation to which you want to assign a keystroke.
2. Click **Select Key**. The box next to the button turns blue to indicate that it is awaiting a keystroke. Press the key or key combination you want to assign. Note that some keys cannot be assigned, such as alphanumeric keys, or **Caps Lock**.
3. If the selected key is already attached to an operation, that operation is listed in the **Current Behavior** box.
4. Click **Set** to assign the key to the operation.

To remove a keystroke:

1. Select the operation from which you want to remove the assigned keystroke.
2. Click **Clear** to remove the assigned keystroke.

Notes

You can only assign one keystroke to each operation.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Custom Tools Options

The **Custom Tools** options pane allows you to add custom menu entries to the **Tools** menu. Once a tool has been added, you can also add it to the tool bar, or assign it an accelerator key.

The **Custom Tools** pane displays a list of tools that have been added. You can perform the following operations:

- To add a tool, click **Add Tool**. The Add Tool pane is displayed.
- To edit a tool, select it from the list and click **Edit Tool**. The Edit Tool pane is displayed. You can also edit a tool by double-clicking it in the list.
- To delete a tool, select it from the list and click **Delete Tool**. A confirmation box is displayed; click **OK** to delete the tool.
- To change the order of tools as displayed in the Tools menu, drag and drop the icons in the Customize Tools page.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Commands

The **Commands** pane allows you to select which commands are displayed in the DesignSync GUI. This is useful if the Administrator wants to control which commands are available to team members.

Make this SyncAdmin page available to users

Include the following commands in the DesignSync GUI:

<input checked="" type="checkbox"/> Delete	<input checked="" type="checkbox"/> Display Filters	<input checked="" type="checkbox"/> Go to Ancestor
<input checked="" type="checkbox"/> Go to Reference	<input checked="" type="checkbox"/> Add Member	<input checked="" type="checkbox"/> Go to Configuration
<input checked="" type="checkbox"/> Add Href	<input checked="" type="checkbox"/> Lock Branch	<input checked="" type="checkbox"/> Show Module Cache
<input checked="" type="checkbox"/> New Module	<input checked="" type="checkbox"/> Module Hierarchy	<input checked="" type="checkbox"/> Move Member
<input checked="" type="checkbox"/> Remove Member	<input checked="" type="checkbox"/> Set Root Folder	<input checked="" type="checkbox"/> Module Status
<input checked="" type="checkbox"/> Make Branch	<input checked="" type="checkbox"/> ProjectSync	<input checked="" type="checkbox"/> Retire
<input checked="" type="checkbox"/> Set Persistent Populate Filters	<input checked="" type="checkbox"/> Tag	<input checked="" type="checkbox"/> Unlock
<input checked="" type="checkbox"/> Workspace Wizard		

Check the box next to any command you want displayed in the DesignSync GUI. Uncheck the box to hide the command.

You can also allow team members to customize their DesignSync GUI by checking the **Make this SyncAdmin page available to users** box.

Module Commands

These commands pertain to module data: Go to Reference, Add Href, New Module, Remove Member, Set Persistent Populate Filters, Display Filters, Add Member, Lock Branch, Module Hierarchy, Set Root Folder, Go to Go to Ancestor, Configuration, Show Module Cache, Move Member, and Module Status.

If your project team does not currently work with modules, de-selecting all of the module related commands will remove from the DesignSync GUI actions that operate on module data. Users will still be able to view modules.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

States Options

An administrator can control the available check-in and check-out states for a project or site from the **States** options pane. You can allow team members to customize their DesignSync GUI by checking the **Make this SyncAdmin page available to users** box.

Make this SyncAdmin page available to users

State	Check In/Cancel	Check Out/Populate
Unlocked copies	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Locked copies	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
References to versions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Links to cache	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Links to mirror	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Locked references		<input checked="" type="checkbox"/>

Check In/Cancel States

By default, DesignSync forms offer users five check-in modes: Unlocked copies, locked copies, references, links to cache, and links to mirror. You can limit the modes available to all users on your LAN by deselecting any of the displayed **Check In States**.

You must select at least one **Check In State**.

Note: In the DesignSync DFII integration, the **Cancel Checkouts => Results** form and the Cancel Options section of the **Show Checkouts => Results** form also obey the allowed check-in states.

Check Out/Populate States

By default, DesignSync forms offer users all of the displayed check-out modes. You can limit the modes available to all users on your LAN by deselecting any of the **Check Out/Populate States**.

You must select at least one **Check Out State**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can

continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

See the following topics in DesignSync Data Manager User's Guide for more information:

Object States

Checking in Design Files

Checking Out Design Files

Populating Your Work Area

Command Options

An administrator can customize the options displayed for DesignSync **Revision Control** and **Modules** commands.

Make this SyncAdmin page available to users

Add Member Options		Cancel Checkout Options	
emptydir	Allowed ▼	exclude	Allowed ▼
filter	Allowed ▼	force	Allowed ▼
modulecontext	Allowed ▼	retain	Advanced ▼
recurse	Allowed ▼	trigarg	Advanced ▼
Check In Options		Check Out/Populate Options	
branch	Advanced ▼	cachemode	Advanced ▼
dryrun	Allowed ▼	cachepath	Advanced ▼
exclude	Allowed ▼	emptydirs	Advanced ▼
force	Allowed ▼	exclude	Allowed ▼
keys	Advanced ▼	force	Allowed ▼
now	Allowed ▼	brofilter	Advanced ▼

There are three choices for each available option associated with a revision control action:

- **Allowed** - provides the command option to the user on the revision control dialog.
- **Advanced** - places the command option in the expanded Advanced portion of the revision control dialog. The user will see the command when they click Advanced in the revision control dialog.
- **Hidden** - the command option is not available to the user.

Note: When setting site-wide settings for SyncAdmin, you can allow team members to customize their DesignSync GUI by checking the **Make this SyncAdmin page available to users** box.

Note: In some cases, options work together. For example, the **-keep** and **-branch** options of **retire** are displayed together in a box on the dialog, so it is not possible to move one to the advanced section without moving the other. The same is true for the **-version** and **-overlay** options on the co/populate dialogs.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

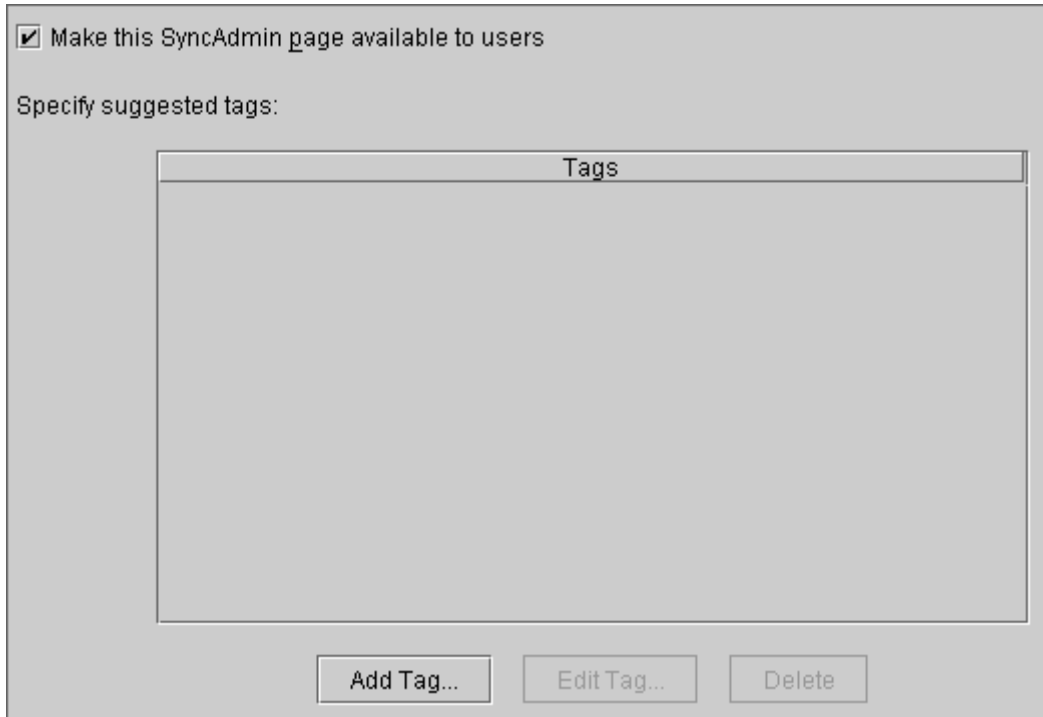
Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Tag Options

An administrator can add specific tag name choices to the drop-down menu in the DesignSync Tag dialog by defining them in the SyncAdmin **Tags** options pane.

You can also allow team members to customize their DesignSync GUI by checking the **Make this SyncAdmin page available to users** box.



- To add a tag, click **Add Tag** and enter the name in the **Add Tag** pane.
- To edit an existing tag name, highlight the trigger in the window and click **Edit Tag**. Modify the tag name in the **Edit Tag** pane.
- To delete a tag name, highlight the tag name in the window and click **Delete**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

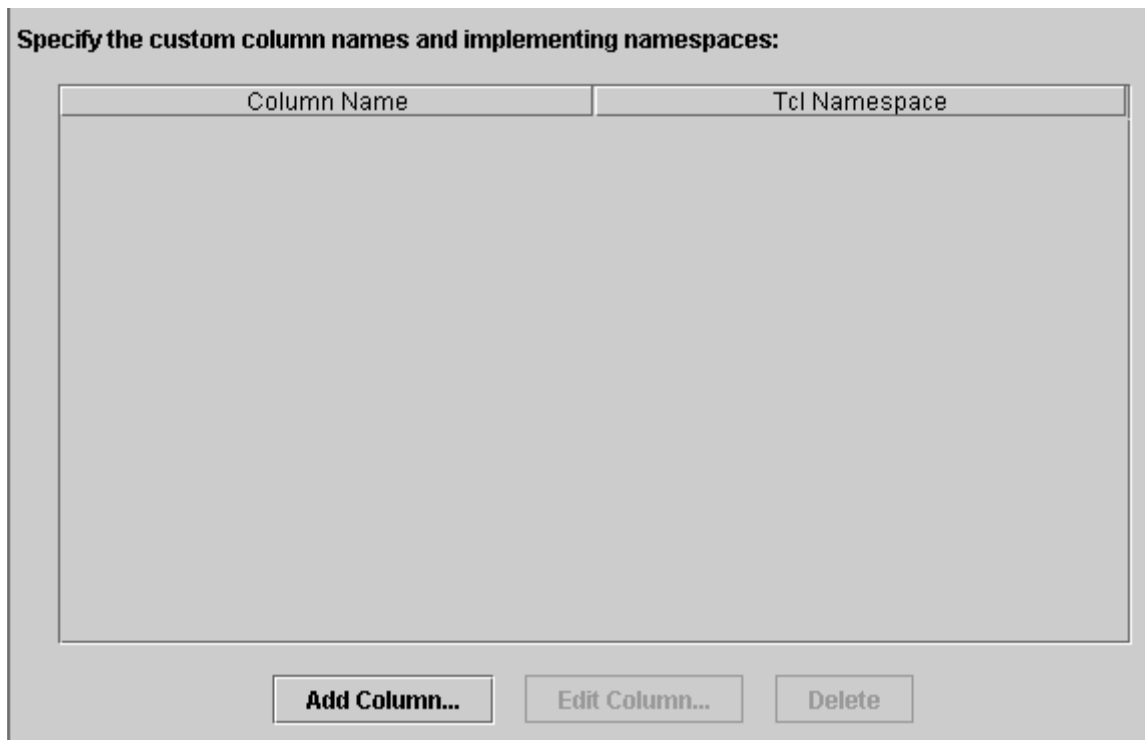
Related Topics

DesignSync Data Manager User's Guide: Tagging Versions and Branches

Custom Columns

Custom Column Options

An administrator or project leader can specify custom column names for display in the List View. For example, the project leader can add a new list view column called "Approved" which indicates whether that file has been approved for inclusion.



- To add a new Column, click **Add Column**. The Add Column pane is displayed.
- To edit a Column, select it from the list and click **Edit Column**. The Edit Column pane is displayed. You can also edit a column by double-clicking it in the list.
- To delete a column, select it from the list and click **Delete**. A confirmation box is displayed; click **OK**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

Customizing Columns

Custom Column Options

An administrator or project leader can specify custom column names for display in the List View. For example, the project leader can add a new list view column called "Approved" which indicates whether that file has been approved for inclusion.

Specify the custom column names and implementing namespaces:

Column Name	Tcl Namespace
-------------	---------------

- To add a new Column, click **Add Column**. The Add Column pane is displayed.
- To edit a Column, select it from the list and click **Edit Column**. The Edit Column pane is displayed. You can also edit a column by double-clicking it in the list.
- To delete a column, select it from the list and click **Delete**. A confirmation box is displayed; click **OK**.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

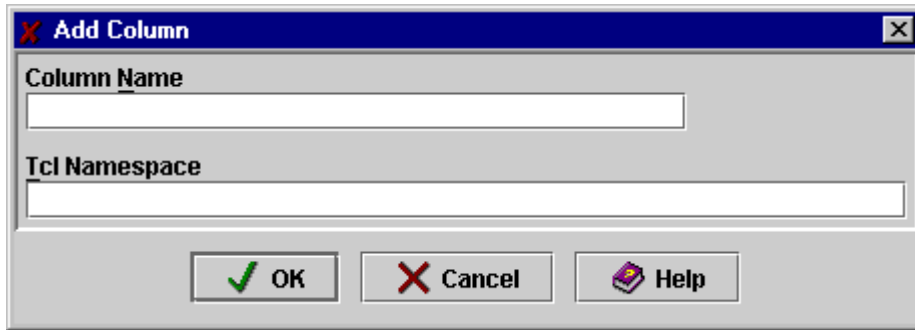
When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

Customizing Columns

Add or Edit a Custom Column

To create a new column after clicking **Add Column**:



1. Enter a name in the **Column Name** field.
2. Enter a namespace in the **Tcl Namespace** field. The referenced namespace should contain the command you want to implement for the custom column. Click here for more information.

To edit an existing column definition, in **Custom Columns Options** click **Edit Column** and edit the fields on the **Edit Column** pane.

OK

When you click on the **OK** button, any changes you have made are written to the registry file that is being read by SyncAdmin. After saving any changes, SyncAdmin exits.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topic

Creating a Tcl Namespace

Creating a Tcl Namespace

1. Create the Tcl file containing the namespace

The Tcl file must define the specified namespace (using `namespace eval`). This file must be loaded by the GUI, either through the startup script, `tclIndex`, or some other method.

2. Implement the column process

DesignSync Data Manager Administrator's Guide

There are 2 processes that can be implemented to define the contents of a custom column, only the first of which is required. The process must be defined within the specified namespace. Both processes take a single argument: the url of the object being displayed.

GetString

This is the only required process. The return value of this process is displayed in the custom column. If the string is bounded by `<html>` and `</html>`, simple html tags may be used (this is automatically provided by Java).

GetIcon

This process is optional. It specifies the name of the icon to appear in the column. The icon may be:

- The name of an icon contained in the DesSync standard jar file (for example, `ok.gif`)
- The full path to an icon in a Java-supported format
- A url to an icon in a Java-supported format.

An empty string indicates no icon.

Example

The following code implements a custom column that displays the value of a custom property called **approve**. This property can have the value **OK**, or a message indicating why approval has not been granted; or the property can not exist (indicated by an empty string). If the property does exist, either a check mark or red X icon is displayed in the column.

```
namespace eval WOApprove {}
proc WOApprove::GetString {url} {
    set ret ""
    catch {set ret [url getprop $url approve]}
    set ret
}
proc WOApprove::GetIcon {url} {
    set approval [url getprop $url approve]
    if {[string length $approval] == 0} {
        return ""
    } elseif {[string compare $approval OK] == 0} {
        return "ok.gif"
    }
    return "canc.gif"
}
```

Diagnostics

Diagnostics Overview

The DesignSync graphical client (DesSync/DesignSync GUI) provides a set of diagnostic tools for troubleshooting. Within the interface, **Tools=>Diagnostics** brings up the **DesignSync Diagnostics** pane. Three tabs are available:

- **Environment Variables** — Displays DesignSync-related environment variables and their current settings. The SyncAdmin tool also has an Environment Variables tab that displays the same list of environment variables.
- **Installation Settings** — Displays settings that were selected during the DesignSync installation or were later overridden using the SyncAdmin tool.
- **Java Properties** — Displays the current Java classes, paths, and versions.

The **DesignSync Diagnostics** dialog box is useful for debugging your DesignSync environment. For example, a DesignSync representative may ask you for information that is available from this dialog box when assisting you with a problem.

All of the information displayed is read-only -- you cannot edit the values from the **DesignSync Diagnostics** dialog box.

Note:The Java Properties and Environment Variables diagnostic information is also available from the Diagnostics menu in SyncAdmin.

Related Topics

[syncinfo Command](#)

[Using Environment Variables](#)

Environment Variables

The **Environment Variables** pane is available from DesignSync **Tools=>Diagnostics=>Environment Variables** and from SyncAdmin **Diagnostics=>Environment Variables**. The **Environment Variables** pane displays the current settings of the DesignSync-related environment variables for your machine.

Although the information in this window is read-only, it can be useful for debugging your DesignSync environment. For example, Customer Support may ask you for information that is available from this pane when assisting you with a problem.

This pane is available to DesignSync users as well as LAN administrators using the SyncAdmin tool.

Name	Value
CDS_INST_DIR	<<not set>>
CDS_SITE	<<not set>>
EDITOR	<<not set>>
HOME	/home/tjames
HOSTNAME	guaraldi
LD_LIBRARY_PATH	/home/tjames/syncinc/jre.sol2/lib/sparc/client/ho...
MGC_LOCATION_MAP	<<not set>>
PATH	/home/tjames/syncinc/bin:/bin:/usr/bin:/usr/sbin/...
SYNC_ARCH	sol2
SYNC_AUTOMOUNT	/tmp_mnt
SYNC_CACHE	<<not set>>
SYNC_CLIENT_SCRIPT	/home/tjames/syncinc/syncinc.custom
SYNC_CLIENT_VAULT	<<not set>>
SYNC_CUSTOM_DIR	/home/tjames/syncinc/custom
SYNC_DAEMON_PREW...	<<not set>>
SYNC_DAEMON_TIMEO...	<<not set>>
SYNC_DAEMON_WAIT	<<not set>>
SYNC_DEBUG_INTERR...	<<not set>>
SYNC_DIR	/home/tjames/syncinc
SYNC_ENT_CUSTOM	/home/tjames/syncinc/custom/enterprise
SYNC_EXTRA_LD_PATH	<<not set>>
SYNC_NO_GDM	<<not set>>

Name

Lists the names of all the DesignSync-related environment variables, regardless of whether a value has been set for them.

Value

Displays the value that has been set for each environment variable. If a value has not yet been defined for a particular environment variable, the value will read <<not set>>.

OK

Click OK to close the **Diagnostics** pane.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Configuring a Multi-User Environment](#)

[Using Environment Variables](#)

Installation Settings

The **Environment Variables** pane is available from DesignSync **Tools=>Diagnostics=>Installation Settings**. The **Installation Settings** pane displays the settings that were selected during the installation of DesignSync software or were later overridden using the SyncAdmin tool.

Environment Variables		Installation Settings	Java Properties
Name	Value		
Client Data Vault Directory	/home/tjames/syncdata/client_vault		
Default Cache Directory	/home/tjames/sync_cache		
Enterprise Registry File	<<not in use>>		
File Editor	emacs		
HTML Browser	nsremote		
Home Directory	/home/tjames		
Project Registry File	<<not in use>>		
Site Registry File	/home/tjames/syncinc/custom/site/config/SiteRe...		
Synchronicity Registry File	/home/tjames/syncinc/share/SyncRegistry.reg		
User Registry File	/home/tjames/.synchronicity/UserRegistry.reg		
opSeq	31 (0) java=0		

Although the information in this window is read-only, it can be useful for debugging your DesignSync environment. For example, Customer Support may ask you for information that is available from this pane when assisting you with a problem.

This pane is available to DesignSync users as well as LAN administrators using the SyncAdmin tool.

Related Topics

ENOVIA Synchronicity Command Reference: syncinfo Command

Using Environment Variables

Java Properties

The **Environment Variables** pane is available from DesignSync **Tools=>Diagnostics=>Java Properties** and from SyncAdmin **Diagnostics=>Java Properties**. The **Java Properties** pane displays the current Java classes, paths, and versions.

DesignSync Data Manager Administrator's Guide

Name ▼	Value
java.class.path	/home/tjames/syncinc/jre.sol2/./classes/dsj.jar
java.class.version	48.0
java.home	/home/tjames/syncinc/jre.sol2
java.vendor	Sun Microsystems Inc.
java.vendor.url	http://java.sun.com/
java.version	1.4.0_01
os.arch	sparc
os.name	SunOS
os.version	5.7
user.name	tjames

Although the information in this window is read-only, it can be useful for debugging your DesignSync environment. For example, Customer Support may ask you for information that is available from this pane when assisting you with a problem.

This pane is available to DesignSync users as well as LAN administrators using the SyncAdmin tool.

Using the DesignSync Web UI

Overview of the DesignSync Web UI




The DesignSync Web User Interface (Web UI) allows you to perform some system management and configuration operations from your Web browser, without launching an application specific client, such as the SyncAdmin or the DesignSync clients, or a command-line interface, such as the dssc client.



IMPORTANT: If you use Internet Explorer as your browser, you must make sure that Compatibility Mode is OFF. The Web UI cannot not function properly when compatibility mode for Internet Explorer is ON.

Note: If the company has a privacy policy in place for use with the DesignSync server, all users are automatically prompted to view and accept the policy either during their initial login, or, if a policy is changed or added while the users are logged on, during their session. For more information, see Logging into the DesignSync Web UI.

- **DesignSync Menu-** From the DesignSync menu, you can view object Data sheets create and manage Enterprise Developments, create and manage mirrors, and create and manage data replication.
- **ProjectSync Menu** - From the ProjectSync menu, you can view object Data sheets, create and manage note types, create and manage note queries, and create and manage projects.
- **Admin Menu** - From the Admin menu, you can create and manage User profiles, create and manage triggers, change administrative settings on the server, reset the server, reset the access controls, or view the Enterprise Development activity queue.

The DesignSync Web UI button bar at the top of the screen features quick links:

- Quick View - Allows you to quickly view and modify the contents of specified notes.
- Me- () - quick link to edit the profile information for the logged in user, switch user, and log out.
- Quick Add () - quick links to all Create/Add Content options, for example: adding a new mirror, adding a new enterprise development, or adding a new note type.
- Share () - Launches the default email client to send the direct link to the page being viewed. The subject of email is "Notification" and the link appears in the body of the email.

- Home () - Returns to the home splash screen. which displays the DesignSync version information.
- Help () - Provides links to User Assistance topics, including this help.

Logging into the DesignSync Web UI

In order to take advantage of the functionality of the DesignSync Web UI, you must log in when you connect to the DesignSync Web UI URL.

If you are using 3DPassport and have already logged into a DesignSync client, all the DesignSync clients, including the DesignSync Web UI recognize your login and you do not need to log in again.


If your server has a privacy policy, or the policy has changed, you must accept the privacy policy in order to perform actions using the DesignSync Web UI, even if you were already logged into the other DesignSync clients.

Viewing the Privacy Policy

In compliance with the European Union General Data Protection Regulation, all users have the right to easily access the privacy policy information for any website that stores personal information. When users log in they can review and accept the privacy policy.

To view the privacy policy, click the View Privacy Policy page from the User Profiles page available from the Me Menu.

Me Menu

The Me Menu () provides user-account based DesignSync Functionality.

- Edit Profile - allows you to update the user profile for the currently logged user. For more information on editing user profiles, see Editing and Deleting User Profiles.
- Switch User - launches the user login screen allowing you to logout of the current session and log in again under a different user name.
- Log out- Logs out of the session. This directs to the log out page. The log out page contains a reminder that if you want to continue working with the DesignSync server, you must log in again.

Add Menu

The Add Menu provides a quick way to create any new objects or configurations that can be created by the DesignSync Web UI for managing either DesignSync or ProjectSync objects.

- **Developments:** Quick links to create basic Enterprise Development components. For more information on using Enterprise Developments, see the *Enterprise DesignSync Administrator User's Guide*.
 - **Add Development** - define the properties specific to the development and associate the development with the appropriate sub-components.
 - **Add Tool Suite** - define the tool suites available to developments. These tool suites can be used by multiple developments. For more information on tool suites, including creating a tool suite that is only available to a single, specified development, see the *Enterprise DesignSync Administrator User's Guide*.
- **Mirrors:** Quick links to Add/Create a mirror.
 - **Add Mirror** - define the properties for creating a mirror.
- **Replicate:** Quick links to create basic replication components.
 - **Add Root** - define the properties for a new replicate root (DRR) on a MAS.
 - **Add Data** - define the properties for a new data replication on a DDR.
- **Note Types:** Quick link to create a new note.
 - **Add Note** - define the properties for a new note.
- **Projects** - Quick links to creating ProjectSync projects, which are file-based objects that use issue tracking within ProjectSync.
 - **Add Project** - define the properties for a new project tracked within ProjectSync.
 - **Add Configuration** - define the properties for a configuration, which is the ProjectSync equivalent of a role to which users who perform that role-function can be assigned and granted access to a project.
- **User Profile:** Quick links to creating a new user profile.
 - **Creating User Profiles** - define the properties for a new user profile.
- **Triggers:** Quick link to create a note-object trigger.
 - **Add Trigger** - define the properties a server-side (note-object) trigger, such as a note activity trigger, or activate the email trigger.

System Administration of DesignSync Client

Configuring Client Default Settings

Running SyncAdmin in site mode after installation allows you to configure the DesignSync client default preferences for all users of the DesignSync installation. For example, you can specify which Web browser to use when displaying Data Sheets. Some of the client configuration values can be set during the client installation.

You can modify the default values by using the SyncAdmin tool.

Related Topics

[ENOVIA Synchronicity Administrator Tool Overview](#)

[Site-Wide Configuration](#)

Configuring the Environment

Overview of Configuration Approaches

DesignSync provides the SyncAdmin tool to help you configure the DesignSync environment for an entire site or a project team. Using SyncAdmin, you specify default preferences. SyncAdmin updates the appropriate registry files, thus ensuring that all users have the settings you selected for your site or project team.

The following topics describe the different ways in which you can configure your multi-user environment:

- [Synchronicity Administrator \(SyncAdmin\) Tool Overview](#)
- [Using Environment Variables](#)
- [Setting Command Line Defaults](#)
- [Site-Wide Configurations](#)
- [Project-Specific Configurations](#)
- [Overview of Registry Files](#)
- [Autoloading stcl Procedures](#)
- [Managing Revision Control of Files Between Windows and UNIX](#)

Using Environment Variables

DesignSync supports a number of environment variables to help you configure your site and user installations. You can use standard environment variables like HOME and USER in startup scripts as well as in the SyncAdmin graphical interface forms to set your client vault, browser, and editor.

The table below describes the DesignSync-specific environment variables.

To set these or other standard environment variables for all users of your UNIX installation, you can add them to `<SYNC_DIR>/syncinc.custom`. If this shell script exists, the `<SYNC_DIR>/syncinc` wrapper script sources it automatically. The wrapper script uses any variables set in `<SYNC_DIR>/syncinc.custom` instead of the DesignSync defaults. You can place your shell script in a different location by using the `SYNC_CLIENT_SCRIPT` environment variable as described in the table below.

To view the values of environment variables from within the DesignSync environment, you can use the **Tools=>Diagnostics** dialog. The **syncinfo** command provides information about many of the environment variables as well. In addition, the **Environment Variables** tab of the **SyncAdmin** tool lets you view the environment variable settings.

Note: When using `dss` or `stcl`, you must restart `syncd` for environment changes to take effect. The `dssc` and `stclc` shells do not use `syncd` and are generally the recommended DesignSync shells.

See the **Environment Variables for Tcl Scripts** topic in the ProjectSync help system for the set of environment variables you can use in `stcl` scripts.

<code>ProxyNamePort</code>	Specifies a proxy name and port for use with DesignSync. Use this format to specify a proxy: <code><ProxyName>:<ProxyPortNumber></code>
<code>SYNC_AUTOMOUNT</code>	Specifies the prefix used for automounts on your installation's network. DesignSync products strip the default automount prefix, <code>/tmp_mnt</code> , from paths. If your network uses a different prefix, specify that prefix using the <code>SYNC_AUTOMOUNT</code> environment variable.
<code>SYNC_CLIENT_SCRIPT</code>	Path to a script used by a system administrator to configure all client installations at a site. The installation scripts automatically source the script pointed to by <code>SYNC_CLIENT_SCRIPT</code> . By default, <code>SYNC_CLIENT_SCRIPT</code> is set to <code><SYNC_DIR>/syncinc.custom</code> . <code>SYNC_CLIENT_SCRIPT</code> must be set in each user's environment.
<code>SYNC_CUSTOM_DIR</code>	By default, DesignSync supports a location for custom scripts and files: <code><SYNC_DIR>/custom</code> . To place these custom files elsewhere, use the <code>SYNC_CUSTOM_DIR</code> environment variable. If a site or project uses <code>SYNC_CUSTOM_DIR</code> , the system administrator or project leader must set this environment variable before installing

	<p>the DesignSync software. Likewise, team members must set this environment variable before using the DesignSync software. The <code>SYNC_CUSTOM_DIR</code> environment variable is supported on Windows as well as UNIX.</p>
<code>SYNC_DAEMON_TIMEOUT</code>	<p>Determines after how many minutes of inactivity the syncd process times out. You can set <code>SYNC_DAEMON_TIMEOUT</code> to any number of minutes. By default <code>SYNC_DAEMON_TIMEOUT</code> is set to 180; syncd times out 180 minutes after the last dss/stcl process communicating with it exits. The syncd process never times out while a dss or stcl session is active or if you have applied a lock to the process using the syncdadmin command.</p> <p>On Windows systems, syncd never times out. However, you can stop syncd from the Windows Start menu (typically Start-> Programs-> Dassault Systemes ProductVersion -> Stop SyncDaemon).</p> <p>Note: Setting the <code>SYNC_DAEMON_TIMEOUT</code> does not take effect until syncd is restarted.</p>
<code>SYNC_DIR</code>	<p>Path to your DesignSync installation directory; on UNIX you need to set this environment variable before running the DesignSync software. You do not have to set this environment variable before installing and configuring the DesignSync software. UNIX users can include this variable in their <code>.cshrc</code> files.</p>
<code>SYNC_HOSTNAME</code>	<p>An alias by which the server's host machine is known. Using this variable allows you to make the <code>SYNC_HOSTNAME</code> name independent of the actual host on which the server runs allowing for automatic system fail-over and easy cut-over to a different host machine.</p>
<code>SYNC_ENT_CUSTOM</code>	<p>Path to a directory for enterprise-wide scripts and files: by default, <code>SYNC_ENT_CUSTOM</code> points to <code><SYNC_CUSTOM_DIR>/enterprise</code>. Note: You use the <code>SYNC_CUSTOM_DIR</code> environment variable to redirect the entire custom hierarchy.</p>
<code>SYNC_EXTRA_LD_PATH</code>	<p>Use the <code>SYNC_EXTRA_LD_PATH</code> environment variable to customize the <code>LD_LIBRARY_PATH</code> environment variable setting. For security reasons, the <code>SYNC_EXTRA_LD_PATH</code> environment variable cannot be used in installations that use SUID mode.</p> <p>By default, the DesignSync software removes the existing DesignSync paths in the <code>LD_LIBRARY_PATH</code> environment variable, replacing the paths with those from the current</p>

	installation. The DesignSync software appends the contents of SYNC_EXTRA_LD_PATH to the beginning of LD_LIBRARY_PATH. If you want to use a different library path, set that path in SYNC_EXTRA_LD_PATH.
SYNC_FLEXMLMRC	<p>Enables or disables the creation of the .flexlmrc file by FlexNet license management software. By default, DesignSync sets this variable to SYNC_FLEXMLMRC 0, which disables the creation of the .flexlmrc file. To enable the file's creation, use SYNC_FLEXMLMRC 1. (For more information, see Setting Up Licensing for DesignSync Products.)</p> <p>Note: If the .flexlmrc file already exists, it cannot be removed by setting this variable. You must delete it manually.</p>
SYNC_PROJECT_CFGDIR	Path to project configuration files. Project leaders can create a project-specific registry and scripts to ensure a consistent configuration for team members. Note: If you change the value of SYNC_PROJECT_CFGDIR, you must restart syncd to point to the new project.
SYNC_SITE_CNFG_DIR	Path to a directory for site configuration scripts and registry files: by default, SYNC_SITE_CNFG_DIR resolves to <SYNC_SITE_CUSTOM>/config which, in turn, resolves to <SYNC_CUSTOM_DIR>/site/config.
SYNC_SITE_CUSTOM	Path to a directory for site-specific scripts and files: by default, SYNC_SITE_CUSTOM points to <SYNC_CUSTOM_DIR>/site. Note: You use the SYNC_CUSTOM_DIR environment variable to redirect the entire custom hierarchy.
SYNC_TRACE	Turns on tracing. Use the following format: SYNC_TRACE 0. You must restart the client or server to enable SYNC_TRACE. (For more information, see Running a DesignSync Client in Debug Mode.)
SYNC_USER_CFGDIR	Path to a directory for user configuration scripts and registry files: by default, SYNC_USER_CFGDIR points to <HOME>/synchronicity. for Unix and %AppData%\Synchronicity on Windows.
SYNC_USER_CONFIG	Path to an optional user configuration file; DesignSync sources this file upon startup if it exists; by default, SYNC_USER_CONFIG points to <SYNC_USER_CFGDIR>/user.cfg.

Related Topics

Diagnostics Overview

syncinfo Command

Setting Command Line Defaults

Users can save default values for commands, by using the Command Line Defaults System. Project leaders and site administrators can save command line default values for a project team, or for all users at a site. See the `sregistry scope` command documentation for details.

Site-Wide Configuration

As a system administrator, you can configure all DesignSync clients and SyncServers in your installation at once by making changes to the site registry. When you install DesignSync software, you answer questions that define many installation-wide setup defaults. As you answer these questions, information is written to the site registry file on UNIX or the Windows registry. Using the SyncAdmin tool, you can make changes to the registry without the risk of file corruption. Any changes that you make using SyncAdmin replace the site-wide settings for all DesignSync clients and SyncServers on your LAN. Some LAN settings are only available on UNIX.

As a UNIX LAN administrator, you must have permission to write to the site-wide registry file, `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg` to make LAN-wide changes with SyncAdmin. (`<SYNC_SITE_CNFG_DIR>` defaults to `<SYNC_CUSTOM_DIR>/site/config`). Without write permission to this file, the choices available in the SyncAdmin tool are limited to those that affect projects or individual DesignSync clients (users).

You can configure your site by

- Setting environment variables for all users of your UNIX installation and storing them in the `<SYNC_DIR>/syncinc.custom` shell script.
- Setting up site-wide data management preferences, such as the default fetch state of objects and whether to compress the data.
- Developing stcl scripts and using the stcl autoload mechanism so that your users can call the stcl procedures without the overhead of setup steps.
- Developing triggers that automatically execute stcl or dss scripts for all users at your site.
- Setting up revision control notes for your site, so that you can specify the consequences to follow particular revision control actions such as checkin and checkout.

- Setting the location of all your users' log files to the same relative location, for example, <USER>/logs.
- Defining various projects for your site so that project leaders can manage and further customize the preferences for their projects.

Related Topics

Overview of Registry Files

Using Environment Variables

ENOVIA Synchronicity Administrator Tool Overview

Autoloading Tcl Procedures

Project-Specific Configuration

As a project leader, you can set preferences for a project in a project-level registry file. Use a project registry to:

- Override any of the site configuration registry settings
- Associate a file cache with a project registry and thus restrict its use to those members who have access to the project registry
- Set up stcl procedures specific to a particular project using the autoloading mechanism

To do so, you

1. Create a project directory for each project team and a `ProjectRegistry.reg` file within that directory.
2. Set your `SYNC_PROJECT_CFGDIR` environment variable to the project directory.
3. Create the cache directories you intend to specify for each project with appropriate permissions.

You can also let DesignSync create the cache directories automatically, but in that case, the permissions are open to the world. See [Setting Permissions on the Cache](#) for more information.

4. Customize the `ProjectRegistry.reg` file using the **Projects** tab of **SyncAdmin**. You can use this tab to define each project's name, vault URL, and cache directory. See [SyncAdmin Help](#) for details.
5. Instruct project team members to set the `SYNC_PROJECT_CFGDIR` environment variable to the appropriate project directory so that their DesignSync clients can read the project registry.

Note: If project team members change the value of `SYNC_PROJECT_CFGDIR`, they must restart `syncd` to point to the new project.

Related Topics

Overview of Registry Files

ENOVIA Synchronicity Administrator Tool Overview

DesignSync Data Management User's Guide: What Is a Project?

Setting Permissions on the Cache

Tips for Administering LAN Caches

Autoloading Tcl Procedures

Overview of Registry Files

DesignSync configuration information is stored in registry files. The DesignSync tools automatically save configuration settings in the appropriate registry file:

- DesignSync Web UI's Administer Server panel writes to the server's port registry file.
- The SyncAdmin (DesignSync Administrator) tool can write to the site-wide registry file, a project registry file, or any specified registry file.
- When invoked via the DesignSync GUI's **Tools=>Options** menu, SyncAdmin writes to the user's registry file.

To programmatically read, write or delete registry values, use the `sregistry` command set.

Less commonly used registry settings do not have a SyncAdmin interface. Those can be found in the Registry Settings topics in this book. The Registry Settings topics cover all available registry settings, noting whether the registry key has a SyncAdmin interface.

Client Registry Files

The registry files that contain configuration information for DesignSync clients are listed below in the order of precedence.

- `<SYNC_USER_CFGDIR>/UserRegistry.reg`

The `UserRegistry.reg` file contains a user's personal preference settings. These settings can be changed using the **Tools=>Options** menu in the DesignSync graphical user interface. `SYNC_USER_CFGDIR` defaults to `<HOME>/synchronicity` on Unix, and `%AppData%\Synchronicity` on Windows.

- `<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`
 - The `ProjectRegistry.reg` file lets a project leader set preferences for a project. To do so, the project leader creates a project directory and a `ProjectRegistry.reg` file within that directory. The project leader sets the `SYNC_PROJECT_CFGDIR` environment variable to the project directory and then customizes the `ProjectRegistry.reg` file using SyncAdmin. Project team members must set the `SYNC_PROJECT_CFGDIR` environment variable to the project directory so that their DesignSync clients can read the project registry. For more information, see the Project Specific Configurations topic.
 - `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`
1. The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. Installing the software generates a default `SiteRegistry.reg` file. This file contains the default settings that are inherited by every user. By using the SyncAdmin tool, a DesignSync administrator can override some of these default settings that will then be inherited by DesignSync clients and SyncServers. Users can override some of these defaults by choosing their own personal preferences in the DesignSync GUI.

Note: `SYNC_SITE_CNFG_DIR` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `SYNC_SITE_CNFG_DIR` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`

1. The `SyncRegistry.reg` file contains information that is required by DesignSync to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Server Registry Files

The registry files that contain configuration information for SyncServers are listed below in the order of precedence.

- `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/PortRegistry.reg` (UNIX only)
- The `PortRegistry.reg` file contains preference settings that are defined for a particular host `<host>` and port `<port>` number for a specific SyncServer. Since multiple servers can be configured in the same `SYNC_DIR` installation, each server has its own `PortRegistry.reg` registry file. A server's `PortRegistry.reg` file is generated when the server is installed. `SYNC_CUSTOM_DIR` defaults to `<SYNC_DIR>/custom`.

If your server preferences are not port-specific, you can instead include your preferences in the site registry file,

`<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`, which is read by all servers in an installation.

Note: The `SYNC_SITE_CNFG_DIR` environment variable defaults to `<SYNC_CUSTOM_DIR>/site/config`.

- `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`
1. The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. Use the site registry to set up default preferences for all servers in an installation. Then use the port registry to set preferences for a particular SyncServer. Installing the software generates a default `SiteRegistry.reg` file.

Note: `SYNC_SITE_CNFG_DIR` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `SYNC_SITE_CNFG_DIR` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`
1. The `SyncRegistry.reg` file contains information that is required by DesignSync to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the server registry files to take effect, you must restart your SyncServer. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Related Topics

Using Environment Variables

Alphabetical List of Registry Keys

Autoloading stcl Procedures

As a system administrator or project leader, you can create `stcl` procedures for your team. Team members can then use the procedures within their DesignSync sessions without having to perform any set-up steps. You can set up `stcl` procedures for your whole site or for a single project. After you have set up autoloading `stcl` procedures, you can create client triggers to execute the `stcl` procedures based on criteria you can define using **SyncAdmin**, the Synchronicity Administrator tool.

As system administrator or project leader, you place the startup files and scripts in designated site or project directories. DesignSync then loads an `stcl` procedure only when a team member invokes it.

The autoload mechanism works with an index file, `tclIndex`, which indexes the `stcl` procedures. After you place the startup files and scripts in their designated directories, DesignSync generates the index file automatically upon invocation. You can also force the index file to be generated using the `auto_mkindex` procedure. The Autoloading Site and Project `stcl` Procedures section of the *ENOVIA Synchronicity stcl Programmer's Guide* describes how to set up the Tcl autoload capability.

Managing Revision Control Files Between Windows and UNIX

You can use Windows clients to populate, check in, and check out files. The client's workspace can physically reside on file systems that are mounted on both UNIX and Windows systems. Or, data checked in from a Windows system can be accessed by UNIX users, via a mirror directory.

Note:The end-of-line terminations in a text file are interpreted differently on UNIX and Windows. In UNIX, the end-of-line is represented as line feed (LF) while in Windows, it is represented as carriage return/linefeed (CR/LF).

When you use a text editor in UNIX to look at the text file created on Windows, you see `^M` which is a visual representation of the line feed (CR/LF) recognized by DOS and any client running a Windows operating system.

- By default, a SyncServer substitutes LF for CR/LF characters when a text file is checked in from a client running Windows. It is possible for a UNIX client to check the files out again without further cleanup.
- When a file is checked out to a Windows client, the SyncServer substitutes CR/LF characters for LF which you will see as `^M`.

In some situations, however, you may need to retain the `^M` characters found in files. This happens in the following scenario:

1. Windows-specific files are checked in by a Windows client, from a mounted disk on a UNIX file system.
2. Data checked by Windows users in step 1 is mirrored to a directory on a UNIX file system. (See the Mirroring Overview for more information.)
3. The files in the mirror directory are then operated on by Windows tools.

To stop the SyncServer removing the `^M` character from text files on Windows, follow these steps:

1. Start **SyncAdmin**.
2. Navigate to the **General Tab**.
3. Turn off the **Enable CR/LF Processing** button.
4. Restart DesignSync to register the change.

Notes:

Binary files are not processed.

This setting does not apply to UNIX clients. If the `^M` character exists in UNIX text files, they will remain during checkin and checkout operations from UNIX clients.

Related Topics

The General Tab

Managing SyncServer Lists

SyncServer List Files

SyncServer lists provide a mechanism for you to access available SyncServers or vaults from DesignSync. SyncServer lists can contain SyncServer definitions (for example, `sync://linus:2647`) or full vault specifications (for example, `sync://linus:2647/Projects/Sportster`). Under the SyncServers icon in the Tree view are three folders: **My Servers**, **Site Servers**, and **Enterprise Servers**. You can also access these server-list definitions from the Workspace Wizard by clicking the **Browse** button when prompted for the vault URL.

- **My Servers** displays the SyncServers or vaults that you define in your user server-list file:

```
<SYNC_USER_CFGDIR>/sync_servers.txt
```

where `SYNC_USER_CFGDIR` is the environment variable that specifies your directory for user-specific DesignSync customization files. If you have not defined the `SYNC_USER_CFGDIR` environment variable, then DesignSync looks for your `sync_servers.txt` file in:

- UNIX: `$HOME/.synchronicity/`
- WINDOWS: `%AppData%\Synchronicity\`

Note: The `%AppData%` variable points to your Application Data directory. The Application Data directory is typically `C:\Documents and Settings\\Application Data`.

- **Site Servers** displays the SyncServers or vaults that your site administrator defines in your site's server-list file:

```
<SYNC_SITE_CNFG_DIR>/sync_servers.txt
```

where `SYNC_SITE_CNFG_DIR` is the environment variable that specifies the directory for site-specific DesignSync customization files. If you have not defined the `SYNC_SITE_CNFG_DIR` environment variable, then DesignSync looks for:

```
<SYNC_CUSTOM_DIR>/site/config/sync_servers.txt
```

- **Enterprise Servers** displays the SyncServers or vaults that your administrator defines in your enterprise-wide server-list file:

```
<SYNC_ENT_CUSTOM>/config/sync_servers.txt
```

where `SYNC_ENT_CUSTOM` is the environment variable that specifies the directory for enterprise-specific DesignSync customization files. If you have not defined the `SYNC_ENT_CUSTOM` environment variable, then DesignSync looks for:

```
<SYNC_CUSTOM_DIR>/enterprise/config/sync_servers.txt
```

The syntax for the server-list files is as follows:

`NAME <name>` Friendly name for the SyncServer or vault

`REFERENCE <url>` The complete URL to the SyncServer or vault. To refer to a module category, end the URL with a slash (/).

`DESCRIPTION <text>` Brief description of the server or vault

The `NAME` keyword begins a new SyncServer or vault definition. It must appear before the `REFERENCE` and `DESCRIPTION` keywords. The `REFERENCE` and `DESCRIPTION` keywords can appear in any order.

The keywords are case insensitive. Keywords must be the first non-whitespace characters on a line. Each `NAME` value must be unique; duplicates are ignored. If the same `NAME` value appears in both the user and site files, the user definition takes precedence.

The `DESCRIPTION` field is optional. The `DESCRIPTION` text can span multiple lines and is terminated by a blank line or a keyword as the first non-whitespace characters on a line. A comment itself can therefore not include a blank line; otherwise, the remaining comment will be ignored.

Comments are indicated by a pound sign (#) as the first non-whitespace character on a line. Text that does not follow a keyword behaves like a comment. However, the supported keywords may change in future releases, so you should precede all comments with #.

Related Topics

[url servers Command](#)

Example Server-List Files

Example Server-List Files

The following is a sample site `sync_servers.txt`:

```
# This sync_servers.txt file is used by the entire
# Marlboro site.
```

```
NAME Doc Vault
```

```
REFERENCE sync://docserver:2647/Projects/docs
```

```
DESCRIPTION Server for documentation source files.
```

```
Only the documentation group can lock files.
```

```
NAME Source
```

```
REFERENCE sync://src:3001
```

The following is a sample user `sync_servers.txt`:

```
NAME My Server
```

```
REFERENCE sync://localhost:2647
```

```
NAME Source
```

```
REFERENCE sync://src.myco.com:3001
```

```
DESCRIPTION The company-wide source repository
```

The local and the site server files can specify the same SyncServers or vaults. In this example, "Source" is defined as both a site and user SyncServer.

Related Topics

[url servers Command](#)

Managing Symbolic Links

How DesignSync Handles Symbolic Links

DesignSync handles symbolic links (UNIX only) as follows:

- **Links to the cache directory:** DesignSync inherently knows how to manage links to the cache. Check-in operations report that the object hasn't changed because cache files are read-only (no one has access to edit them). Check-out operations change the local object (the link), if necessary, to the appropriate state (locked or unlocked file, link to the mirror) depending on your check-out mode.

Note: You can alternatively use hard links instead of symbolic links for cache files. For more information, see [Tips for Administering LAN Caches](#).

- **Links to a mirror directory:** Same as links to the cache.
- **Links to files (other than files in the mirror or cache):** By default, DesignSync dereferences links and operates on the file to which the link points. DesignSync checks in the file pointed to by the link, rather than the link itself, and leaves behind the file, a link to the cache or mirror, or a reference, depending on your check-in mode. When the file (from the dereferenced link) is checked back out, the file is retrieved. (This behavior of links is similar to **tar -h**.)

Your project leader can change this behavior so that DesignSync revision-controls the links themselves.

- **Links to directories (folders):** By default, DesignSync treats links to folders as though the link were the folder itself. The link appears in the Tree View, and you can follow the link, descending into the folder pointed to by the link. You can check in the contents of the folder by performing a recursive checkin (either from the GUI or by specifying **-recursive** from the command line). If you try to check in a link to a folder without recursing the folder, DesignSync tries to check in the folder itself, which fails because DesignSync doesn't revision-control folders.

Your project leader can change this behavior so that DesignSync revision controls the links themselves.

DesignSync uses a link overlay with the object icon (file or folder) for a symbolic link icon no matter what the link points to (file, folder, cached object, mirrored object). For example, a replica of a symbolic link that is

under revision control has a link overlay, not a replica overlay. The Type field in the List View provides information about the revision-control state.

Related Topics

Guidelines for Creating Symbolic Links

Revision Controlling Symbolic Links to Files

Revision Controlling Symbolic Links to Folders

Guidelines for Creating Symbolic Links

This is a set of best practices and guidelines for creating and managing symbolic links on UNIX:

- For a client vault, do not create symbolic links.
- For a server vault:
 - You can create symbolic links at the folder level. For example, you can create a symbolic link from a vault folder to a directory on another disk.
 - Establish links at the project level, to avoid problems that can occur with links on lower levels of the hierarchy.
 - Do not create symbolic links to vaults, for example, to individual *.rca files, view folders, or cell folders.

Important: You cannot check symbolic links to module members into a CopyVault. For more information on Vault types, see About Vault Types.

For example, suppose your ASIC project on `sync://<host>:<port>/Projects/ASIC` runs out of disk space. To provide space for the project files, you can move the ASIC vault folder to another disk and create a symbolic link from the vault folder to the directory on the other disk:

1. Stop the SyncServer.
2. Change directory to the Projects directory of the vault.
3. Move the ASIC project directory to disk 2.
4. Within the syncdata directory (without a `sync://` protocol), create a symbolic link to the ASIC folder of the vault.

```
% cd ...syncdata/<host>/<port>/server_vault/Projects
% mv ASIC <disk2 path>/ASIC
% ln -s <disk 2 path>/ASIC ASIC
```

Now `...syncdata/<host>/<port>/server_vault/Projects/ASIC` is a symbolic link to `<disk2 path>/ASIC`.

5. Restart the SyncServer.

Note: Users can link from a work area to a module cache. For information about module caches, see the topic DesignSync Data Management User's Guide: What is a Module Cache?

Related Topics:

How DesignSync Handles Symbolic Links

Revision Controlling Symbolic Links to Files

Revision Controlling Symbolic Links to Folders

Revision Controlling Symbolic Links to Files

By default, DesignSync handles symbolic links to files (other than links to the cache or mirror) by dereferencing the link and operating on the file pointed to by the link. You, the project leader or LAN administrator, can change this behavior so that DesignSync maintains and manages links rather than dereferencing them. Individual team members cannot change the DesignSync treatment of links because a team methodology must be enforced to ensure data consistency.

When DesignSync manages links, the link itself, not the file pointed to by the link, becomes a versionable object. A checkout of the link recreates the link in your work area. This methodology relies on the proper use and management of links by the team sharing the vault. For example, if the link is defined using an absolute path, that path needs to be accessible by all team members to be meaningful. Links defined with relative paths are typically more useful.

You can control the management of Revision Control Symbolic Links through SyncAdmin.

Cache behavior

DesignSync converts a checkout of a symbolic link in link-to-cache (-share) mode to a checkout of an unlocked copy (-get). DesignSync does not support checking out a link as a link to the cache because the link would not resolve correctly. DesignSync generates its own names for files it manages in the cache because more than one version of a file can exist in the cache.

If DesignSync were to support the checkout of a link (top_link.v) to a file (top.v) in the cache, the link would not be resolved because the filename, top.v, pointed to by the link does not actually exist in the cache. The file top.v may exist in the cache, but it would have a DesignSync-generated filename. Instead, DesignSync copies the link directly into your work area upon a checkout of the link in link-to-cache (-share) mode. Thus, the

link points correctly to top.v in your work area even if top.v itself is a link to the cache. Note that checking out a symbolic link as a link to a file in the mirror directory does not need this special treatment, because mirror files have the actual filenames.

Mirror behavior

DesignSync converts a checkout of a symbolic link to an object in Links to mirror (-mirror) mode to a checkout of an unlocked copy (-get). DesignSync copies the link directly into the your work area upon a checkout of the link using Link to Mirror (-mirror) mode.

Related Topics

[Guidelines for Creating Symbolic Links](#)

[How DesignSync Handles Symbolic Links](#)

[Revision Controlling Symbolic Links to Folders](#)

[ENOVIA Synchronicity Administrator Tool Overview](#)

Revision Controlling Symbolic Links to Folders

By default, DesignSync treats symbolic links to folders as though the link were the folder itself. Trying to check in a link to a folder would therefore fail because DesignSync does not revision control folders. (Users can perform, however, a recursive checkin on a folder to check in the folder's contents). You, the project leader or LAN administrator, can change this behavior so that DesignSync maintains and manages links rather than dereferencing them. Individual team members cannot change the DesignSync treatment of links because a team methodology must be enforced to ensure data consistency.

In "manage link" mode, links to folders are versionable objects. When you check out a link, DesignSync recreates the link using the same link definition that existed when the link was checked in. This methodology relies on the proper use and management of links by the team sharing the vault. For example, if the link is defined using an absolute path, that path needs to be accessible by all team members to be meaningful. Links defined with relative paths are typically more useful.

Users are able to browse into the folder pointed to by a revision-controlled link to folder. However, the linked folder appears under **Bookmarks** instead of under **My Computer** as other folders do. This behavior differentiates a link to a folder from an actual folder.

You can set the default behavior for Revision Control Symbolic Links using the SyncAdmin tool.

Cache behavior

DesignSync converts a checkout of a symbolic link to a folder in link-to-cache (-share) mode to a checkout of an unlocked copy (-get). Checking out a symbolic link as a link to the cache would result in undesirable behavior if the symbolic link were defined using a relative path. This is because a cache does not contain the design hierarchy -- the folder referred to by the link would not exist in the cache. Instead, DesignSync copies the link directly into the your work area upon a checkout of the link in link-to-cache (-share) mode. Thus, the link points correctly to the folder in your work area.

Mirror behavior

DesignSync converts a checkout of a symbolic link to a folder in Links to mirror (-mirror) mode to a checkout of an unlocked copy (-get). DesignSync copies the link directly into the your work area upon a checkout of the link using Link to Mirror (-mirror) mode.

Related Topics

[Guidelines for Creating Symbolic Links](#)

[Revision Controlling Symbolic Links to Files](#)

[How DesignSync Handles Symbolic Links](#)

[ENOVIA Synchronicity Administrator Tool Overview](#)

Setting up Client Triggers

Triggers Overview

Triggers are watchpoints that cause an action to occur automatically in response to some other action, such as checking in a file. DesignSync triggers can be defined to execute a Tcl script whenever a revision-control operation occurs. For example, you could create a trigger that compresses files when they are checked in and uncompresses them again when the files are checked out.

You create and manipulate triggers from any DesignSync client (dss, dssc, stcl, stcl, or DesSync command bar). Your DesignSync administrator can also create site-wide client triggers using the SyncAdmin tool.

Important: When creating preObject triggers, do not use `url` commands that call the server. The `url` command can collide with the command that spawned the trigger resulting in unexpected failures.

Note: The trigger mechanism is client-side only; triggers are launched from the DesignSync client.

The trigger commands let you:

- Create new triggers.
- Replace the properties of a trigger.
- Delete triggers.
- Temporarily enable or disable the operation of individual triggers.
- Determine whether a trigger is enabled.
- Fire (execute) triggers
- Get information about triggers.
- List available triggers.
- Determine the status of available triggers.

The trigger commands give you a great deal of control over how and when your triggers fire. For each revision-control operation, such as checking in, you can set triggers that fire only under specified conditions. For example, you can create a trigger that checks whether a user is permitted to check in certain types of files and aborts the check-in of a file if the user does not have permission.

To give you this refined control over the actions of your triggers, the trigger commands let you specify not only the revision-control operation, but also the events and properties of events that occur as part of the operation.

Supported Commands

The Supported Trigger Commands topic provides a list of commands you can use to generate trigger events.

Events and Event Properties

Every revision-control operation generates a series of **events**. For example, when you issue the **tag** command, the operation creates the following events:

- A pre-command event (**preCommand**), before any objects have been tagged
- A pre-folder event (**preFolder**), before the objects in a folder (directory) have been tagged
- A pre-object event (**preObject**), before each object is tagged
- A post-object event (**postObject**), after each object has been tagged
- A post-command event (**postCommand**), when the tag command has completed

Every event, in turn, has an associated set of **properties** that describe the event. For example, when you tag an object, the pre-object event (**preObject**) has a path property (**objURL**) that corresponds to the URL of the object to be tagged.

Each revision-control operation has its own set of associated events, and each event has its own set of associated event properties.

You can specify events and event properties in trigger scripts so that your triggers fire in response to both events and event properties.

See Events Overview for more information about events and event properties.

Working with Triggers

When you work with triggers, keep in mind the following guidelines:

- If you use Tcl expressions in your trigger command, you should work with triggers in stcl mode (stcl or stclc shell); otherwise, you can work in either stcl or dss mode. However, because dss mode (dss/dssc) does not correctly parse Tcl expressions (such as lists), all such expressions must be quoted.
- If you create triggers in dss mode, use quotation marks (") to enclose lists, not curly braces ({ }).
- You can use the **-exec** option to specify a command and its arguments to be executed when the trigger is activated. To include event property names as arguments:

If you are using the **dss or dssc command shell**, precede the event property name with a dollar sign (using Tcl variable syntax). For example:

```
dss> trigger create test -require type preObject -exec
"xterm -e vi $objURL"
```

If you are using the **stcl or stclc command shell**, precede the event property name with a backslash and a dollar sign or put the entire argument inside curly braces. For example:

```
stcl> trigger create -replace test -require type
preObject -exec "xterm -e vi \ $objURL"
```

```
stcl> trigger create -replace test -require type
preObject -exec {xterm -e vi $objURL}
```

- When you start a dss or stcl session, the syncd process reads any existing trigger information in your registry. If you later create a trigger from the GUI command shell window, you need to reread the registry by restarting the dss or stcl syncd process. Otherwise, the trigger you created in the GUI is not recognized when you issue command-line dss or stcl commands. The dssc and stclc shells do not use the syncd process and are generally the recommended DesignSync shells.

- You can specify multiple triggers to fire for one event. However, you cannot determine the order in which multiple triggers will fire.
- A trigger applied to collection objects fires once for each collection object, not for each object contained in the collection.
- If you use temporary filenames to communicate information between **preObject** and **postObject** triggers, you need to make sure your filenames are unique. With the multithreading optimization, multiple **preObject** triggers might be run before any **postObject** triggers are run. For example, a **preObject** trigger which writes to a temporary file, `<TMP_DIR>/tempfile`, might not work, because the data will be written multiple times before any **postObject** trigger gets a chance to read it. If your triggers use a temporary filename that is not unique, your DesignSync administrator can turn off multithreading using the Performance tab of the SyncAdmin tool.

Related Topics

[Creating Triggers](#)

[Manipulating Triggers](#)

[Supported Trigger Commands](#)

[Understanding Trigger Hooks](#)

[Events Overview](#)

[Event Properties](#)

Creating Triggers

The **trigger create** command lets you define triggers that fire in response to specified events created by a revision-control operation.

By default, triggers fire for every event in an operation. Because one operation (e.g., checking out) is composed of several events, your trigger will fire repeatedly unless you limit the number of events the trigger responds to.

Important: Client triggers should not be used to modify data being processed by the calling command, specifically:

1. Triggers should not attempt to modify the data being processed with additional DesignSync commands within the trigger.
2. Triggers should not use the `exec` command to launch DesignSync commands that modify the data being processed within a trigger.
3. Triggers should not use URL commands to modify the data being processed.

To prevent the trigger from firing for every event, the **trigger create** command includes options that let you specify properties that:

- Must exist for the trigger to fire.
- Prevent the trigger from firing if they exist.

(See Events Overview for more information on events and the association between events and operations.)

A trigger consists of a name, events or event properties to require or exclude, and the action to perform. The **trigger create** command has the following syntax:

```
trigger create <name>
[-require <property> <value_list> ...]
[-exclude <property> <value_list> ...]
[-exec <command_and_arguments> | -tcl_script <script>
| -tcl_file <filename> | -tcl_store <filename>]
```

<name>	The name you assign to the trigger.
-require	<p>Fires the trigger only when the specified condition exists. The arguments to -require are an event property name followed by a value list. For example, if you specify</p> <pre>-require type preObject</pre> <p>type is the name of an event property; preObject is the value of the type event property. In this case, the trigger fires only when the event has a property named type and its value matches the expression preObject.</p> <p>Or, if you specify</p> <pre>-require command {tag co}</pre> <p>command is the name of an event property; tag and co are the values of the command event property. In this case, the trigger fires only when the event has a property named command and its value is either tag or co.</p> <p>Note: You cannot specify populate as a value for the command event property. Use co instead.</p> <p>Or, if you specify</p>

	<pre>-require objPath {*.c *.cpp}</pre> <p>objPath is the name of an event property; *.c and *.cpp are the values of the objPath event property. In this case, the trigger fires only when the event has a property named objPath and the value of the property is a name ending with .c or ,cpp.</p>
-exclude	<p>Fires the trigger only when the specified condition does not exist. The arguments are specified in the same format as those for -require.</p> <p>For example, if you specify</p> <pre>-exclude user {bob joe}</pre> <p>the trigger fires only when the user is not bob or joe.</p>
-exec	<p>Executes the specified command and its arguments. Before execution, any variables in the command line are replaced by the named event property value.</p> <p>To include event property names as arguments:</p> <p>If you are using the dss or dssc command shell, precede the event property name with a dollar sign (using Tcl variable syntax). For example:</p> <pre>-exec "xterm -e vi \$objURL"</pre> <p>If you are using the stcl or stcl command shell, precede the event property name with a backslash and a dollar sign or put the entire argument inside curly braces. For example:</p> <pre>-exec "xterm -e vi \\$objURL"</pre> <pre>-exec {xterm -e vi \$objURL}</pre>
-tcl_script	<p>When the trigger fires, stores and executes the Tcl script that you enter on the command line. For example:</p> <pre>-tcl_script { puts "Hello, world" }</pre> <p>You should be in stcl mode to execute a trigger create command using this option; otherwise, you may have difficulty escaping the script.</p>
-tcl_file	<p>Stores the specified file name. When the trigger fires, loads and executes the specified file. For example:</p> <pre>-tcl_file /home/tmmf/scripts/reportCheckouts.tcl</pre>

	<p>Use this option when you want to edit the trigger script and have your changes take effect without calling trigger create again.</p> <p>Note: The trigger searches only the current folder for the specified Tcl file. If you want to use a script in another folder, you must give the full file system path. Triggers do not recognize environment variables in your path specifications.</p>
-tcl_store	<p>Loads and stores the content of the specified file, but does not execute the script until the trigger fires. For example:</p> <pre>-tcl_store /home/tmmf/scripts/listCheckoutTimes.tcl</pre> <p>If you use this option, you must call trigger create again after you make any edits to your trigger script; otherwise, the stored version of your trigger is not updated.</p>

Note: DesignSync does not allow a trigger to require or exclude an event property that has a null value or is an empty string (has a value of {}, "", or ' '). For example, if you define a trigger that has the option: `-require comment "{}"`, DesignSync creates and saves the trigger but drops the comment from its definition.

To create a trigger that fires based on the presence or absence of a comment, use the **isComment** event property. See Example Client Triggers for an example of this type of trigger.

The following command creates a trigger named **logCheckins** that maintains a log file of all **.v** and **.vlog** files that are checked in by all users except **jackie**. (Because this trigger uses **-tcl_script**, it would be typed in as one line on the command line; here, it is formatted for readability.)

```
trigger create logCheckins \
  -require command ci \
  -require objPath "*.v *.vlog" \
  -require type postObject \
  -exclude user jackie \
  -tcl_script {
set fd [open [glob ~/checkin.log] a];
puts $fd "$objURL";
```

```
close $fd
}
```

In this example, the first **-require** argument takes **command** as the property and the **ci** command is the value. This option makes the trigger fire only when check-in is the revision-control operation.

The second **-require** option takes **objPath** as the property and **"*.v, *.vlog"** are the values. This option makes the trigger fire only when the checked-in file has the file extension **.v** or **.vlog**.

The third **-require** option takes **type** as the property and the **postObject** event is the value. This option makes the trigger fire after each file is checked in.

The **-exclude** option takes **user** as the property name and **jackie** is the value. This option makes the trigger fire only when the user is not Jackie, even if all other required conditions are met.

The **-tcl_script** option indicates that the Tcl script is entered on the command line.

See Event Properties for a list of the event properties and the associated values that you can use with the **-require** and **-exclude** options.

Note: The **trigger create** command also has a **-replace** option. See **trigger create -replace** for more information.

See **trigger create** in the Command Reference for complete details on the **trigger create** command.

Related Topics

[Triggers Overview](#)

[Manipulating Triggers](#)

[Events Overview](#)

[Event Properties](#)

Supported Trigger Commands

Supported Trigger Commands

The following operations support the use of triggers. The operations in the list below are displayed with the associated command used in the trigger. Click on the operation or

command below for a definition of each supported property and event type for the command.

Notes:

The command event property value for the DesignSync `populate` command is "co," as noted below.

The `mkmod -checkin` operation fires `ci` triggers at the command level, and at the object level.

- Cancel (`cancel`)
- Check in (`ci`)
- Check out (`co`)
- Delete file (`rmfile`)
- Delete folder (`rmfolder`)
- Delete vault (`rmvault`)
- Delete version (`rmversion`)
- Populate (`co`)
- Retire (`retire`)
- Make Area (`sdamk`)
- Remove Area (`sdarm`)
- Tag (`tag`)
- Unlock (`unlock`)

Related Topics

[Creating Triggers](#)

[Manipulating Triggers](#)

[Events Overview](#)

[Event Properties](#)

Supported Trigger Commands

The following operations support the use of triggers. The operations in the list below are displayed with the associated command used in the trigger. Click on the operation or command below for a definition of each supported property and event type for the command.

Notes:

The command event property value for the DesignSync `populate` command is "co," as noted below.

The `mkmod -checkin` operation fires `ci` triggers at the command level, and at the object level.

- Cancel (`cancel`)
- Check in (`ci`)
- Check out (`co`)
- Delete file (`rmfile`)
- Delete folder (`rmfolder`)
- Delete vault (`rmvault`)
- Delete version (`rmversion`)
- Populate (`co`)
- Retire (`retire`)
- Make Area (`sdamk`)
- Remove Area (`sdarm`)
- Tag (`tag`)
- Unlock (`unlock`)

Related Topics

[Creating Triggers](#)

[Manipulating Triggers](#)

[Events Overview](#)

[Event Properties](#)

Cancel (`cancel`) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
fetchState		X	X	
groupID	X	X	X	X
host	X	X	X	X
isForce	X	X	X	X
isKeep	X	X	X	X

DesignSync Data Manager Administrator's Guide

isMirror	X	X	X	X
isRecursive	X			X
isReference	X	X	X	X
isServer	X	X	X	X
isShare	X	X	X	X
objPath		X	X	
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X
parentPID	X	X	X	X
selector		X	X	
statusOK			X	
statusText			X	X
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

Check in (ci) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Folder	pre Object	from VaultFilter	to VaultFilter	post Object	post Command
command	X	X	X		X	X	X
comment	X	X	X		X	X	X
errorCount							X
fetchState			X		X	X	
fromFile				X	X	X	
groupID	X	X	X	X	X	X	X
host	X	X	X	X	X	X	X
isBranch	X						X
isComment	X	X	X		X	X	X
isForce	X	X	X		X	X	X
isKeep	X	X	X		X	X	X
isLock	X	X	X		X	X	X
isMirror	X	X	X		X	X	X
isNew	X	X	X		X	X	X
isRecursive	X	X					X

isRetain	X	X	X		X	X	X
isServer	X	X	X	X	X	X	X
isShare	X	X	X		X	X	X
isSkip	X	X	X		X	X	X
keys	X	X	X		X	X	X
objList		X					
objPath		X	X	X	X	X	
objURL		X	X		X	X	
okCount							X
osName	X	X	X	X	X	X	X
osVersion	X	X	X	X	X	X	X
parentPID	X	X	X	X	X	X	X
selector	X		X		X	X	X
statusOK						X	
statusText						X	X
toFile				X	X	X	
trigArgs	X	X	X		X	X	X
trigger	X	X	X	X	X	X	X
type	X	X	X	X	X	X	X
user	X	X	X	X	X	X	X
	pre Command	pre Folder	pre Object	from VaultFilter	to VaultFilter	post Object	post Command

Check out (co) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Note

fromVaultFilter triggers are not fired in a set-user-id (SUID) installation for -share and -mirror operations.

Property/Type	pre Command	pre Object	from VaultFilter	post Object	post Command
command	X	X		X	X
comment	X	X		X	X
errorCount					X
fetchState		X		X	
fromFile			X		
groupID	X	X	X	X	X
host	X	X	X	X	X
isComment	X	X		X	X

isForce	X	X		X	X
isGet	X	X		X	X
isLock	X	X		X	X
isMerge	X	X		X	X
isMirror	X	X		X	X
isOverlay	X	X		X	X
isRecursive	X				X
isRetain	X	X		X	X
isServer	X	X	X	X	X
isShare	X	X		X	X
isVersion	X				X
keys	X	X		X	X
objPath		X	X	X	
objURL		X		X	
okCount					X
osName	X	X	X	X	X
osVersion	X	X	X	X	X
parentPID	X	X	X	X	X
selector	X	X		X	X
statusOK				X	
statusText				X	X
toFile			X		
trigArgs	X	X		X	X
trigger	X	X	X	X	X
type	X	X	X	X	X
user	X	X	X	X	X
version		X		X	
	pre Command	pre Object	from VaultFilter	post Object	post Command

Populate (co) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Notes

fromVaultFilter triggers are not fired in a set-user-id (SUID) installation for -share and -mirror operations.

The command event property value for the DesignSync `populate` command is "co".

Property/Type	pre Command	pre Object	from VaultFilter	post Object	post Command
command	X	X		X	X

comment	X	X		X	X
errorCount					X
fetchState		X		X	
fromFile			X		
groupID	X	X	X	X	X
host	X	X	X	X	X
isComment	X	X		X	X
isForce	X	X		X	X
isGet	X	X		X	X
isLock	X	X		X	X
isMerge	X	X		X	X
isMirror	X	X		X	X
isOverlay	X	X		X	X
isRecursive	X				X
isRetain	X	X		X	X
isServer	X	X	X	X	X
isShare	X	X		X	X
keys	X	X		X	X
objPath		X	X	X	
objURL		X		X	
okCount					X
osName	X	X	X	X	X
osVersion	X	X	X	X	X
parentPID	X	X	X	X	X
selector		X		X	
statusOK				X	
statusText				X	X
toFile			X		
trigArgs	X	X		X	X
trigger	X	X	X	X	X
type	X	X	X	X	X
user	X	X	X	X	X
version		X		X	
	pre Command	pre Object	from VaultFilter	post Object	post Command

Retire (retire) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
branchURL		X	X	

DesignSync Data Manager Administrator's Guide

command	X	X	X	X
errorCount				X
groupID	X	X	X	X
host	X	X	X	X
isForce	X	X	X	X
isKeep	X	X	X	X
isRecursive	X			X
isServer	X	X	X	X
isUnretire	X	X	X	X
objPath		X	X	
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X
parentPID	X	X	X	X
selector	X			X
statusOK			X	
statusText			X	X
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

Delete File (rmfile) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
fetchState		X	X	
groupID	X	X	X	X
host	X	X	X	X
isServer	X	X	X	X
objPath		X	X	
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X

parentPID	X	X	X	X
statusOK			X	
statusText			X	
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

Delete Folder (rmfolder) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
groupID	X	X	X	X
host	X	X	X	X
isKeepVID	X	X	X	X
isNoKeepVID	X	X	X	X
isRecursive	X	X	X	X
isServer	X	X	X	X
objPath		X*	X*	
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X
parentPID	X	X	X	X
statusOK			X	
statusText			X	
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

*** Note:**

These properties only occur when a local folder is being removed. If a server folder is being removed, there is no objPath property.

Delete Vault (rmvault) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
groupID	X	X	X	X
host	X	X	X	X
isForce	X	X	X	X
isKeepVID	X	X	X	X
isNoKeepVID	X	X	X	X
isServer	X	X	X	X
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X
parentPID	X	X	X	X
statusOK			X	
statusText			X	
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

Delete Version (rmversion) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
groupID	X	X	X	X
host	X	X	X	X
isForce	X	X	X	X
isServer	X	X	X	X
objURL		X	X	
okCount				X
osName	X	X	X	X

osVersion	X	X	X	X
parentPID	X	X	X	X
statusOK			X	
statusText			X	
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X
user	X	X	X	X
	pre Command	pre Object	post Object	post Command

Tag (tag) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Property/Type	pre Command	pre Folder	pre Object	post Object	post Command
command	X	X	X	X	X
comment	X				X
errorCount					X
groupID	X	X	X	X	X
host	X	X	X	X	X
isBranch	X		X	X	X
isComment	X				X
isDelete	X	X	X	X	X
isImmutable	X				X
isRecursive	X				X
isReplace	X	X	X	X	X
isServer	X	X	X	X	X
isVersion	X				X
objList		X			
objPath		X			
objURL		X	X	X	
okCount					X
osName	X	X	X	X	X
osVersion	X	X	X	X	X
parentPID	X	X	X	X	X
selector	X				X
statusOK				X	
statusText				X	X
tag	X	X	X	X	X
trigArgs	X	X	X	X	X

trigger	X	X	X	X	X
type	X	X	X	X	X
user	X	X	X	X	X
	pre Command	pre Folder	pre Object	post Object	post Command

Make Area (sdamk) Trigger

As part of the operation of making a new development area, the sda mk command populates the development area. The trigger operations are tied to the execution of populate. The PreCommand trigger runs before the populate operation, and the postCommand trigger runs after the populate operation.

Make Area PreCommand Trigger Properties

When Creating Triggers to use with Make Area, the triggers are passed to the Populate Command Trigger with the following properties:

Property	Description
command	The name of the command, <code>sdamk</code> .
type	The type of trigger, <code>preCommand</code>
objPath	The path to the development area.
objURL	The directory name where the objects are going to be populated.
isModule	Boolean value indicating whether the objURL is a module.
vaultURL	The vault URL.
selector	The selector for the development.
view	The view set for the development, if applicable.

Make Area PostCommand Trigger Properties

When Creating Triggers to use with Make Area, the Populate Command Trigger returns the following properties to the Make Area command trigger:

Property	Description
command	The name of the command, <code>sdamk</code>
type	The type of trigger, <code>postCommand</code>
objPath	The path to the development area.
topDir	The directory name where the objects were populated.
isModule	Boolean value indicating whether the objURL is a module.
vaultURL	The vault URL.
selector	The selector for the development.

view	The view set for the development, if applicable.
statusOK	Indicates whether the command completed successfully: 1 - Enclosed command completed successfully. 0 - Enclosed command failed.
errorCount	Integer count of the objects that failed to populate.
okCount	Integer count of the objects that successfully populated.

Remove Area (sdarm) Trigger Properties

As part of the operation of removing a development area, the `sda rm` command removes the development area using the `rmfolder` command. The trigger operations are tied to the execution of `rmfolder`. The `PreCommand` trigger runs before the `rmfolder` operation, and the `postCommand` trigger runs after the `rmfolder` operation.

Remove Area PreCommand Trigger Properties

When Creating Triggers to use with Remove Area, the triggers are passed to the Delete Folder Command Trigger with the following properties:

Property	Description
command	The name of the command, <code>sdarm</code> .
type	The type of trigger, <code>preCommand</code>
objPath	The path to the development area.
objURL	The directory name being removed.
isModule	Boolean value indicating whether the <code>objURL</code> is a module.
vaultURL	The vault URL.
selector	The selector for the development.
view	The view set for the development, if applicable.

Remove Area PostCommand Trigger Properties

When Creating Triggers to use with Remove Area, the Delete Folder Command Trigger returns the following properties to the Make Area command trigger:

Property	Description
command	The name of the command, <code>sdarm</code>
type	The type of trigger, <code>postCommand</code>
objPath	The path to the development area.
topDir	The top level directory in which the objects had been populated.
isModule	Boolean value indicating whether the <code>objURL</code> is a module.
vaultURL	The vault URL.

selector	The selector for the development.
view	The view set for the development, if applicable.
statusOK	Indicates whether the command completed successfully: 1 - Enclosed command completed successfully. 0 - Enclosed command failed.
errorCount	Integer count of the objects that failed to populate. (Optional)
okCount	Integer count of the objects that successfully populated. (Optional)

Unlock (unlock) Trigger Properties

"X" indicates that the property is available for use by the trigger.

Note:

If you use `unlock` without the `-branch` option, the unlock trigger will generate `preObject` and `postObject` cancel events.

Property/Type	pre Command	pre Object	post Object	post Command
command	X	X	X	X
errorCount				X
fetchState		X*	X*	
groupID	X	X	X	X
host	X	X	X	X
isForce		X*	X*	
isKeep		X*	X*	
isMirror		X*	X*	
isRecursive	X			X
isReference		X*	X*	
isServer	X	X	X	X
isShare		X*	X*	
objPath		X*	X*	
objURL		X	X	
okCount				X
osName	X	X	X	X
osVersion	X	X	X	X
parentPID	X	X	X	X
selector	X	X*	X*	X
statusOK			X	
statusText			X	X
trigArgs	X	X	X	X
trigger	X	X	X	X
type	X	X	X	X

user	X	X	X	X
	pre Command	pre Object	post Object	post Command

* will only appear if `unlock` is used without `-branch`.

DesignSync Provided Trigger Hooks

Understanding Trigger Hooks

Triggers use hooks into the code to perform specific tasks. To use the DesignSync hooks, you supply the hook name as an argument to the trigger command and provide the required input and/or handle the output, depending on the hook's requirements.

The following table lists the documented trigger hooks. Click on the trigger hook name for additional information:

Hook Name	Description
<code>externalAuthHook</code>	Provides a mechanism for DesignSync user authentication when the software requires a login.
<code>moduleSelectionHook</code>	Provides logic to help smart module detection determine the appropriate module to select for add and checkin operations.
<code>proxyDiscoveryHook</code>	Allows any client program you run discover which proxy to use for any server.
<code>transferHook</code>	Controls communications between the client calling the hook and the server(s) defined within the trigger allowing you to restrict delta transfer to servers that benefit from the delta transfer.
<code>transferDeltaHook</code>	Determines whether to optimize delta transfer by defining which objects are excluded from attempting to use the delta transfer feature.

Module Selection Hook

DesignSync smart module detection features the ability to customize the selection of a module in a case where the folder containing a candidate module member belongs to more than one module using the `moduleSelectionHook`.

The `moduleSelectionHook` takes two input properties:

- `memberPath` - path to the candidate module member.
- `modulesInstanceList` - list of lists for candidate module instances. Each sub-list contains two values for a module instance: module instance name and module base directory

and one output property:

- `selectedModule` - the module determined to be the appropriate target module for the candidate member. This will either be a value from the `modulesInstanceList` or a null string ("") which will then tell the spawning checkin or add operation to prompt the user to select a module from the generated list of module instances.

Example of a Module Selection Client Trigger

This example identifies the target module as the module with the base directory closest to the candidate module member.

```
trigger create closestModuleHook -require type
moduleSelectionHook -tcl_script {

    set selectedModule ""

    set length 0

    foreach module $moduleInstanceList {

        set baseDir [lindex $module 1]

        if {[string first $baseDir $memberPath] == 0} {

set baseLength [string length $baseDir]

        if {$length < $baseLength} {

            set length $baseLength

            set selectedModule [lindex $module 0]

        }

    }

}
```

```

    }
}
}

```

Related Topics

Understanding Smart Module Detection

Understanding Trigger Hooks

Using Delta Transfer Hooks to Tune Client/Server Communications

DesignSync features the option to transfer only the changes between subsequent file versions to provide more efficient file transfers over high latency/poor bandwidth networks thereby saving engineering time and network resources

This feature, called delta transfer, is a transfer compression mechanism that relies on computing and transferring the difference between the file that needs to be sent and another file with a similar content (base file). On the receiving end, the original file is restored by applying the difference to a copy of the base file. In cases where differences are small, this mechanism can result in a drastic reduction of the transfer size. For more information on how to use delta transfer, see [Enable file delta transfer between client and server](#).

You can enable or disable using delta transfer as the default transfer mechanism for client/server interaction with SyncAdmin using the Communications tab. You can also use client triggers. DesignSync provides two client triggers, `TransferHook`, to control the transfer mechanism on a per-server basis, and `TransferDeltaHook`, to control the transfer mechanism on a per-object basis.

Using the `TransferHook` client trigger

The `TransferHook` client trigger controls communications between the specific client calling the hook and the server(s) defined within the trigger. This allows you to restrict delta transfer to servers that benefit from the delta transfer.

The `TransferHook` trigger also allows you to set the buffer size to increase bandwidth use according to the available network.

DesignSync allows the following output values for `deltaTransfer` and `compressedTransfer`:

- **ALLOW** - allow the operation to use the specified transfer mechanism.
- **DENY** - do not allow the operation to use the specified transfer mechanism.

- UNKNOWN - use the default transfer mechanism.

Example of a TransferHook client trigger

This example shows a `transferHook` trigger. It uses an event with two input properties: `serverName` and `serverIP`. It may have one or more output properties: `deltaTransfer`, `compressedTransfer`, and `socketBufferSize`.

```
trigger create wanHook -require type transferHook -tcl_script {
  if {[string first $serverIP "10.1."] != 0} {
    # Server is on the WAN
    set deltaTransfer ALLOW
    set compressedTransfer DENY
    # Socket buffer size in KB.
    set socketBufferSize 512
  }
}
```

Using the TransferDeltaHook client trigger

The `TransferDeltaHook` client trigger examines all the objects being operated on by the command that called the trigger, so can be expensive in terms of time to run, however it can save processing time on the server by excluding objects from attempting to use the delta transfer feature. Using `TransferDeltaHook` you can exclude objects that meet any of the following criteria:

- compressed files
- files below a certain size (which therefore do not reap the benefits of delta transfer)

The `TransferDeltaHook` is only called when the operation is being performed on a server that has delta transfer enabled.

Note: The `TransferHook` must set an `ENABLE` value on `DeltaTransfer` in order to use the `TransferDeltaHook` trigger on the desired objects.

Example of a TransferDeltaHook client trigger

This example shows a `transferDeltaHook` trigger and uses an event with one input property `objPath`, and one output property, `deltaTransfer` with values **ALLOW**, **DENY**, **UNKNOWN**. To enable delta transfer for a file the hook must return **ALLOW**. This hook is invoked per file and disables delta transfer for any files compressed with gnu zip or tar.

```
trigger create deltaHook -require type transferDeltaHook -tcl_script {
  if {[string equal [file extension $objPath] ".gz"] |
      [string equal [file extension $objPath] ".tar"]} {
    # Compressed file
    set deltaTransfer DENY
  } else {
    set deltaTransfer ALLOW
  }
}
```

Related Topics

Communications

Data Compression

Delta Transfer Compression

User Authentication Hook

To set up user authentication, you create a trigger that fires whenever a user tries to access an area of the DesignSync Web UI that requires login information. A special event, `externalAuthHook`, is provided to enable user authentication.

The `externalAuthHook` event takes four variables:

- `user` - The user name supplied by the user,
- `password` - The password supplied by the user.
- `clientIP` - The IP address of the client program.
- `clientType` - The type of client program. The value of this variable is either `DesSync`, when the client is DesignSync, or `UNKNOWN` for any other client, including the internet browser.

You cannot change these variables within your script; any changes are ignored and the variables are reset. So, for example, you cannot change a user name with your trigger.

The result of the authorization is stored in the `authResult` variable. It takes one of the following return values:

- `ALLOW` - Authentication succeeded and the user is granted access.
- `DENY` - Authentication failed and the user is denied permission.
- `UNKNOWN` - Neither `ALLOW` nor `DENY` was returned. When the return value is `UNKNOWN`, DesignSync attempts to access the LDAP client. If this attempt fails, the user is denied access. `UNKNOWN` is the default value.
- `DECLINE` - DesignSync's authentication routines are ignored and attempts to authenticate the user using other methods (for example, Apache authentication).

If you have more than one trigger, the `authResult` variable is reset before each trigger is invoked. Only the result of the last-invoked trigger is used.

The following simple Tcl script illustrates the use of the `externalAuthHook` event and its return values:

DesignSync Data Manager Administrator's Guide

```
trigger create authAll -replace \  
    -require type externalAuthHook -tcl_script {  
if { $user == "joe" } {  
    # Deny access to joe  
    set authResult DENY  
    return  
}  
  
# If the user doesn't exist, create the user.  
if { [catch {note getprop /Users/$user Name}] } {  
    user create $user [list Name $user \  
        EmailAddr "$user@company.com" ClearKey $password]  
}  
  
# The authResult variable does not need to be set,  
# because it is already equal to UNKNOWN.  
# Therefore, ProjectSync rechecks the user in the  
# database after the trigger is invoked.  
  
# However, if ALLOW is assigned to authResult,  
# registered users can enter with any password.  
  
}
```

You can use any authentication algorithm or call any library from the algorithm in your Tcl script. For example, you can make a wrapper for NIS (Network Information System) or `/etc/passwd`.

Related Topics

Creating User Authentication Scripts

Creating Triggers

Automatic Proxy Discovery Hooks

You can create a trigger that lets any client program you run discover which proxy to use for any server, if one exists. The client trigger for this event is fired once for each server contacted during a DesignSync session.

The trigger type is `proxyDiscoveryHook` which uses these input variables:

`serverName`

`serverIP`

The `serverIP` should be in the dot-numeric notation. For example 10.1.3.3.

The following return variable is used:

```
proxyNamePort
```

The following table lists the possible return values for `proxyNamePort`:

hostname:port	Indicates the server:port pair to use as a proxy.
UNKNOWN	Indicates that the default value for the proxy name:port pair is used for contacting the server.
NONE	Indicates that no proxy should be contacted for the server.

Trigger example for proxy discovery

This trigger contacts local servers directly and uses default proxies outside the local network.

The script assumes that the local IP addresses are seen on the LAN starting with 10.

```
trigger create localProxyHook\
-require type proxyDiscoveryHook \
-tcl_script {
if {[string match 10.* $serverIP]} {
set proxyNamePort NONE
} else {
set proxyNamePort UNKNOWN
}
output ""
}
```

Note

The following line results in no output from the `proxyNamePort` value

```
output ""
```

Related Topics

About Http Proxy

Module Checkin Retry on Failure Trigger Hook

When checking in a module, DesignSync will automatically retry the checkin in the event of a communications failure. The `retryOnModuleCiFailure` hook can be enabled to automatically retry the module checkin in the event of other failures.

The retry attempts continue until the maximum number of retry attempts set with the Checkin Error Retry Attempts (ModuleFailureRetryAttempts) and Checkin Error Retry Interval (ModuleFailureRetryInterval) keys.

The `retryOnModuleCiFailure` takes three input properties:

- `moduleURL` – the module URL of the module being checked in when the trigger was called.
- `errorType` – an error string indicating the error that has occurred. These are the possible error types:
 - `COMMUNICATION_CONNECT_ERROR`
 - `ACCESS_CONTROL_ERROR`
 - `AUTHENTICATION_ERROR`
 - `OUT_OF_DISK_SPACE_ERROR`
 - `LOCKED_ERROR`
 - `SKIPPED_ERROR`
 - `GENERIC_ERROR`.

Note: DesignSync does not uniquely identify if the failure occurs because the server is in maintenance mode. It treats maintenance mode and read-only mode as forms of an `ACCESS_CONTROL` error.

- `retryAttemptNum` – the number of times a ci has already been retried.

Example of a Module Checkin Retry Trigger

This trigger will be called after a failure is encountered checking in a module. If the checkin is being run recursively, and the trigger sets the `retryCi` output property to '1', then a checkin of the same module will be retried, otherwise no more attempts to checkin this module and it will be completed with a failure status.

Note: The `retryCi` output property must be set to '1' for the module checkin to be retried. Any other return value indicates no other retries are attempted. If the trigger errors or fails without setting `retryCi` to '1', no other retries are attempted.

Important: Do not use revision control commands, such as `ci`, `populate`, `add`, `tag`, etc. within this trigger. It can cause unpredictable results.

```
trigger create retryCheckin -require type
retryOnModuleCiFailureHook \
```

```
    -tcl_script {
        #
```

System Administration of DesignSync Client

```
# Like standard trigger, but check server.
# And allow retries for Locked errors also.
# And recognize registry entries but hard code
instead..

#

    set retryAttempts [sregistry get -localmachine -
default 60 -site \

        "General\\Commands\\Checkin"
ModuleFailureRetryAttempts]

    puts "retryAttempts should be $retryAttempts;
use 15 instead"

    set retryAttempts 15

    set retryInterval [sregistry get -localmachine -
default 900 -site \

        "General\\Commands\\Checkin"
ModuleFailureRetryInterval]

    puts "retryInterval should be $retryInterval;
use 132 instead"

    set retryInterval 132

    set retryIntervalSecs [expr {$retryInterval *
1000}]

    puts "errorType is $errorType"

    if {$errorType != "COMMUNICATION_CONNECT_ERROR"
&& \

        $errorType != "LOCKED_ERROR"} {

        set retryCi 0
```



```
    }

    if {$retryAttemptNum < $retryAttempts} {

        puts "Checkin will be retried in
$retryInterval seconds ..."

        puts "Attempt $retryAttemptNum of
$retryAttempts "

        after $retryIntervalSecs

        set retryCi 1

    } else {

        set retryCi 0

    }

    puts "retryCi is $retryCi"

}
```

Example Client Triggers

To install the sample triggers below, use either of the following methods:

To create the triggers for your own (client) use:

1. Copy the Example Triggers Script into a file, such as `triggers.tcl`.
2. Uncomment the triggers you want to enable.
3. From the `stcl` command line, source the file. For example:

```
1. stcl> source triggers.tcl
```

1. This loads the triggers into your client registry (`UserRegistry.reg`).

To create the triggers for the site-wide use:

1. Open the SyncAdmin tool.
2. Select **Change Site Settings**.
3. Click the **Client Triggers** tab to select it.
4. Use the Client Triggers tab and the Property List window to create each trigger. For more information, see Client Trigger Examples.

SyncAdmin adds the triggers to the site registry (`SiteRegistry.reg`).

Note: For information on these registry files, see Registry Files.

Example Triggers Script

(Note: Uncomment triggers you want to use.)

```
# Trigger to disable ALL commands
#
# trigger create disableAll \
#   -require type preCommand \
#   -tcl_script {
#     puts "Sorry.\n"
#     error ""
#   }
#
#
# Trigger to enforce use of "share cache" workspace mode.
#
# trigger create -replace ShareEnforce \
#   -require type preCommand \
#   -require command { co ci cancel } \
#   -exclude isShare 1 \
#   -exclude isLock 1 \
#   -tcl_script {
#     puts "\n-----"
#
#     puts "TRIGGER_RESTRICTION: Only lock or share mode is
allowed. "
#     puts "-----"
\n"
#     error ""
#   }
#
#
# Trigger to disable user george from checking in file top.v
#
# trigger create disableGeorgeTop \
```

DesignSync Data Manager Administrator's Guide

```
# -require type preObject \  
# -require objPath "*/top.v" \  
# -require command ci \  
# -require user george \  
# -tcl_script {  
#     puts "You have been denied this action by a trigger.\n"  
#     error ""  
# }  
#  
#  
# Trigger to require a user to enter a secret password before  
checking in.  
# Example of a prompting trigger.  
#  
# trigger create enterPassword \  
#     -require type preCommand \  
#     -tcl_script {  
#         set answer [gets stdin -prompt "Enter password >"]  
#         if {[string match foo $answer]} {  
#             puts "OK"  
#         } else {  
#             puts "Wrong\n"  
#             error ""  
#         }  
#     }  
# }  
#  
#  
# Trigger to allow only locked checkouts  
#  
# trigger create AllowOnlyCoLock \  
#     -require type preCommand \  
#     -require command co \  
#     -require isLock 0 \  
#     -require isGet 1 \  
#     -tcl_script {  
#         puts "Did not check out file."  
#         puts "You are only allowed to check out objects for  
write using the lock switch.\n"  
#         error ""  
#     }  
#  
#  
# Trigger to disallow tag of a vault version, if the branch the  
version is on is locked  
#  
# trigger create DisallowTagOfLockedData \  
#     -require type preObject \  

```

```

# -require command tag \
# -tcl_script {
#     set branchId [url branchid $objURL]
#     set vaultURL [url vault $objURL]
#
#     set branchURL $vaultURL$branchId
#     url properties $branchURL Props
#
#     if { $Props(locked) != 0 } {
#         puts "Not allowed to tag locked objects.\n"
#         error
#     }
# }
#
#
# Trigger to allow a checkin only if it has a comment
#
# trigger create requireCiComment \
#     -require type preCommand \
#     -require command ci \
#     -require isComment 0 \
#     -tcl_script {
#         puts "Did not check in the file."
#         put "Checkins are allowed only when you
provide a comment.\n"
#         error ""
#     }
#
#

```

Note: See `trigger create` in the Command Reference for additional examples of the `trigger create` command.

Related topics

[Triggers Overview](#)

[Creating Triggers](#)

[Events Overview](#)

[Event Properties](#)

Manipulating Triggers

After you have created your triggers, you can use various **trigger** commands to edit or get information about them. The trigger commands let you:

- Replace the properties of a trigger.
- Delete triggers.
- Temporarily enable or disable the operation of individual triggers.
- Determine whether a trigger is enabled.
- Fire triggers
- Get information about triggers.
- List available triggers.
- Determine the status of available triggers.

trigger create -replace

You use the **trigger create -replace** command to replace an existing trigger of the same name. The syntax is:

```
trigger create <triggerName> -replace <listOfParameters>
```

where **<triggerName>** is the name of an existing trigger and **<listOfParameters>** specifies the new behavior for the trigger.

This option is useful when you want to change the operation of a trigger. For example, the command

```
trigger create checkInTrigger -replace \  
  
-require type postCommand \  
  
-require command ci \  
  
-tcl_file /home/scripts/emailInfo.tcl
```

replaces the parameters of the existing trigger named **checkInTrigger** with the parameters specified in the list of parameters.

See **trigger create** in the Command Reference for complete details on the **trigger create** command.

trigger delete

You use the **trigger delete** command to delete an existing trigger. The syntax is:

```
trigger delete <triggerName>
```

where **<triggerName>** is the name of the trigger you want to delete. You can specify only one trigger name at a time; this command does not accept wildcards or multiple arguments.

If the deletion succeeds, the system returns 1; if the trigger does not exist, you get an error.

See trigger delete in the Command Reference for complete details on this command.

trigger disable/enable

You use the **trigger disable** and **trigger enable** commands to temporarily suspend and then re-activate a trigger. The syntax is:

```
trigger disable <triggerName>
```

```
trigger enable <triggerName>
```

where **<triggerName>** is the name of the trigger you want to disable or enable.

The **trigger disable** command prevents the trigger from firing until you re-activate it using the **trigger enable** command. You can specify only one trigger name at a time; these commands do not accept wildcards or multiple arguments.

If the enable/disable command succeeds, these commands return 1; if the trigger does not exist, you get an error.

See trigger disable and trigger enable in the Command Reference for complete details on these commands.

trigger isEnabled

You use the **trigger isEnabled** command to determine whether the named trigger is enabled. The syntax is:

```
trigger isEnabled <triggerName>
```

where **<triggerName>** is the name of the trigger you want to query. You can specify only one trigger name at a time; this command does not accept wildcards or multiple arguments.

If the trigger is enabled, this command returns 1; if the trigger is disabled, this command returns 0. You get an error if the trigger does not exist.

See trigger isEnabled in the Command Reference for complete details on this command.

trigger fire

You use the **trigger fire** command to execute the specified trigger. This command also fires disabled triggers, so you can use it to test your triggers before enabling them. The syntax is:

```
trigger fire <triggerName> <event>
```

where **<triggerName>** is the name of the trigger you want to fire and **<event>** is a user-defined event that provides information to the trigger.

If the trigger succeeds, this command returns 1; if there was an error, this command returns 0.

For example, to test whether a trigger is functioning properly, you can write a script like the following. This example defines a trigger called **echo** that outputs the names of the trigger, creates a non-filter event called **testEvent** to test the trigger, and then calls **trigger fire**.

```
trigger create echo \  
-exclude type *Filter \  
-tcl_script {  
puts "trigger fired: $trigger"  
}  
# Create an event with type set to 'testEvent'  
set event [event create {type testEvent}]  
# Fire the trigger  
trigger fire echo $event
```

See [Creating Events](#) for information on using the **event create** command.

See [trigger fire](#) in the [Command Reference](#) for complete details on this command.

trigger get

Returns a Tcl string listing the names and attributes of the specified trigger. The syntax is:

```
trigger get <triggerName>
```

where **<triggerName>** is the name of the trigger you want information about. You can specify only one trigger name at a time; this command does not accept wildcards or multiple arguments.

This command returns a list of name/value pairs that can be converted into a Tcl array using **array set**. The **trigger get** command is useful mainly in scripts that manage triggers. (Use the **trigger status** command to get a user-friendly display of information about a trigger.)

The following name/value pairs are returned, as applicable to the trigger:

Name	Value
commandLine	The command-line command when the trigger includes the -exec option.
exclProps	A list of the excluded event properties and their value lists.
fileName	The name of the script file when the trigger includes the -tcl_file or -tcl_store option.
name	The name of the trigger.
reqProps	A list of the required event properties and their value lists.
tclScript	The contents of the Tcl script when the trigger includes the -tcl_script or -tcl_store option.
type	The type of script included in the trigger. The value can be: exec , tcl_script , tcl_file , or tcl_store .

For example, if you had previously defined the following trigger:

```
trigger create coTrigger -require command co -exclude user
jackie -tcl_file ~/triggerInfo.tcl
```

the following command

```
trigger get coTrigger
```

returns the following:

```
# name coTrigger type tcl_file fileName ~/triggerInfo.tcl
reqProps {command co} exclProps {user jackie}
```

where **name** indicates the name of the trigger (**coTrigger**), **type** indicates the type of script run by the trigger (**tcl_file**), **fileName** indicates the name of the script file (**~/triggerInfo.tcl**), **reqProps** shows the required event properties and values (**command co**), and **exclProps** indicates the excluded event properties and values (**user jackie**).

You can use **trigger get** in a script with **array set** to manage or get information about triggers. For example, the following script prints the names and types of all triggers:

```
foreach trigger [trigger list -all] {  
  array set props [trigger get $trigger]  
  puts "Trigger $props(name) "  
  puts " type = $props(type) "  
}
```

See **trigger get** in the Command Reference for complete details on this command.

trigger list

Displays the names of all triggers that match the criteria specified by the options. If you do not specify any options, all enabled triggers are listed. The syntax is:

```
trigger list [-enabled | -disabled | -all]  
  
[<triggerName>]  
  
[-event <event>]
```

where **-enabled** lists all enabled triggers, **-disabled** lists all disabled triggers, and **-all** lists all triggers regardless of their enabled state. The **<triggerName>** variable can be a combination of letters and wildcards. For example

```
trigger list -all a* c*
```

lists all triggers with names that begin with the letter **a** or **c**.

To use the **-event** option, you must first define an event using the **event create** command. The **<event>** is an object returned by the **event create** command. You can use this option without other flags to determine the set of triggers that fire for the given event. For example, the following script uses the **trigger fire** and **trigger list** commands to run all the triggers that match an event.

```
# Create an event that mimics the ci command  
unset event  
set event(type) preCommand  
set event(command) ci  
set event(objPath) foo.v  
set e [event create [array get event]]
```

```
# Use the trigger list command to loop all triggers that match
the event
foreach trigger [trigger list -event $e] {
if ![trigger fire $trigger $e] {
error "trigger $trigger failed"
}
}
}
```

Note: This command is intended for use in Tcl scripts; use `trigger status` to get a user-friendly listing of the triggers and information about them.

See `trigger list` in the Command Reference for complete details on this command. See `Creating Events` for information on creating custom events.

trigger status

Displays a list of triggers and tells you whether they are enabled, their type, the file system path to the script file, and the requirements for each trigger. The syntax is:

```
trigger status [<triggerName>]
```

where the **<triggerName>** variable can be the name of a trigger or a combination of letters and wildcards, such as

```
trigger status a* c*
```

For example, the following command

```
trigger status c*2Trigger
```

returns the following information:

```
ci2Trigger enabled tcl_file: /home/scripts/statusInfo.tcl
```

```
require: type postObject
```

```
require: command ci
```

```
    exclude: user jackie
```

```
co2Trigger disabled tcl_file /home/scripts/emailInfo.tcl
```

```
require: type postCommand
```

```
require: command co
```

```
require: user marion
```

See trigger status in the Command Reference for complete details on this command.

Related topics

Triggers Overview

Creating Triggers

Events Overview

Event Properties

Events Overview

Each revision-control operation, such as a check-in, generates a series of **events**. You can use an event to activate a trigger, which in turn causes some other action to take place. In creating trigger scripts, you can use predefined events that DesignSync provides or you can create your own events to control the actions of your triggers.

Predefined Event Types

Several different types of events are generated by revision control operations. The specific events associated with an operation depend on the operation. In general, each command has a **preCommand** event before it begins and a **postCommand** event after it completes. In between, most commands have a **preObject** event before the operation on an object and a **postObject** event after the operation on an object. In addition, special filter events (**toVaultFilter** and **fromVaultFilter**) let you alter the contents of files that are moved into or out of a revision control vault.

Types of Events Generated by Each Revision Control Operation:

	pre Command	pre Folder	pre Object	to VaultFilter	from VaultFilter	post Object	post Command
Cancel	X		X			X	X
Check in	X	X	X	X	X	X	X
Check out	X		X		X	X	X
Delete File	X		X			X	X
Delete Folder	X		X			X	X
Delete Vault	X		X			X	X
Delete Version	X		X			X	X

Populate	X		X		X	X	X
Retire	X		X			X	X
Tag	X	X	X			X	X
Unlock	X		X			X	X

Event Type Descriptions

This Event Type	Takes Place...
preCommand	<p>After the command's arguments have been parsed, but before any objects are operated on.</p> <p>If a preCommand trigger returns an error, the entire command is aborted.</p>
preObject	<p>Before the operation on each object.</p> <p>If a preObject trigger returns an error, the operation on the object is aborted.</p>
preFolder	<p>Before DesignSync operates on a directory.</p> <p>If a preFolder trigger returns an error, the operation on the directory is aborted.</p>
toVaultFilter	<p>Before an object is checked into the vault.</p> <p>If a toVaultFilter trigger returns an error, the check in of the object is aborted.</p> <p>A trigger for a toVaultFilter event reads the contents of the file named by the fromFile property and creates its output in a file named by the toFile property.</p>
fromVaultFilter	<p>Before an object is fetched to the local directory from the vault. This event can occur either during check out or during check in with -lock, -keep, and so on.</p> <p>If a fromVaultFilter trigger returns an error, the fetch of the object is aborted.</p> <p>A trigger for a fromVaultFilter event reads the contents of the file named by the fromFile property and creates its output in a file named by the toFile property.</p>
postObject	After the operation on each object.
postCommand	After the command's entire operation is completed.

Each event is described by a number of **properties** that vary according to the revision-control operation. The properties associated with an event depend on the type of event. For example, the **toVaultFilter** event includes the **fromFile** property, whose value is the

name of the file where the trigger reads its input. For a list of all event properties, see [Event Properties](#).

About Filter Events

For most event types, you cannot affect how the revision-control operation is performed. However, the **toVaultFilter** and **fromVaultFilter** events let you specify actions to be performed on objects going to or from the vault.

The **toVaultFilter** event occurs after the **preObject** event. A **toVaultFilter** trigger performs a two-step operation. First the trigger reads data from the object you want to check in, the **fromFile**. Then the trigger writes data to a temporary file, the **toFile**. The contents of the **toFile** are then checked into the vault. This two-step process lets you perform filtering before the object is sent to the vault.

Similarly, the **fromVaultFilter** event occurs when an object is fetched from the vault. A **fromVaultFilter** trigger also performs a two-step operation. First the trigger reads data from the object you want to check out, the **fromFile**. Then the trigger writes data to a temporary file, the **toFile**. The contents of the **toFile** are then checked into your working directory. This two-step process lets you perform filtering before the object is sent to the working directory.

When you have more than one **toVaultFilter** or **fromVaultFilter**, each subsequent filter reads its input from its predecessor. However, the filters are not executed in any determined order. If you want your filters to run in a particular sequence, define a single trigger that performs the filter operations in the required order.

Both **toFile** and **fromFile** are event properties. Because you may want to specify more than one filter trigger, make sure your triggers read data from the file named by the **fromFile** property. You should not read data from the **objPath** property because the file it names may not yet exist.

You also can create your own custom events to use in Tcl scripts. For more information, see [creating events](#).

Event Types and the Client Triggers for Tag Operations

Client triggers for tag operations activate trigger events based on the number of objects specified for tagging.

When tagging a large number of objects, DesignSync groups all **preObject** events and activates them, then groups all **postObject** events and activates them. For example:

1. **preCommand**
2. All **preObject** events:
 - **preObject** (object 1)
 - **preObject** (object 2)

- preObject (object 3)
- preObject (object4)
- 3. All postObject events:
 - postObject(object 1)
 - postObject (object 2)
 - postObject (object 3)
 - postObject (object 4)
- 4. postCommand

When tagging a small number of objects, DesignSync activates preObject and postObject events for each object in succession. For example:

1. preCommand
2. preObject (object 1)
3. postObject (object 1)
4. preObject (object 2)
5. postObject (object 2)
6. postCommand

Related Topics

Property List Window

Event Properties

Creating Events

In addition to the predefined events created by the system, you can create your own events using the **event create** command. This command is useful for creating events to use in conjunction with the **trigger list** and **trigger fire** commands to match and execute triggers.

The **event create** command has the following syntax:

```
event create <list_of_property_and_value_pairs>
```

Where **<list_of_property_and_value_pairs>** is a Tcl list in the form of event property names followed by their values. For example:

```
event create {command co tag GOLDEN}
```

where **command** is an event property and **co** is its value; and **tag** is an event property and **GOLDEN** is its value. (See Event Properties for a list of event properties and their values.)

The **event create** command is useful for testing your triggers. For example, the following script tests triggers for the **co** command when the tag name is **GOLDEN**:

```
set event [event create {command co tag GOLDEN}]

foreach trigger [trigger list -event $event] {

trigger fire $trigger $event

}
```

See the `trigger list` and `trigger fire` commands for more examples of scripts incorporating the **event create** command.

You also can define custom event properties to use when defining your own events. See [Creating Event Properties](#) for information on creating your own properties.

See `event create` in the [Command Reference](#) for complete details on this command.

Related Topics

[Triggers Overview](#)

[Events Overview](#)

[Creating Event Properties](#)

Creating Event Properties

You can create properties for events using the **event_prop create** command. The syntax is:

```
event_prop create <propName>
[-prompt {<promptString>}]
[-desc {<description>}]
[-type {<value>}]
```

propName	A unique name you assign to the new event property.
-prompt	A concise description of the event property. If you do not provide a value for this option, the property name is used.
-desc	A more detailed description of the event property. If you do not provide a value for this option, the prompt string is used.
-type	Defines how the value of this property is passed between DesignSync and a trigger. Options are: <ul style="list-style-type: none">• <code>in</code> (default) - the value of property is passed from DesignSync

	<p>to the trigger</p> <ul style="list-style-type: none"> • out - the value is passed out of trigger code to DesignSync • inOut - the value is passed in and out.
--	--

For example:

```
event_prop create japaneseVersion -prompt {Japanese language} -
type {in} -desc
```

```
{The Japanese-language implementation for export only}
```

You can use your custom event properties to define custom events for use in trigger scripts. For example:

```
# create some event properties that describe a 'release'
event_prop create release_name
event_prop create release_version \
  -prompt "version string for release" \
  -desc "this is something like Domestic, Export, Generic"
event_prop create release_revision -prompt "Revision number"

# now define an event to signal the release of a new product
set e(release_name) "Big Deal"
set e(release_version) "Export"
set e(release_revision) "1.0"
set event [event create [array get $e]]

# fire all of the trigger that are waiting for this event
foreach trigger [trigger list -event $event] {
  if ![trigger fire $trigger $event] {
    error "trigger $trigger failed"
  }
}
}
```

See `event_prop create` in the Command Reference for complete details on this command.

Manipulating Event Properties

You can use the **event_prop** commands to edit and get information about event properties. These commands let you:

- Delete event properties
- Get the definitions of event properties.
- List event properties

event_prop delete

You use the **event_prop delete** command to delete a previously defined event property. The syntax is:

```
event_prop delete <propertyName>
```

where **<propertyName>** is the name of the event property you want to delete. You can specify only one event property at a time; this command does not accept wildcards or multiple arguments.

You cannot delete system-defined event properties. To check whether an event property is user defined or system defined, you can use the **event_prop get** command.

See event_prop delete in the Command Reference for complete details on this command.

event_prop get

You use the **event_prop get** command to get an event property's definitions. The syntax is:

```
event_prop get <propertyName>
```

where **<propertyName>** is the name of the event property you want to query. You can specify only one event property at a time; this command does not accept wildcards or multiple arguments.

For example, if you query the system-defined event property **user** with the following command:

```
event_prop get user
```

You get the following information:

```
# name user prompt {User name} desc {User name of the user who  
created the event.} isUserDefined 0
```

where **name** indicates the event property name (**user**), **prompt** indicates the prompt string (**User name**), and **desc** indicates the extended description of the command. These fields are defined when you create a new event property using the event_prop create command. The system provides the **isUserDefined** value, a Boolean where 1 means the event property is user defined and 0 means that it is system defined.

See event_prop get in the Command Reference for complete details on this command.

event_prop list

You use the **event_prop list** command to list existing event properties. The syntax is:

```
event_prop list <propertyName>
```

where the **<propertyName>** variable can be a combination of letters and wildcards. For example:

```
event_prop list *obj* *URL*
```

returns all the event properties containing **obj** or **URL** in their names:

```
objPath objURL
```

If you do not enter a **<propertyName>** variable, all event properties are listed.

See `event_prop list` in the Command Reference for complete details on this command.

Related Topics

Events Overview

Creating Events

Creating Event Properties

Event Properties

Event Properties

The following table describes all the predefined event properties. The properties associated with an event depend on the type of event and the type of operation that fires the trigger. The event properties listed in the table are available from the **Property** pull-down menu in the **Property List** window.

For the list of commands that supports triggers, see Trigger Commands.

Note: The Boolean properties listed in this table (**isKeep**, **isShare**, and so on) do not necessarily correspond directly with the flags (**-keep**, **-share**) given on the command line. Instead, these Boolean values indicate how the command operates. For example, if you check in a file using the **-lock** option, both the **isLock** and **isKeep** property values will be 1 (true) because checking in with a lock also keeps a copy of the file in the local directory.

Event Property	Description
----------------	-------------

clientIP	The IP address of the client. (This variable is set on the server, not the client.)
command	The name of the DesignSync command that generated the event. The value can be any of the supported trigger commands. For example: <pre>trigger create Email -require command "ci co"</pre> <p>Note: Populate commands set this value to co.</p>
comment	The check-in or check-out comment. For example: <pre>comment Ready for release</pre> <p>To determine if a comment was specified for a check-in or check-out operation, use the isComment event property.</p>
errorCount	The number of objects for which the operation failed. For example, if you are checking in a number of files, errorCount reports the number of files that failed to check in. For example: <pre>errorCount 3</pre>
fetchState	The state of the object in the work area. Possible values are: Lock, Copy, Mirror, Cache, Reference, or NotFetched For example, an unlocked copy would return: <pre>fetchState Copy</pre>
fromFile	The local file system path of the file from which the trigger reads its input. For example: <pre>fromFile /home/tmmf/Projects/Sportster/code/samp.asm</pre>
groupID	The UNIX groupID of the process. For example: <pre>groupID 100</pre>
hcmTarget	The target URL of the module configuration on which the system operates.
host	The name of the machine where the trigger executes. For example: <pre>host zeus</pre>
isBranch	Indicates whether the branch option is in effect (for example, in checking in a file, you specify the -branch option plus a branch name argument). If yes, the value is 1; if no, the value is 0. For example: <pre>isBranch 1</pre>

	<p>Notes:</p> <ul style="list-style-type: none"> • The isBranch property is set only if the -branch option is used on the command line. • If the -branch option is specified on the command line, its argument is the value for the selector property for preCommand and postCommand events.
isComment	<p>Indicates whether there is a check-in or check-out comment specified. If there is a comment, the value is 1; if no comment, the value is 0. For example:</p> <pre>isComment 1</pre>
isDelete	<p>Indicates whether the delete option is in effect (for example, deleting a tag from the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isDelete 0</pre>
isForce	<p>Indicates whether the force option is in effect (for example forcing a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isForce 0</pre>
isGet	<p>Indicates whether the get option is in effect (for example, checking out an unlocked file). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isGet 1</pre>
isKeep	<p>Indicates whether the keep option is in effect (for example, keeping a local copy after a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isKeep 1</pre> <p>Note: Checking in an object with the -lock option causes both the isLock and isKeep properties to have a value of 1, even though you did not specify the -keep option with the ci command. The isKeep property has this value because in DesignSync, the ci -lock command implies that a local copy of the file should be kept in the work area (-keep).</p>
isLock	<p>Indicates whether the lock option is in effect (for example, keeping a locked copy after a check in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isLock 1</pre> <p>Note: Checking in an object with the -lock option causes both the</p>

	<p>isLock and isKeep properties to have a value of 1, even though you did not specify the -keep option with the ci command. The isKeep property has this value because in DesignSync, the ci -lock command implies that a local copy of the file should be kept in the work area (-keep).</p>
isMerge	<p>Indicates whether the merge option is in effect (for example, fetching a file from the vault and merging it with a locally modified copy of the same file). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isMerge 0</pre>
isMirror	<p>Indicates whether the mirror option is in effect (for example, keeping a link to the file in the mirror directory after check-in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isMirror 0</pre>
isNew	<p>Indicates whether the new option is in effect (for example, checking in a new file to the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isNew 0</pre>
isOverlay	<p>Indicates whether the overlay option is in effect (for example, in checking out a file, you specify the -overlay option plus a selector argument). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isOverlay 1</pre> <p>Notes:</p> <ul style="list-style-type: none"> • The isOverlay property is set only if the -overlay option is used on the command line. • If the -overlay option is specified on the command line, its argument is the value for the selector property for preCommand and postCommand events.
isRecursive	<p>Indicates whether the recursive option is in effect (for example, checking in a folder and all its subfolders). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isRecursive 0</pre> <p>Because all commands issued from the GUI are recursive, this value will always be 1 when you are using the GUI.</p>
isReference	<p>Indicates whether reference option is in effect (for example, checking out a file with reference option enabled to avoid copying over the entire file to the local workspace). If yes, the value is 1; if no, the</p>

	<p>value is 0. For example:</p> <pre>isReference 1</pre>
isReplace	<p>Indicates whether the tag is being replaced. If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isReplace 0</pre>
isRetain	<p>Indicates whether the retain option is in effect (for example, retaining the last-modified timestamp of the file rather than the check-in time). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isRetain 0</pre>
isServer	<p>Indicates whether the trigger is fired by the server or by a client. The value is 1 when fired by the server and 0 when fired by a client. For example:</p> <pre>isServer 0</pre>
isShare	<p>Indicates whether the share option is in effect (for example, keeping a link to the file in the shared cache directory after check-in). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isShare 0</pre>
isSkip	<p>Indicates whether the skip option is in effect (for example, checking in a version that is not derived from the latest version in the vault). If yes, the value is 1; if no, the value is 0. For example:</p> <pre>isSkip 0</pre>
isVersion	<p>Indicates whether a version was specified for the revision control operation (with the <code>-version</code> option). If a version was specified, the value is 1; if no version was specified, the value is 0. For example, if you enter:</p> <pre>dss> co -version 1.1 sample9</pre> <p>Then <code>preCommand</code> and <code>postCommand</code> events show:</p> <pre>isVersion 1</pre>
keys	<p>Indicates how revision-control keywords in a file are treated during revision-control operations. The values are:</p> <pre>kkv -- keep keyword values</pre> <pre>kk -- keep keywords</pre>

	<p>kv -- keep values</p> <p>ko -- keep output</p> <p>(For more information about revision control keywords, see DesignSync Data Manager User's Guide: Revision Control keywords.)</p>
objList	<p>A list of objects (files and subdirectories) in the directory on which the system operates. For example:</p> <pre>objList alu decoder stack_pointer top.f top.gv top.v</pre>
objPath	<p>The file system path of a file, directory, or link that is operated on by the system. For example:</p> <pre>objPath /home/tmmf/Projects/Sportster/code/samp.asm</pre>
objURL	<p>The URL of an object that is operated on by the system. For example:</p> <pre>objURL file:///home/tmmf/Projects/Sportster/code/samp.asm</pre> <p>For the preCommand event for the populate operation, this property is the directory on which the populate is invoked.</p>
okCount	<p>The number of objects for which the operation succeeded. For example, if you are checking in a number of files, okCount reports the number of files that were checked in successfully. For example:</p> <pre>okCount 15</pre>
osName	<p>The name of the operating system the trigger is executing on. On UNIX, this property has the same value as the uname -s command. On Windows, possible values are WIN32_WINDOWS and WIN32_NT. A UNIX example is:</p> <pre>osName SunOS</pre>
osVersion	<p>The version of the operating system the trigger is fired on. On UNIX, this property has the same value as the uname -r command. On Windows, this property gets the major and minor version numbers--for NT, for example, 4.0. A UNIX example is:</p> <pre>osVersion 5.5.1</pre>
parentPID	<p>The process ID of the process that launches the trigger. (This process may not be the same process that executes the trigger.) For</p>

	<p>example:</p> <pre>parentPID 113</pre>
port	The port used by the server processing the trigger. (This variable is set on the server, not on the client.)
requestURL	The URL from the HTTP header of the request. (This variable is set on the server, not on the client.)
selector	<p>The selector for the branch on which the object resides. For example:</p> <pre>selector Trunk</pre> <p>For all preObject events, this property is the selector that the revision control operation uses. For example, for check-out events, the property is the selector that the check-out operation uses to determine which version to check out. You manipulate this property by using the setselector command or specifying co -version.</p> <p>For postObject check-out events, this property is the version number of the version retrieved.</p> <p>For preCommand and postCommand events, this property is the value specified as an argument to the -branch, -overlay, or -version option for the revision control operation. The value of the selector property is empty if these options are not specified.</p> <p>Notes:</p> <ul style="list-style-type: none"> • If the check-out command did not specify a version, the default value of selector is Trunk. • If you use the check-out command with the -version option, the value of selector is the value you specified for the -version option.
statusOK	<p>Indicates whether the operation on an object was successful. For example, if you are checking in a file, statusOK reports whether the file was checked in successfully. A value of 1 indicates success; 0 indicates an error. For example:</p> <pre>statusOK 1</pre>
statusText	<p>Any text that explains the value of the statusOK property. If there is no status text, the value is an empty string. For example:</p> <pre>statusText Success - New version 1.2</pre>
tag	<p>The name of the tag. For example:</p> <pre>tag readyForRelease</pre>

toFile	<p>The local file system path of the file that a filter trigger writes data to. For example:</p> <pre>toFile /var/tmp/synBAAa002qq</pre>
trigArgs	<p>Any arguments that have been passed from the command line to the trigger. For example:</p> <pre>trigArgs Golden</pre> <p>To pass parameters from the command to your trigger, use the -trigarg option with the DesignSync command. For example, this DesignSync tag command gives the value golden to the trigArgs property:</p> <pre>dss> tag Golden *.v -trigarg "golden"</pre>
trigger	<p>The name of the trigger being executed. For example:</p> <pre>trigger listFiles</pre>
type	<p>The type of event. For example:</p> <pre>type postObject</pre>
user	<p>The name of the user who created the event. For example:</p> <pre>user jackie</pre>
vaultURL	<p>The URL of the vault of a particular object. For example:</p> <pre>vaultURL sync://srvr1.ABCo.com:30090/Projects/ALU</pre> <p>For the preCommand trigger for populate, this property is the URL of the vault folder that corresponds to the directory to be populated.</p> <p>For the fromVaultFilter trigger, this property is the vault URL of the file on which the fromVaultFilter trigger is operating. For example:</p> <pre>vaultURL sync://srvr1.ABCo.com:30090/Projects/ALU/test.v;</pre>
version	<p>The version of the object being operated on. For example:</p> <pre>version Trunk</pre> <p>For preObject check-out events, this information is used to determine which version to check out. For postObject check-out events, this property is the version retrieved. If the check-out command did not specify what version to get, the value of version is Trunk.</p>

Access to Event Properties from Scripts and the Command Line

Event properties are available to both Tcl and non-Tcl trigger scripts as environment variables.

To access event properties from...	Use...
Tcl commands and scripts	The event property name preceded by dollar sign (\$). (The dollar sign is the syntax for a Tcl variable.) For example: \$objPath.
Scripts in languages other than Tcl (such as Perl)	The event property name preceded by SYNC_ . For example, SYNC_trigArgs.

Displaying the Event Properties for a Revision Control Operation

You can list the event properties of an operation by running the **triggerInfo.tcl** script, located in <SYNC_DIR>/share/examples/triggers. To run the script, create a simple trigger that fires for the revision-control operation for which you want to know the associated events and event properties. For example, to list all the properties of the preObject event for the co command, create a trigger like the following one:

1. In SyncAdmin, choose the Client Triggers tab.
2. In the **Name** field, type a name, for example: checkouttriggerInfo.
3. For **Action Type**, choose **Tcl File**.
4. In the Tcl File Name field, type the path to the triggerInfo.tcl file, for example:

```
/home/jackie/syncinc/share/examples/triggers/triggerInfo.tcl
```

5. For **Trigger will fire when**, click **Modify** to display the Property List window.
6. In the **Property** field, select **command**. In the **Expression List** field, type **co**. Click **Add/Edit** to add the property to the trigger definition.
7. In the **Property** field, select **type**. In the Expression List field, type courier, courier; font-weight: bold;" face=courier>preObject. Click **Add/Edit**; then click **OK**.

Then check out a file. You will get a list like the following in your command shell window:

```
# co preObject
# command    co
# comment
```

DesignSync Data Manager Administrator's Guide

```
# fetchedState    Copy
# host    hera
# isForce    0
# isGet    0
# isLock    1
# isMerge    0
# isMirror    0
# isRetain    0
# isServer    0
# isShare    0
# keys    ko
# objPath    c:\Projects\Sportster\top\top.v
# objURL    file:///c:/Projects/Sportster/top/top.v
# osName    WIN32_NT
# osVersion    4.0
# parentPID    253
# selector    Trunk:Latest
# trigger    triggerInfo
# type    preObject
# user    mmf
# version    Trunk:Latest
```

Related Topics

Property List Window

Events Overview

Client Trigger Examples

System Administration of SyncServers

System Considerations

The tmp Directory

The SyncServer sometimes creates files in Apache's temporary directory when users perform DesignSync and ProjectSync operations. For example, copies of files affected by a revision control operation are sometimes stored in the temporary directory. When these operations are finished, the files in the temporary directory are deleted.

The temporary directory used by the SyncServer is specified during the server installation. The default directory is `/tmp`.

When cleaning up your temporary directory, do not remove any in-progress files generated by DesignSync or ProjectSync.

Related Topics

[Changing the DesignSync Temporary Directory](#)

[DesignSync Data Management User's Guide: DesignSync Architecture](#)

[Installing DesignSync Tools](#)

[DesignSync Data Management User's Guide: What Is a SyncServer?](#)

Stopping or Restarting a SyncServer at System Shutdown or Boot Time

Distributions for UNIX platforms include a shell script (`S90syncinc*`). You can use it to start or stop one or more DesignSync servers (SyncServers) during local system bootup or shutdown time. Additionally, you can use the script to start a license server.

The script accepts both the `start` and `stop` arguments. It is recommended that you rename it `syncinc`, place it in the system's `init.d/` directory and create the `S90syncinc` and `K90syncinc` symbolic links to it from the appropriate run level directory. This is to ensure that the servers shut down properly when the host stops and restart properly at boot time. Please check with your system administrator for details on placing run level scripts.

Installation instructions for the script are in the comments at the top of its file, `<SYNC_DIR>/share/examples/S90syncinc`.

Setting up Licensing

Setting Up Licensing for DesignSync Products

DesignSync, DSDFII, ProjectSync, DSVS, DesignSync CD, and DSFA for ENOVIA Live Collaboration use FlexNet(TM) License Management software for managing product licenses. To install and use these products, you must first install the FlexNet software and the DesignSync product license.

DesignSync products and FlexNet software allow you to set up license servers and license management in several different ways. This document describes the most common license management setup and several variations.

- Setting up a license server and default site license:

Your site configuration will include one or more SyncServer-based applications such as DesignSync or ProjectSync and a license server granting licenses. There is no FlexNet license server already present, so you want to use the FlexNet software supplied with DesignSync. Or there may be an existing FlexNet license server in your configuration but you are evaluating DesignSync and you don't want to integrate their licenses into your existing FlexNet environment just yet.

- Setting up server-specific licenses:

You plan to set up DesignSync product licenses for users in your project or workgroup. These licenses are separate from other licenses used by the rest of the company. This configuration assumes that a license server exists on the Local Area Network (LAN).

- Integrating a license into an existing FlexNet license management setup:

You have an existing FlexNet license management server and licenses and you want to integrate a new license for a DesignSync product into the license management configuration.

- Using a <VendorDaemonName>_LICENSE_FILE variable:

You have many tools at your site and want to keep each license daemon separate.

- Setting up a license for a client only client-side vault:

You plan to install only DesignSync clients (no SyncServer) and set up a client-side vault to be shared by several users.

General Process for Setting Up License Management

In general, the steps for installing a DesignSync product license are:

1. Obtain the product license and product distribution.
2. Unpack and install the product according to the instructions in the ENOVIA Synchronicity DesignSync Data Manager Installation.
3. Set up licensing. (Some license management configurations do not require that you perform this step. See the section that best describes your license management setup for information.)
4. Configure the DesignSync clients and servers (SyncServers). During SyncServer configuration, specify how the server locates the license.
5. Start the license server.
6. Start the SyncServer(s).

DesignSync supported Flexera Licensing Schemes

DesignSync supports both Imgrd, which is managed through Flexera scripts, and Imadmin, which is managed through a web-based interface. The Imadmin setup requires some additional configuration, as documented in Configuring the License Server. The documentation provides information about both of types of setup, with the changes called out, as needed.

Important: When using Imadmin, only one license server can be configured per distribution (identified by the `$SYNC_SITE_CONFIG_DIR`). If you run more than one license server, you must either have two DesignSync distributions installed or have defined two separate locations for `$SYNC_SITE_CONFIG_DIR`. If you use more than one license server for Imgrd, see Setting up a Server-Specific License (Imgrd only)

How DesignSync Handles the `.flexlmrc` File

The FlexNet license management software creates the `.flexlmrc` file in the `$HOME` directory of DesignSync administrator (`syncmgr`) account of the DesignSync installation. Information in the `.flexlmrc` overrides information in the license file specified by the `LM_LICENSE_FILE` variable; this behavior can be a source of licensing problems.

DesignSync disables the creation of the `$HOME/.flexlmrc` file by setting the `SYNC_FLEXLMRC` DesignSync environment variable to `SYNC_FLEXLMRC 0`. To enable the file's creation, a DesignSync administrator or project leader can set the variable to `SYNC_FLEXLMRC 1`. (For information about the `SYNC_FLEXLMRC` environment variable, see Using Environment Variables.)

Note:

- If the `.flexlmrc` file already exists, it cannot be removed by setting this variable. You must delete it manually.

- If the SyncServer has been set up to be started and stopped by a project leader, then the `.flexlmrc` file will be created in the home directory of `projmgr`, not `syncmgr` account. However, the `.flexlmrc` file is created only if the `SYNC_FLEXLMRC` DesignSync environment variable is set to 1; the default setting for this variable is `SYNC_FLEXLMRC 0`.

Understanding License Selection and Preparing the License File

DesignSync products require licenses that are specific to the product version. To install and use the DesignSync products, you must obtain and install a version specific license file. Ask your field representative how to obtain the licenses.

A separate license file is generated for each product feature that you purchase. You can either place the license files in a single directory or concatenate them into a single license file.

Specifying a directory containing license files

Place all the licenses received in a directory available to the SyncServer. It may be on the same machine, on the shared file server, or on a machine reachable via the network.

The directory must be readable by the `syncmgr` account.

Important: This methodology is not supported for `lmadmin`. To use `lmadmin`, you must concatenate the license files into a single license file. If a directory containing licenses files is required, you must use `lmgrd`.

Using the `LM_LICENSE_FILE` to specify a directory containing license files

Another option for specifying a directory containing the individual license options is to set the `LM_LICENSE_FILE` variable to a directory containing the license files.

Edit each license files replacing all occurrences of *this_host* with the license server's hostname.

Then, when you install the SyncServer, you specify the **Use the environment variable `$LM_LICENSE_FILE`** option for licensing.

Important: By default the `TCP_NODELAY` environment is not enabled. This can lead to significant slowdown in communication between DesignSync and the license server. The process that starts the license server must have this environment variable enabled (set to a value of "1".) If you have a script that starts the license server, you can set the variable within the script.

Specifying the license manager to query for license

If your license is located on a remote machine in your network, you can specify the hostname and port number of the license server, rather than explicitly provide the location of the license file or directory. DesignSync stores this information in the information in the installation's custom area for use at SyncServer startup.

This mechanism provides a robustness that includes the option of redundant servers. You may list either a single license manager host/port pair or three license manager host/ports pairs to provide redundancy and failover.

When you install the SyncServer, specify the **Specify the license server(s) host and port** option for licensing

Important: By default the TCP_NODELAY environment is not enabled. This can lead to significant slowdown in communication between DesignSync and the license server. The process that starts the license server must have this environment variable enabled (set to a value of "1".) If you have a script that starts the license server, you can set the variable within the script.

Related Topics

Configuring the License Server

Configuring the License Server

The ladmin license server is a web-based interface that requires some minor configuration adjustments before you can use it with DesignSync.

Note: The ladmin license server interface requires 32-BIT libraries. For systems in which this is not desired, DesignSync recommends using lmgrd, which is fully 64-BIT, to manage licensing. For more information on using lmgrd, see [Setting Up a Server-Specific License \(lmgrd only\)](#).

To configure the license server (ladmin):

DesignSync provides a script, `setup_license_server` to perform the configuration necessary to support using ladmin to manage the Flexera licenses.

Note: You may run this script either before or after running the `sync_setup` script to configure the DesignSync server.

The `setup_license_server` script eliminates the need to manually configure, using the web interface, the license server. The script performs the following operations:

- Creates the ladmin data directory within the `$SYNC_SITE_CNFG_DIR` directory structure. For information `$SYNC_SITE_CNFG_DIR`, see [Environment Settings for Users](#).

- Prompts the user for a license file to import. If the user has already configured the DesignSync server and specified a site wide license, the license file (`$SYNC_SITE_CNFG_DIR/syncinc.lic`) is displayed as the default. Otherwise the default is blank.
- Copies or links the license file from the specified location to `$SYNC_SITE_CNFG_DIR/lmadmin/syncinc.lic`.
- Prompts for the port number to use for the lmadmin web server the first time the script is run. This value is saved in the site registry and used for subsequent script runs. By default, the port is 8080.
Important: Running lmadmin as root or on a root port is not a supported configuration. If you choose a different web server port, avoid using a root port which can cause problems for the license server.
- Runs lmadmin with the values set by the script.

Notes:

- Regardless of the directory `setup_license_server` is invoked from, the current working directory is `$SYNC_SITE_CNFG_DIR/lmadmin` when the script completes.
- When you run the `startup_license_server` script, it always outputs the following message, "WARNING: Configuration file does not have a valid signature."

Setting Up a License Server and Default Site License

This configuration sets up a DesignSync installation license server and a default site license.

Note: This configuration assumes that you want to set up a license server using the FlexNet software that DesignSync supplies.

This configuration:

- Installs the FlexNet license manager daemon and the DesignSync vendor daemon (SYNCINC).
- Copies the license file to the directory defined in the `SYNC_SITE_CNFG_DIR` environment variable and renames the license file to `syncinc.lic`.

Note: The default value for `SYNC_SITE_CNFG_DIR` is `<SYNC_SITE_CUSTOM>/config`. The default value for `<SYNC_SITE_CUSTOM>` is `<SYNC_CUSTOM_DIR>/site`. The default value for `<SYNC_CUSTOM_DIR>` is `<SYNC_DIR>/custom`.

Note: If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file. Old information cached in this file can be a source of licensing problems. See License Management Problems and Solutions for information.

To install the license server and set up a site license:

Note: You should perform this installation from the `syncmgr` account.

1. Obtain the product licenses and product distribution.
2. Unpack the product distribution.
3. Install the product.
4. Run the **sync_setup** script to configure the server.
5. Choose **Set up default licensing**.
6. When the script asks: "Would you like licensing set up for you? (y/n) [y]" **answer y**.
7. Choose the license setup appropriate for your system from the prompt:

[1] Specify a license file.

[2] Specify a directory containing license files.

[3] Specify the license server(s) host and port.

Note: If you specify a single license file (option 1), the licensing setup script copies the license file to the directory defined in the `SYNC_SITE_CNFG_DIR` environment variable (the default definition is `<SYNC_CUSTOM_DIR>/site/config`).

These options are described in detail in Understanding License Selection and Preparing the License File.

Notes:

- The license selected here is the default license setup for the site. If you install SyncServers, you can specify that they use this **default site license**.
- You also can specify that the server use a license file other than the default site license. See Setting Up a Server-Specific License.

The `sync_setup` script returns you to the prompt: "What would you like to set up or configure?"

8. Complete the installation or upgrade by configuring client settings and SyncServers.

Note: During the configuration of each server, when the script asks: "How will the server find its license file?" choose **Use the default site license file**.

Starting the License Server

Important: Before you start the license server, you must configure lmadmin. To configure lmadmin, see Configuring the License Server. If you have already configured lmadmin, you do not need to run the procedure again.

After completing the `sync_setup` installation or upgrade steps, start the license server. At the UNIX prompt on the computer that will be the license server, type

```
start_license_server to start the license server for lmadmin.
start_license_server.lmgrd to start the license server for lmgrd
```

The script starts the FlexNet license server and the DesignSync license manager daemon (SYNCINC), directing them to use the license file you specified during licensing setup.

Note for lmgrd users: To start a license server other than the default, use the `start_license_server.lmgrd` command with the `-syncserver` argument and specify the server's host name and port. For example:

```
% start_license_server.lmgrd -syncserver sitelicenseserver:2647
```

Integrating a License into an Existing FlexNet License Management Configuration

If the FlexNet license management software is already installed on your computer, you already have a license server and license file with other FEATURE lines. If you want to use the existing license server, you need to:

1. Copy/move the DesignSync vendor daemon (SYNCINC) to a location where your existing license server can access it.
2. Integrate the DesignSync product license with the existing license(s).

To integrate a new DesignSync license with an existing license for another product, use either of the following methods:

- Add the path for the DesignSync license file (`syncinc.lic`) to the definition for the `LM_LICENSE_FILE` environment variable.

FlexNet v6+ allows the `LM_LICENSE_FILE` variable to contain a colon-separated list of pathnames to license files (UNIX).

When you configure the SyncServer, you can specify that it use the environment variable `LM_LICENSE_FILE` to locate the license file.

- Modify the existing license file to add the `VENDOR` and `FEATURE` lines from the DesignSync license.

For information on either of these methods, see the FlexNet End User Manual on the Flexera web site at <http://www.flexerasoftware.com>.

3. Run the License Manager Reread utility to have the license manager recognize the new license information and to start the vendor daemon (`SYNCINC`).

Guidelines:

- The license manager daemon must be FlexNet v6.0 or later in order to recognize the `VENDOR` line of the DesignSync product license. If your license server is running a FlexNet version earlier than v6.0, you should upgrade to the latest version. Alternately, you can edit the license file to change the keyword `VENDOR` to `DAEMON` and specify the full path to the `SYNCINC` daemon.
- The FlexNet license management daemon looks for the `SYNCINC` daemon in the same directory in which it resides. If you start the license manager from another directory, you must specify the `FULL` path to the license server. If you give a relative path, the `SYNCINC` daemon cannot communicate with the `lmgrd` daemon and license requests fail. To avoid this problem, add the path for the `SYNCINC` daemon to `VENDOR` line in the license file. (**Note:** If you use the `lmgrd` and the `start_license_server` script, it starts the `lmgrd` daemon for you.)
- The `start_sync_server` script creates the environment variable `LM_LICENSE_FILE` and defines its path to the location of the `syncinc.lic`. If the `LM_LICENSE_FILE` variable is already set, the `syncserver` startup script adds the `syncinc.lic` path to the end of the definition.
- If you are using `lmgrd` and you set `LM_LICENSE_FILE` to a directory, then you cannot use `start_sync_server` to start the license manager daemon.
- If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file. Old information cached in this file can be a source of licensing problems. See License Management Problems and Solutions for information.

Using a `<VendorDaemonName>_LICENSE_FILE` Variable

If you have a site that uses a corporate Flex license server, you can use a `<VendorDaemonName>_LICENSE_FILE` variable where you specify the name of the

license file you want to use. This capability is provided by the FlexNet License Management Software.

Using this method of managing the license files lets you specify a separate environment variable for any tools you use, and ensures your licensing is manageable.

The `<VendorDaemonName>_LICENSE_FILE` variable method of setting the license is similar to using `LM_LICENSE_FILE`, but is daemon specific.

To use this method with DesignSync, define a `SYNCINC_LICENSE_FILE` variable in your `$SYNC_DIR/.syncinc.custom` file. When you start any DesignSync tools, the `SYNC_DIR/.syncinc.custom` file is sourced, and the `SYNCINC_LICENSE_FILE` variable is prepended to the license path. `SYNCINC_LICENSE_FILE` can be set to a file or a directory.

It is important to note that setting this variable takes precedence over any `LM_LICENSE_FILE` variable that is set. The `SYNCINC_LICENSE_FILE` variable also takes precedence over any license file, default or server specific, that you specified during the DesignSync installation.

Setting Up Licensing for a Client-Only Client-Side Vault

In the case of DesignSync or DesignSync DFII client-side vaults, there is no server involved. The client application (DesignSync or DesignSync DFII) requests the license directly from the FlexNet license manager.

You need a license only if you are not the owner of a client-side vault. If you browse a SyncServer or a local workspace or vault, the client does not need a license.

To set up licensing for access to a client-side vault:

1. Obtain the product licenses and product distribution.
2. Unpack and install the product according to the instructions in the ENOVIA Synchroncity DesignSync Data Manager Installation.
3. Run the **sync_setup** script to configure the licensing.
4. Choose **Set up default licensing**.
5. When the script asks: "Would you like licensing set up for you? (y/n) [y]" answer **yes**.
6. When the script prompts you, enter the location of your product license file.

The licensing setup script copies the license file to the directory defined in the `SYNC_SITE_CNFG_DIR` environment variable.

Note: The default value for `SYNC_SITE_CNFG_DIR` is `<SYNC_SITE_CUSTOM>/config`. The default value for

`<SYNC_SITE_CUSTOM>` is `<SYNC_CUSTOM_DIR>/site`. The default value for `<SYNC_CUSTOM_DIR>` is `<SYNC_DIR>/custom`.

The `sync_setup` script returns you to the prompt, "What would you like to set up or configure?"

7. After completing the license configuration with `sync_setup`, start the license server. If you are using `ladmin`, you must perform the Configuring the License Server procedure first. At the Unix prompt on the computer that will be the license server, enter
start_license_server for `ladmin`
start_license_server.lmgrd for `lmgrd`

The script starts the FlexNet license server daemon and the DesignSync license manager daemon (SYNCINC), directing them to use the license file you specified during licensing setup.

Notes:

- If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file. Old information cached in this file can be a source of licensing problems. See License Management Problems and Solutions for information.
- You can automatically start a license server at system boot time by using the `S90syncinc` script. In addition to automatically starting a SyncServer, the `S90syncinc` script starts the license server. For information on using this script, see Stopping or Restarting a SyncServer at System Shutdown or Boot Time.

The License File

DesignSync responds to your request for a product license by sending the license information to you via email. You copy the information to an ASCII file and save the file.

Here is an example of a purchased license for DesignSync:

```
*****  
  
SERVER this_host 00093d10c65c  
  
USE_SERVER  
  
VENDOR SYNCINC  
  
INCREMENT DesignSync SYNCINC 6.0 18-oct-2010 5 53EA9647EB90 \
```

```
HOSTID=ANY OVERDRAFT=20 BORROW=720 vendor_info="License \
Delivered to : Velizy_DS" ISSUED=19-Oct-2010 \
NOTICE="Copyright Dassault Systemes 2010" SN=EM100063 \
SIGN=B6426BF89604
```

```
*****
*****
```

The line that begins with...	Identifies...
SERVER	<p>The license server for the license. This line specifies:</p> <ul style="list-style-type: none"> • Server hostname • Server hostid <p>Notes:</p> <ul style="list-style-type: none"> • You may edit only the hostname. • The SERVER line of license files do not contain a default port for the license manager daemon.
VENDOR	<p>The name and path of the DesignSync vendor daemon (SYNCINC), which keeps track of the DesignSync licenses.</p> <p>Note: The path name is optional and you may edit it.</p>
INCREMENT	<p>Each licensed product. This information includes:</p> <ul style="list-style-type: none"> • Product name • Name of the vendor daemon that manages the license • Product version • License expiration date • Number of licenses • License key (identifier) • Type of license <p>Notes:</p> <ul style="list-style-type: none"> • Do not edit any of this information. • The COP feature line shows that a Cadence Objects Processing license is present. This is a requirement for running DSDFII.

UPGRADE	Licenses converted from one product version (identified by a FEATURE line) to another version of that product. Note: Used only for licenses under a partial maintenance agreement.
---------	---

Types of Licenses

DesignSync provides these licenses:

A Purchase license...	Demonstration and Evaluation licenses...
Is granted for a specific computer; the SERVER line of the license specifies the computer's hostname and hostid.	Are open licenses; not granted for a specific computer.
Expires after a year, except in the case of multi-year subscriptions. (These licenses can be renewed.)	Expire after a short period of time (15 - 60 days).

Understanding DesignSync Product License Management

Licensing for the DesignSync products is per user, within the domain of a license server. Multiple SyncServers can reference a single license server (or a license triad). Throughout all such servers, a user will consume a single license, no matter how many of those servers the user is accessing, and from any client. Users do not require separate licenses for different clients, nor are there any special requirements for WAN-ing.

There are times when a client may wish to access servers that use different license servers. For example, a contract designer may wish to work with two different companies. Each of those companies will have its own set of SyncServers and license servers. In these situations, the user will need separate licenses for each license server.

License checks are performed whenever a user manipulates a vault (for example, checkin, checkout, populate, or tag operations). When a user contacts the SyncServer to perform an operation, the SyncServer contacts the FlexNet license server/manager for a license. In the case of a client-side vault (where there is no SyncServer), the client contacts the license server for a license when a user who is not the vault owner manipulates the vault.

Note: The FlexNet license server/manager and its license enforcement mechanism are separate from the SyncServer or other DesignSync server-based applications. For information on FlexNet functionality, see the Flexera web site at <http://www.flexerasoftware.com>.

The FlexNet license manager daemon accepts the request for a license from the SyncServer and hands it off to the appropriate vendor daemon -- SYNCINC. The SYNCINC daemon tracks how many DesignSync licenses are being used and which users are using them.

The SyncServer establishes a connection to the SYNCINC daemon and requests a license. The daemon grants or denies the license, based on the number of licenses available.

Viewing the Status of License Activity

To find out who has licenses checked out and view other status information for licenses, you may do one of the following:

1. Check the `$SYNC_CUSTOM_DIR/site/config/lmadmin/logs/SYNCINC.log` file
2. Use the lmadmin web interface, `http://<host>:<port>`, where `<host>` and `<port>` are that of the license server.

For more information about using the FlexNet web interface, see the FlexNet user documentation at <http://www.flexerasoftware.com>.

Starting Up and Shutting Down the License Server (Imgrd only)

This topic discusses starting the License Server using the DesignSync license server script and starting and stopping the License Server using the Flexera-provided license server management tools.

- Starting the License Server Using the DesignSync Script
 - Starting the default license server (the site license server)
 - Starting a license server other than the default
 - Calling the `start_license_server` script from another script
- Starting a license server manually
- Shutting Down the License Server

Starting the License Server Using the DesignSync Script

To start the license server, run the `start_license_server.Imgrd` script. This script invokes the FlexNet license server with some DesignSync-specific arguments.

Note: The `start_license_server` script assumes you are using the default site license setup and that your license file is resident at `$SYNC_CUSTOM_DIR/site/config/syncinc.lic`. If you set `LM_LICENSE_FILE` to a directory, then you must start the license server directly

Using the `start_license_server.Imgrd` Script

You can use the `start_license_server.lmgrd` script with the following arguments:

If you specify...	The script...
No arguments	Invokes the <code>lmgrd</code> daemon and directs it to use the license file you specified in the license server setup section of the <code>sync_setup</code> script.
-help	Displays the help information for the <code>start_license_server</code> script and its arguments
-c <license_file>	Invokes the <code>lmgrd</code> daemon and directs it to use the license file for the server specified by the -c argument.
-syncserver <hostname:port> Where <hostname:port> is the hostname and port number of the DesignSync server you set up with <code>sync_setup</code>	Invokes the <code>lmgrd</code> daemon and directs it to use the license file for the server specified in the -syncserver argument. The script uses the hostname and port to locate the directory in the <code><SYNC_CUSTOM_DIR>/servers</code> hierarchy where that server's <code>syncinc.lic</code> resides.

Starting the default license server (the site license server)

To start the default license server, enter the following command at the UNIX prompt:

```
start_license_server.lmgrd
```

Note: You can automatically start a license server at system boot time by using the `S90syncinc` script. In addition to automatically starting a SyncServer, the `S90syncinc` script starts the license server. For information on using this script, see [Stopping or Restarting a SyncServer at System Shutdown or Boot Time](#).

Starting a license server other than the default

Use the `start_license_server.lmgrd` command with the `-syncserver` argument specifying the server's host name and host ID. For example:

```
% start_license_server.lmgrd -syncserver projlicenseserver:2647
```

Calling the start_license_server script from another script

When you need the license server script to return to the prompt so that the script that calls it can continue processing, use this procedure.

Use the `start_license_server.lmgrd` command with the `-l` `<debug_log_path>` argument, specifying the name of the log file. For example:

```
start_license_server.lmgrd -l licensestart.log
```

The `start_license_server.lmgrd` script passes the `-l` argument to the `lmgrd` daemon, which redirects the screen output to the file you specified.

Note: The `-l <debug_log_path>` argument is an argument to the FlexNet `lmgrd` daemon. With the `start_license_server` script, you can use any argument you can pass to the FlexNet `lmgrd` daemon program. For information on these arguments, see the FlexNet End User Manual on the Flexera web site at <http://www.flexerasoftware.com>.

Starting a license server manually

When you use the `LM_LICENSE_FILE`, a directory containing license files, or specify the host/post pair to start a license server, you cannot use the DesignSync provided script. You must manually start the license server.

To manually start the license server, type:

```
lmgrd -c <license_file_list>
```

Where the license file list is:

- the `LM_LICENSE_FILE` variable
- the full path to a single license file
- the full path to the directory containing the license files

Shutting Down the License Server

To perform an orderly shutdown of the license server, use the `lmdown` utility. Enter:

```
lmutil lmdown
```

Where:

`lmutil` is a script, included with DesignSync, that calls FlexNet utilities.

`lmdown` invokes the FlexNet license shutdown utility

The utility shuts down all license management (`lmgrd`) and vendor daemons.

Note: The `lmdown` utility also provides arguments that let you shut down a particular vendor daemon or all daemons specified in a license file. For more information about

the `lmdown` utility, see the FlexNet End User Manual on the Flexera web site at <http://www.flexerasoftware.com>.

Setting Up a Server-Specific License (Imgrd only)

This configuration sets up a separate license for a SyncServer, a configuration useful if you want to set up licenses specifically for use by project team members.

Note: These steps assume that a license server exists on the same Local Area Network (LAN) as the SyncServers you plan to configure.

This configuration:

- Installs the FlexNet license manager daemon (Imgrd) and the DesignSync vendor daemon (SYNCINC).

Copies the license file to the appropriate directory

```
$SYNC_CUSTOM_DIR/servers/<hostname>/<port>/syncinc.lic.
```

Note: If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file. Old information cached in this file can be a source of licensing problems. See License Management Problems and Solutions for information.

To set up a server-specific license:

Note: You should perform this setup from the `syncmgr` account unless you are allowing project leaders to set up servers by using UNIX group permissions. See the questions in the `sync_setup` script for more information.

1. Obtain the product licenses and product distribution.
2. Unpack and install the product according to the instructions in the ENOVIA Synchronicity DesignSync Data Manager Installation.
3. Run the **sync_setup** script to configure the server.
4. Choose **Configure/Upgrade a server** and answer script prompts for port numbers, vault and metadata paths.
5. In response to the prompt, "How will the server find its licenses?" choose the appropriate method. For information about the different licensing options, see Understanding License Selection and Preparing the License File.
6. Answer the prompts for email notification addresses and DNS name for the host.

When server configuration is complete, the script displays a success message and reminds you to start the SyncServer.

7. At the prompt, "What would you like to set up or configure?" choose **Exit**.
8. Start the license server. On the license server machine, enter:

```
% start_license_server.lmgrd -syncserver  
<hostname>:<port>
```

Where `-syncserver` specifies information that the script uses to locate the license file.

For example:

```
% start_license_server.lmgrd -syncserver  
ASICprojectserver:30038
```

Note: The license server must reside on the machine specified in the `SERVER` line of the license file.

The script starts the FlexNet license manager daemon (`lmgrd`) which reads the license.

9. Start the SyncServer by logging into the sync server host, making sure the `<SYNC_DIR>/bin` directory is in your directory path, and typing `start_sync_server.lmgrd`

The `start_sync_server` script:

- Defines the environment variable `LM_LICENSE_FILE` as the path for `syncinc.lic`. If the `LM_LICENSE_FILE` variable is already defined, the script adds the `syncinc.lic` path to the end of that definition.
- Starts the specified SyncServer.

License Management Problems and Solutions

To help you manage DesignSync product licenses, here are some frequently-asked questions (and answers) and some problem descriptions (and solutions).

How do I know when DesignSync licenses are expiring and what do I do?

The DesignSync server notifies you by email when your DesignSync product license is about to expire. The address for the email and the number of days' notification time are the address and notification time you specified during SyncServer configuration.

The email contains instructions on how to obtain a new license.

Can I Transfer a License from One Computer to Another?

DesignSync allows transfer of a license file to another computer (different hostname and `hostid`) for these reasons:

- Hardware failure
- Hardware performance upgrade
- Movement of facility or users results in a new server.

When I specify a port other than the default, how do I set the VENDOR path?

If you specify a port other than the default for the SYNCINC vendor daemon, you should specify a full path to the SYNCINC daemon on the VENDOR line.

For example, suppose you edit the VENDOR line of the license and specify a port for the SYNCINC daemon by adding the `port=portnum` argument. In addition to the `port=portnum` argument, you should also specify a full path to the SYNCINC daemon. The VENDOR line then might look like the following example:

```
SERVER server1 0000861FB938
```

```
VENDOR SYNCINC /usr/local/vendor_daemons/SYNCINC port=30078
```

Note: The port number on the VENDOR line of the license cannot be the same as the port number on the SERVER line.

For more information on the `port=portnum` argument, see the FlexNet End User Manual on the Flexera web site at <http://www.flexerasoftware.com>.

Important: If you put the SYNCINC vendor daemon in a different directory from the license manager daemon, you must edit the license and specify the full path to the SYNCINC daemon on the VENDOR line. Specifying the full path to the SYNCINC daemon ensures that the license manager daemon can communicate with the SYNCINC daemon. For example:

```
SERVER server1 0000861FB938
```

```
VENDOR SYNCINC /usr/local/vendor_daemons/SYNCINC
```

start_license_server - c starts lmgrd but not SYNCINC

Problem: You use `start_license_server.lmgrd - c` and it starts the lmgrd license manager daemon but not the SYNCINC vendor daemon.

Possible Cause: If a port is specified on the SERVER line of the license, `start_license_server.lmgrd - c` starts only start the lmgrd daemon, not the SYNCINC vendor daemon.

Solution: Specify the full path to the SYNCINC daemon on the VENDOR line of the license file. Specify the VENDOR line as:

```
VENDOR SYNCINC <path_to_SYNCINC>
```

Where `< path_to_SYNCINC >` is only the path and does not include SYNCINC. For example:

```
VENDOR SYNCINC /usr/local/vendor_daemons
```

UNSUPPORTED warning appears in FlexNet log file

Problem: The `/home/syncmgr/lm.log` file contains UNSUPPORTED warnings for DesignSync products.

Possible Cause: License information in the `.flexlmrc` file (created by FlexNet license management software) overrides information in the license file specified by the `LM_LICENSE_FILE`. Old information cached in this file can be a source of licensing problems.

Solution: If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file.

Note: The DesignSync `SYNC_FLEXLMRC` environment variable controls the creation of the `$HOME/.flexlmrc` file. By default, this variable is set to 0, disabling the file's creation. However, if the `.flexlmrc` file already exists, you must delete it manually. (For information about the `SYNC_FLEXLMRC` environment variable, see Using Environment Variables.)

Licenses being used from a different license file than expected

Problem: Licenses are being used from a license file other than the SyncServer specific license file that was specified when `sync_setup` was run to configure the SyncServer. `start_license_server.lmgrd -syncserver <host:port>` was used to start the license server.

Possible Cause: License information in the `.flexlmrc` file (created by FlexNet license management software) overrides information in the license file specified by the `LM_LICENSE_FILE`. Old information cached in this file can be a source of licensing problems.

Solution: If you make changes to your licensing installation, you should delete your `$HOME/.flexlmrc` file.

Note: The DesignSync `SYNC_FLEXLMRC` environment variable controls the creation of the `$HOME/.flexlmrc` file. By default, this variable is set to 0, disabling the file's creation. However, if the `.flexlmrc` file already exists, you must delete it manually.

(For information about the SYNC_FLEXLMRC environment variable, see Using Environment Variables.)

SyncServer Administration

Overview of SyncServer Administration

As a system administrator, you need to understand how SyncSync is architected so that you can set up and administrate with confidence. The following topics in this section explain how DesignSync is organized and installed:

- Installing DesignSync Tools - Gives pointers to information on installing and licensing for DesignSync.
- SyncServer Architecture - Describes the SyncServer architecture and the database.

The topics in the section "Administering the SyncServer" describe the Administer Server panels and other routine server administration tasks:

- Using the Administer Server Panel - Provides an overview of the Administer Server panel, which gives you access to common SyncServer administration tasks.
- Changing Vault Types - Describes how to change vault type settings and add new vault types including copy, zipped copy, and RCE to the SyncServer.
- Setting RevisionControl Note Generation - Describes how to enable RevisionControl note generation in response to DesignSync revision-control operations such as check in, check out, and populate.
- Upgrading RevisionControl Notes - Explains how to upgrade your RevisionControl note type if you have just upgraded to the product version.
- Backing Up Your Server - Describes how to back up and restore your SyncServer.
- Accessing Apache Document Root - Describes how to open access to the Apache document root (`htdocs`) directory.
- Controlling HTTP Headers - Describes how to suspend DesignSync's default HTTP headers and output your own headers.

The topics in the section "Administering Users" section explain different methods for importing users into DesignSync from other user databases, such as LDAP:

- Enabling LDAP - Describes how to enable a server's DesignSync LDAP client so that users in an LDAP database are imported into user database.
- Making Advanced LDAP Settings - Describes how to configure the DesignSync LDAP client in registry files.
- Creating User Authentication Scripts - describes how to write a script to import users when you do not use an LDAP database.

Finally, the topics in the "Troubleshooting" section will help you resolve technical issues.

- Troubleshooting - Gives pointers to the DesignSync Known Problems and Solutions documentation and to information on troubleshooting your customizations.
- SyncServer Error Log File Messages - Gives information on how to use SyncServer log files.

Related Topics

Using the Administer Server Panel

Setting Up a SyncServer

Run `sync_setup` to set up a new server, choosing the menu option to configure a new server. You will be prompted to specify communication port numbers, the location of the server data, and other administrative information. For more information, see [Configuring the SyncServer](#).

Resetting the SyncServer

The **Reset Server** and **Access Reset** hyperlinks in the ProjectSync menu let you reset your SyncServer or your access control settings. These hyperlinks appear only when you have `AdministrateServer` access permissions. (See the [ENOVIA Synchronicity Access Control Guide](#) for details on access rules.)

Reset Server

A reset is similar to stopping and restarting the server. You should reset the server whenever you make changes to any of your configuration files. When you reset the server, ProjectSync reloads all the files that are read at server startup - for example, your `*.conf` configuration files, Tcl configuration files, access control settings, and so on.

To reset the SyncServer:

1. Click **Reset Server** in the **Server** section of the **Admin Menu** of the DesignSync Web UI.
2. Click **OK** on the confirmation pop-up window.

When you initiate a server reset, the SyncServer will complete any operations that are in progress. New requests may fail during the brief period while the server reset is proceeding.

When the server has been reset, you see an Operation Successful panel that confirms the operation.

Access Reset

If you change your access control settings, you can reset them without stopping and restarting the SyncServer. Clicking the **Access Reset** button causes the server to reread the access control files.

To reset your access control settings:

1. Click **Access Reset** in the **Server** section of the **Admin Menu** of the DesignSync WebUI..
2. Click **OK** on the confirmation pop-up window.

Note: When you reset your access control settings, DesignSync checks the AccessControl file for syntactic errors before changing the server state. If the file contains such errors, the command aborts leaving the server state (and access controls) unchanged. Errors in the AccessControl file can cause your server to become unresponsive if the errors are not corrected quickly. To avoid this problem, correct access control errors immediately and reset the server.

Manual Reset for Registry, Environment Variable, and Revision Control Changes

When you make changes to your registry settings, environment variables, or RevisionControl note generation settings, you need to manually shut down and restart your server using the `stop_sync_server` and `start_sync_server` commands. Resetting the server does not reread these settings.

Important: To shut down the server always use `stop_sync_server`, which stops both the SyncServer and the database server.

If you cannot stop PostgreSQL with `stop_sync_server`, use `pgcontrol stop fast` or, as a last resort, `pgcontrol stop immediate`. **Never use `kill -9` on the postgres or postmaster process.**

Using the Administer Server Panel

The Administer Server panel lets you access common SyncServer administration tasks. These tasks include backing up and restoring your server, changing vault types, enabling the generation of RevisionControl notes, and enabling alternate user authentication such as 3DPassport, or LDAP.

You cannot access the Administer Server panel unless you have permission to perform administrative functions. See the ENOVIA Synchronicity Access Control Guide for details.

To access the Administer Server panel:

1. On the DesignSync Web UI menu, open the **Admin Menu** to view the administration options.
2. From the **Server** section of the menu, select **Administer Server**. This menu option appears only when you have permission to perform administrative functions.
3. On the Administer Server: Choose Operation panel, click the hyperlink for the type of operation you want to perform:
 - **Server Settings** - Takes you to the Administer Server Settings panel. This panel contains a tabbed dialog box where you can enable generation of RevisionControl notes (see Setting RevisionControl Note Generation), enable 3DPassport (Central Authentication Server) login, (see Enabling 3DPassport), enable LDAP login (see Enabling LDAP).or add or edit vault types (see Adding or Editing a Vault Association and Changing Vault Types).

You must enter your user name and password before you can access the dialog box on the Administer Server Settings panel.

- **Backup** - Takes you to the Backing Up Your Server panel (see Backing Up Your Server), where you can back up your entire server vault, metadata, and notes.
- **Restore** - Takes you to the Restore From Backup panel (see Restoring Your Server Vault Data), where you can restore individual vault items and small sections of the vault hierarchy from a backup. (See Restoring All Server Backup Data for information on how to restore an entire server.)

Related Topics

[Administering ProjectSync](#)

[Backing Up Your Server](#)

[Adding or Editing a Vault Association](#)

[Changing Vault Types](#)

[Enabling 3DPassport](#)

[Enabling LDAP](#)

[Restoring All Server Backup Data](#)

[Restoring Your Server Vault Data](#)

Setting RevisionControl Note Generation

Enterprise Design Synchronization Queue

DesignSync maintains synchronization between DesignSync modules and related Enterprise Design Objects. The data being synchronized can be automatically pushed to the Enterprise server or manually synchronized.

Before synchronization occurs, the system maintains a list of transactions queued for processing on the Enterprise Server. This list is displayed on the Enterprise Design Synchronization Queue in the DesignSync WebUI. Transactions in the queue are listed in the order in which they were created, which is also the order in which they will be processed.

If you are performing manual synchronization, or require certain transactions to be processed sooner than the automatic synchronization, or want to synchronize in a particular order, you can select transactions and launch synchronization of those transactions directly from this page.

When synchronization is automatically pushed to the Enterprise server, DesignSync uses the Alarm Trigger Maintenance Timeout setting to determine how often the data is pushed to the Enterprise server.

Viewing the Enterprise Design Synchronization Queue

From the DesignSync Web UI **Admin Menu**, select **Server | Enterprise Queue**. This displays a table containing the list of all pending transactions. If there are no pending transactions, the list is empty. The table is not sortable. It will always display the list of transaction in order of processing.

Beneath the table is a list of operations.

Note: To select all transactions, click the CheckBox in the header row.

Enable

Places the selected transactions in an active state which allows them to be synchronized when the next synchronization action occurs.

Disable

Places the selected transactions in an inactive state so they will not be synchronized when the next push occurs. Transaction are automatically disabled, by default, after 10 unsuccessful attempts to process the transaction.

Delete

Removes the selected transaction from the queue so it will never be synchronized. This action cannot be undone.

Synchronize

Synchronizes all selected, enabled transactions, clearing them from the queue.

Related Topics

Enterprise Servers

Site Options

Alarm Trigger Frequency (MaintenanceTimeout)

Enterprise DesignSync Administration User's Guide: Enterprise Design Synchronization

Setting up Secure Communications

Overview of Secure Communications

DesignSync servers (SyncServers) are capable of establishing a secure communications channel with DesignSync clients (DesSync graphical interface, dss, dssc, stcl, stcl, or the DesignSync web UI). This secure communications channel is built on the Secure Socket Layer (**SSL**) protocol and the RSA Public Key encryption algorithm.

DesignSync products use the HTTP protocol for both secure and non-secure transactions. The secure version includes an SSL encryption of the HTTP protocol.

Enabling the secure communications channel involves obtaining an X.509 Certificate from a **Certificate Authority** (CA) vendor and installing it into the configuration directory of the SyncServer machine. (X.509 Certificates contain encryption keys and information describing an individual or an organization -- for example, a SyncServer.) Then during server configuration, you specify a secure (SSL) port in addition to the standard non-secure (cleartext) port.

Secure communications between the DesignSync client and SyncServer are made possible through a redirect mechanism working with access controls that you, as DesignSync administrator, set up on the server.

Once secure communications are set up, users do not need to specify a secure protocol (https or syncs) or a certain port in order to use a secure communications channel. They specify the usual cleartext port and protocol (sync or http) in DesignSync. The SyncServer detects that a user has accessed the server via a cleartext port but needs a

secure port and redirects the DesignSync client software to a secure communication channel, as indicated in the access control file.

The only time users might explicitly specify a secure protocol is when they want more security than the minimum required by the server.

Related Topics

Enabling Secure Communications

Using Secure Communications

Setting Access Controls for Secure Communications

Setting Access Controls for Secure Communications

By default, DesignSync allows cleartext (non SSL) communications for every user, from any IP address. Enabling secure communications involves requiring that communications from outside networks be SSL (secure) communications. To impose this requirement, you create a site-wide or server-specific access control file that contains access control commands to deny non SSL communications.

To require secure communications:

1. In a text editor, open the `AccessControl` file in your custom site or servers share directory. (If this file does not already exist, the DesignSync installation copies a sample `AccessControl` file to both the `<SYNC_DIR>/custom/site/share` directory and the `<SYNC_DIR>/custom/servers /<host>/<port>/share` directory.)

DesignSync includes this file so you can use it as a template for creating a site-wide or server-specific access control file. The example file includes access control commands for requiring SSL communications.

See *Access Control Guide: Setting up Access Controls* for more information on using this example file.

2. In the custom access control file, make the following changes:
 - Add or modify the access control command to deny cleartext (NonSSL) communications for all users:
 - `access deny NonSSL everyone`
 - Enter the command to allow cleartext communications for certain local network addresses only. For example:

```
access allow NonSSL everyone when client_ip
10.1.1.*
```

3. To have your SyncServer read the new access control setting, use the `access reset` command.

Related Topics

Overview of Secure Communications

Using Secure Communications

Access Control Overview

Enabling Secure Communications (UNIX only)

To enable secure communications, obtain and install an X.509 Certificate and then modify the server's access controls.

Obtaining and Installing an X.509 Certificate

X.509 certificates are provided by Certificate Authority (CA) vendors. To obtain a certificate:

1. Find a CA vendor.

There are many CA vendors; you can locate one by performing a search with your web browser.

You can also use the list of Certificate Authority vendors your browser provides. To view the list, display the browser's security information.

2. Follow directions on the CA's web site and obtain a PEM-encoded **X.509 Server Certificate** for an Apache server.

Note: In the Distinguished Name field of the certificate, you must enter the host name of the SyncServer (for example: `myserver.mycompany.com`), not an individual or organization.

3. Save the certificate (or key) files with the names `httpsd.key` and `httpsd.pem` and specify the SyncServer's configuration directory as their location:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/conf/httpsd.key
```

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/conf/httpsd.pem
```

4. Configure the SyncServer and specify an SSL port in addition to the cleartext port. (To configure the SyncServer, run the

`$SYNC_DIR/sync_setup` script and choose **Configure/Upgrade a server.**)

The script modifies the SyncServer's configuration file (`httpd.conf`) to recognize the certificates and to open a secure (SSL) port.

5. Start the SyncServer. (Enter `start_sync_server <port>`.)

Disabling Secure Communications

If your users use only the cleartext port and do not need secure communications, you may want to disable the Secure Socket Layer (SSL) port assigned during server configuration.

To disable the SSL port, edit the `httpd.conf` file to remove references to SSL:

1. Stop the SyncServer. (Enter `stop_sync_server <port>`).
2. In a text editor, open the `httpd.conf` file. This file is located in the directory `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/conf`.
3. Comment out the definitions for the following nine directives (which you can find by searching on SSL):

```
SSLPassPhraseDialog
SSLSessionCache
SSLSessionCacheTimeout
SSLMutex sysvsem
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
```

4. Comment out the line that contains "Listen" and the SSL port you specified during server configuration.

For example, if you used the default port of 2679, you would comment out this line: Listen 2679

5. Comment out the entire "Virtual Host Configuration" section, which contains these lines:

```
<VirtualHost host.domain.com>

SSLEnable

</VirtualHost>
```

- Restart the SyncServer. (Enter **start_sync_server**<port>.)

Related Topics

Overview of Secure Communications

Enabling Secure Communications

Using Secure Communications

For users to use the secure communications channel, the DesignSync administrator must first obtain a certificate and enable these communications. See Enabling Secure Communications for information.

There are two ways to access a server that provides secure communications:

- Use the standard cleartext http or sync protocol. For example:
http://host:port/path
- Use the SSL (encrypted) https or syncs protocol. For example:
https://host:sslport/path

Once secure communications have been enabled, specify the cleartext protocol (http or sync) and port as you usually do. If the SyncServer detects that your communications need a secure port, it redirects the DesignSync client software to a secure communications port, as indicated in the access control file.

If you want more security than the minimum required by the server, specify a secure protocol (syncs or https).

Notes:

- **Avoid mixing cleartext protocol types with SSL port numbers and vice versa.**

If you specify the http or sync protocol with an SSL port number, or https or syncs with a cleartext port number, access to the server fails.

For example, suppose a cleartext port was configured as port number 1084 and an SSL port as port 1085. Then in DesignSync you specify https:// at port 1084.

DesignSync returns the message "Communication Read Failure" or "Communication Write Failure." (The DesignSync Web UI and ProjectSync return the message "Document Contains No Data.")

- **Sometimes you see evidence that the SyncServer has redirected your communications from the cleartext port to the SSL port.**

When you specify the standard http or sync protocol, DesignSync may display the protocol as https or syncs if the SyncServer redirected the communication to an SSL port. Or the DesignSync Web UI and ProjectSync may display https URLs or the SSL port number if the communication was redirected from the cleartext port.

This situation may cause some confusion but should not present a problem for server communications.

Related Topics

Overview of Secure Communications

Enabling Secure Communications

About HTTP Proxy

HTTP Proxy is a program used to enhance the security of a company LAN by controlling HTTP access from the company to the outside world. This program is typically run on a firewall gateway host. Every request from the web browser to the Internet addresses outside the company LAN gets redirected to the proxy, which in turn examines it and, if satisfied, forwards it to the requested IP address. The response from the HTTP server is passed in a similar manner back to the browser.

Most web browsers permit proxy specification in their setup.

In DesignSync, you can set up an HTTP proxy to access remote SyncServers from within a DesignSync session. You specify the proxy IP address and port in the client registry file by setting a registry value. For more information on specifying the proxy in the registry, see HTTP Proxies (ProxyNamePort). The HTTP proxy server can also be specified as an environment variable. For more information on specifying the proxy in an environment variable, see Using Environment Variables.

DesignSync includes information about the type of request being made in the HTTP header of all server calls being sent to the DesignSync server and the origin of those requests. This provides the ability for the proxy server to appropriately dispatch the requests.

DesignSync identifies four basic operation types:

- **READ_CONTENT** - Read file content and server metadata from the DesignSync server. These operations do not perform any modification on the server and include commands like populate, diff, and showhrefs.

- **WRITE_CONTENT** - Change file content and server metadata on the server by adding, deleting, or modifying the content of a module version, configuration, or file. This includes commands like `ci`, `mkbranch`, `remove`, and `upload`.
- **READ_METADATA** - Read server metadata from the DesignSync server. These operations do not perform any modification on the server and include commands like `ls`, `compare`, `showstatus`, and `showmods`.
- **WRITE_METADATA** - Write server metadata information on the server. This includes commands like `lock`, `tag`, `branch`, and `cancel`.

Note: The above list is a small, representative, subset of commands for each of the operation types. The full list is included in [Working with HTTP Proxy Header Information](#).

DesignSync identifies three basic originators. :

- **-USER**: Originates from a client process.
- **-MIRROR** : command comes from an update to a mirror.
- **-ADMIN** : command comes from a client spawned for server-to-server communication, specifically any processes spawned to communicate with the Enterprise server. The mirror system also uses the ADMIN originator type when the Mirror Admin Daemon (MAD) or the Mirror Push Daemon (MPD) spawns an `stlc` to communicate with the Repository Server (RS) or the Mirror Administration Server (MAS).

For more information on how to use and filter HTTP Proxy Header Information, see [Working with HTTP Proxy Header Information](#).

Related Topics

[Using Automatic Proxy Discovery](#)

[Working with HTTP Proxy Header Information](#)

[Proxy Errors](#)

[HTTP Proxies](#)

Working with HTTP Proxy Header Information

The DesignSync clients include information about the type of request being made in the HTTP header of all server calls being sent to the DesignSync server. This provides the ability for the proxy server to appropriately dispatch the requests.

The header information is sent in *PropertyName:Value[-Source]* pairs. The PropertyName for DesignSync headers is always `SyncHeaderInfoCmdType`. The Value is one of the four basic operation types:

- READ_CONTENT
- WRITE_CONTENT
- READ_METADATA
- WRITE_METADATA

The Source is one of three basic source types.

- -USER: command comes from a user process.
- -MIRROR : command comes from an update to a mirror.
- -ADMIN : command comes from a client spawned for server-to-server communication, specifically any processes spawned to communicate with the Enterprise server. The mirror system also uses the ADMIN originator type when the Mirror Admin Daemon (MAD) or the Mirror Push Daemon (MPD) spawns an stlc to communicate with the Repository Server (RS) or the Mirror Administration Server (MAS).

This allows you to filter mirror write commands separately from user-spawned write commands, for example. For more information on understanding how the data replication system uses the HTTP Proxy Headers, see [Processing HTTP Proxy Header information Used by the Data Replication System](#).

Note: The header does not specify the actual command being passed, only the type of operation. When commands include more than one type of operation, such as a checkin operation, which reads the metadata before initiating changes to the contents on the server, the command is sent with the WRITE_CONTENT operation type, because the primary function of the command is to change information on the server.

Understanding Operation Types

Commands which do not appear on these lists, such as add or datasheet may be client-side only commands and do not interact with the server, or by default use the operation type, READ_METADATA.

READ_CONTENT

Read file content and server metadata from the DesignSync server. These operations do not perform any modification on the server. These DesignSync commands are READ_CONTENT types.

Command	Operation Type	Notes
diff	READ_CONTENT	
exportmod	READ_CONTENT	

import	READ_CONTENT	
populate	READ_CONTENT	
showhrefs	READ_CONTENT	

WRITE_CONTENT

Change file content and server metadata on the server by adding, deleting, or modifying the content of a module version, configuration, or file. The following commands are WRITE_CONTENT type operations.

Command	Operation Type	Notes
addhref	WRITE_CONTENT	
ci	WRITE_CONTENT	
edithrefs	WRITE_CONTENT	
hcm *	WRITE_CONTENT	All legacy module (hcm) commands, use WRITE_CONTENT including commands that show module information, such as showconfs, showhrefs, etc.
importmod	WRITE_CONTENT	
migratetag	WRITE_METADATA	
mkbranch	WRITE_CONTENT	
mkfolder	WRITE_CONTENT	
mkmod	WRITE_CONTENT	
mvfile - allconfigs	WRITE_CONTENT	This command writes to file-based vaults.
mvfolder	WRITE_CONTENT	
mvmember	WRITE_CONTENT	
purge	WRITE_CONTENT	
remove	WRITE_CONTENT	
rmfolder	WRITE_CONTENT	
rmhref	WRITE_CONTENT	
rmmod	WRITE_CONTENT	
rmvault	WRITE_CONTENT	
rstcl	WRITE_CONTENT	
unremove	WRITE_CONTENT	
upgrade	WRITE_CONTENT	
upload	WRITE_CONTENT	

READ_METADATA

Read server metadata from the DesignSync server. These operations do not perform any modification on the server. This is not a comprehensive list of READ_METADATA commands, but it does include the most commonly used commands that perform READ_METADATA operations.

Command	Operation Type	Notes
annotate	READ_METADATA	
compare	READ_METADATA	
contents	READ_METADATA	
enobj	READ_METADATA	
entobj show	READ_METADATA	
entobj synchronize	READ_METADATA	This command uses vhistory to get information and thus the operation is a READ_METADATA.
ls	READ_METADATA	
setvault	READ_METADATA	
showlocks	READ_METADATA	
showmods	READ_METADATA	
showproduct	READ_METADATA	Obsolete command. Use entjob show
showstatus	READ_METADATA	
url commands	READ_METADATA	The only url command that isn't a READ_METADATA or a client-side only operation is url setprop.
vhistory	READ_METADATA	
whereused	READ_METADATA	

WRITE_METADATA

Write server metadata information on the server. The following commands are WRITE_METADATA type operations.

addbackref	WRITE_METADATA	
cancel	WRITE_METADATA	
caching	WRITE_METADATA	
deleteversion	WRITE_METADATA	This command is obsolete. Users should migrate operations to the rmversion command.
freezemode	WRITE_METADATA	
lock	WRITE_METADATA	

mkedg	WRITE_METADATA	
mvmod	WRITE_METADATA	
populate -lock	WRITE_METADATA	
reconnectmod	WRITE_METADATA	
retire	WRITE_METADATA	The retire command both retires and unretires (retire -unretire) objects. Both forms of the command are a metadata write operation.
rmedg	WRITE_METADATA	
rmversion	WRITE_METADATA	
rollback	WRITE_METADATA	
setowner	WRITE_METADATA	
switchlocker	WRITE_METADATA	
tag	WRITE_METADATA	
unfreezmod	WRITE_METADATA	
unlock	WRITE_METADATA	
url setprop	WRITE_METADATA	The url setprop command can set server properties or client-side local properties. When url setprop is used to set a server property, it uses the WRITE_METADATA operation.

Alphabetical List of DesignSync Commands with their HTTP Proxy Operation Type value.

Note: Any commands not listed in this table use the operation type READ_METADATA.

Command	Operation Type	Notes
addbackref	WRITE_METADATA	
addhref	WRITE_CONTENT	
annotate	READ_METADATA	
caching	WRITE_METADATA	
cancel	WRITE_METADATA	
ci	WRITE_CONTENT	
compare	READ_METADATA	
contents	READ_METADATA	
deleteversion	WRITE_METADATA	This command is obsolete. Users should migrate operations to the rmversion command.
diff	READ_CONTENT	
edithrefs	WRITE_CONTENT	

enobj	READ_METADATA	
entobj show	READ_METADATA	
entobj synchronize	READ_METADATA	This command uses vhistory to get information and thus the operation is a READ_METADATA.
exportmod	READ_CONTENT	
freezemode	WRITE_METADATA	
hcm *	WRITE_CONTENT	All legacy module (hcm) commands, use WRITE_CONTENT including commands that show module information, such as showconfs, showhrefs, etc.
import	READ_CONTENT	
importmod	WRITE_CONTENT	
lock	WRITE_METADATA	
ls	READ_METADATA	
migratetag	WRITE_CONTENT	
mkbranch	WRITE_CONTENT	
mkedge	WRITE_METADATA	
mkfolder	WRITE_CONTENT	
mkmod	WRITE_CONTENT	
mvfile -allconfigs	WRITE_CONTENT	This command writes to file-based vaults.
mvfolder	WRITE_CONTENT	
mvmember	WRITE_CONTENT	
mvmod	WRITE_METADATA	
populate	READ_CONTENT	
populate -lock	WRITE_METADATA	
purge	WRITE_CONTENT	
reconnectmod	WRITE_METADATA	
remove	WRITE_CONTENT	
retire	WRITE_METADATA	The retire command both retires and unretires (retire -unretire) objects. Both forms of the command are a metadata write operation.
rmedge	WRITE_METADATA	
rmfolder	WRITE_CONTENT	
rmhref	WRITE_CONTENT	
rmmod	WRITE_CONTENT	
rmvault	WRITE_CONTENT	
rmversion	WRITE_METADATA	
rollback	WRITE_METADATA	

rstcl	WRITE_CONTENT	
setowner	WRITE_METADATA	
setvault	READ_METADATA	
showhrefs	READ_CONTENT	
showlocks	READ_METADATA	
showmods	READ_METADATA	
showproduct	READ_METADATA	Obsolete command. Use entjob show
showstatus	READ_METADATA	
switchlocker	WRITE_METADATA	
tag	WRITE_METADATA	
unfreezemode	WRITE_METADATA	
unlock	WRITE_METADATA	
unremove	WRITE_CONTENT	
upgrade	WRITE_CONTENT	
upload	WRITE_CONTENT	
url commands	READ_METADATA	The only url command that isn't a READ_METADATA or a client-side only operation is url setprop.
url setprop	WRITE_METADATA	The url setprop command can set server properties or client-side local properties. When url setprop is used to set a server property, it uses the WRITE_METADATA operation.
vhistory	READ_METADATA	
whereused	READ_METADATA	

Processing HTTP Proxy Header information Used by the Data Replication System

The mirror system contains an MPD (Mirror Process Daemon) spawned by the RS (Repository Server) in addition to a MAD (Mirror Admin Daemon) spawned by the MAS (Mirror Administration Server). Both the MPD and the MAS processes send transactions using the -ADMIN process type. For example, when mirrors are created, the MAD sends a WRITE_CONTENT-ADMIN operation type in the header.

During enterprise object synchronization tasks, the update process may do some local processing and so send a READ_CONTENT-USER operation type followed by the WRITE_CONTENT-ADMIN operation used to update the mirror.

When a mirror needs to be updated, the MAD spawns a MUP (mirror update process) to update the mirror. Operation types sent from a MUP use the -MIRROR process type. for example, when the MUP runs a populate operation to update the mirror, it sends the READ_CONTENT-MIRROR operation type, when it creates or removes a submirror, the MUP sends a WRITE_CONTENT -MIRROR operation type. .

Automatic Proxy Discovery Hooks

You can create a trigger that lets any client program you run discover which proxy to use for any server, if one exists. The client trigger for this event is fired once for each server contacted during a DesignSync session.

The trigger type is `proxyDiscoveryHook` which uses these input variables:

`serverName`

`serverIP`

The `serverIP` should be in the dot-numeric notation. For example 10.1.3.3.

The following return variable is used:

`proxyNamePort`

The following table lists the possible return values for `proxyNamePort`:

<code>hostname:port</code>	Indicates the <code>server:port</code> pair to use as a proxy.
<code>UNKNOWN</code>	Indicates that the default value for the proxy <code>name:port</code> pair is used for contacting the server.
<code>NONE</code>	Indicates that no proxy should be contacted for the server.

Trigger example for proxy discovery

This trigger contacts local servers directly and uses default proxies outside the local network.

The script assumes that the local IP addresses are seen on the LAN starting with 10.

```
trigger create localProxyHook\
-require type proxyDiscoveryHook \
-tcl_script {
if {[string match 10.* $serverIP]} {
set proxyNamePort NONE
} else {
set proxyNamePort UNKNOWN
}
output ""
}
```

Note

The following line results in no output from the `proxyNamePort` value

```
output ""
```

Related Topics

About Http Proxy

Setting up User Profiles

What Are User Profiles?

User profiles are added by a system administrator or project leader for the purpose of registering each team member who will be using DesignSync. A user profile is associated with a specific server. You may create multiple user profiles for a user who accesses more than one server.

A user profile includes the following information:

- Full name
- User name (login name)
- Email address
- Password
- Phone number (optional)
- Pager number (optional)

Once the first user profile is created, DesignSync's authentication layer is activated. At that point, DesignSync denies access to any user who does not have a user profile.

DesignSync maintains its own user profile database. However, you can configure DesignSync to import users from the LDAP database or 3DPassport database into DesignSync's user database. See [Enabling LDAP](#) or [Enabling 3DPassport](#) for details.

DesignSync does not recognize a user's UNIX user names and passwords. However, we recommend that you use the same names and passwords used by UNIX for your user profiles. This way, users are less likely to forget the information. In addition, the DesignSync design management product uses UNIX user names for workspace management.

No one can extract a password from DesignSync. If a user forgets his or her password, a project leader or administrator with the appropriate access rights must change it (see [Changing User Passwords](#)).

Related Topics

Creating User Profiles

Editing and Deleting User Profiles

Cleaning Up References to Deleted Users

Creating User Profiles

You can use the Create New User Profile panel to add new users to DesignSync. To add a new profile:

1. Select **Add** from **User Profiles** on the **Admin Menu** of the DesignSync web client.
2. In the **Create New User Profile** panel, enter the full name of the user.
3. Enter the individual's username. A username is required to enable the web interface to DesignSync. The username should consist of alphanumeric characters. Do not include single (') or double quotation marks ("), spaces, forward slashes (/), dollar signs (\$), ampersands (&), question marks (?) or leading dashes (-) in user names. Each username must be unique. If you attempt to create a profile with a username that is already in use, you will get an error message. To proceed, select a different username.
4. Enter the user's email address.
5. Enter a password for the user.
6. Re-enter the password to verify that you entered it correctly the first time.
7. Optionally, enter a phone number and pager number for the user.
8. Click on the **Create Profile** button.

If you prefer to use the command-line interface to create your user profiles, see the user create command in the ENOVIA Synchronicity Command Reference.

Importing Existing User Profiles and Passwords

DesignSync offers you a number of ways to authenticate users who are not in the user database.

If you use LDAP to store user information, you can configure DesignSync to import users from an LDAP database into DesignSync's user database. See [Enabling LDAP](#) for details.

If you use 3DPassport/CAS to authenticate users, you can enable 3DPassport and use those user definitions. See [Enabling 3DPassport](#) for details.

For other user databases, you can set up user authentication by creating a trigger that fires whenever a user tries to access an area of DesignSync that requires login information. See [Creating User Authentication Scripts](#) for details.

You also can create a user profile with the same login name and password as in your system's `/etc/passwd` file using Tcl commands (ProjectSync version 2.0.1 and higher). Follow the steps below :

1. Create the script `importUser.tcl`, in one of the following custom directories:
 - o `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/tcl` (server-specific).
 - o `<SYNC_SITE_CUSTOM>/share/tcl` (site-wide).

For more information on creating a Tcl script to import users, view the sample script documented in [Tcl Scripts for Importing Users](#).

2. In order to use this script, you must either design your script to use a security token to validate the authenticity of the request, or you must disable CSRF security by copying the `TclScript.tcl` script to the custom directory and removing this line: `validateDialogToken`.
For information about designing your script to use secure panels, see *ProjectSync Advanced Customization Guide: Securing Custom Panels*.
3. For each user profile to be added, invoke this server-side Tcl script as:

```
http://<host>:<port>/scripts/isynch.dll?panel=TclScript&
file=importUser.tcl&user=username
```

For more information on creating Tcl scripts see [The stcl Programmer's Guide](#).

Related Topics

[Editing and Deleting User Profiles](#)

[Enabling LDAP](#)

[Enabling 3DPassport](#)

[Changing User Passwords](#)

Changing User Passwords

The DesignSync Web UI can change users' passwords, in addition to their other profile attributes. By default, anyone who can edit other users' profiles can also change their passwords. We recommend that you set access rights to limit this capability to key

personnel such as system administrators. To do this, see the ENOVIA Synchronicity Access Control Guide.

Related Topics

Editing and Deleting User Profiles

Editing and Deleting User Profiles

The Edit User Profile panel lets you to view the current privacy policy, view and modify information for DesignSync users or delete users.

Depending on how your installation is set up, you may be able to view or edit only your own user profile or the profiles of other team members. If you do not have the permissions to delete users, when you open the User Profile, you see a simplified form that doesn't provide delete and maintenance options

Note: If a user declines to accept the privacy policy, they will not be able to perform operations using the DesignSync Web UI.

To access the Edit User Profile Panel, select **Edit** under **User Profiles** in the **Admin** menu of the DesignSync Web UI, or if you do not have edit or delete privileges for other user profiles, select **Edit User Profile** from the Me Menu.

To edit or delete a user profile:

1. From the **All Profiles** field, select the name of the profile you want to edit.
2. After choosing a profile, you can:
 - Change the profile attributes and save your changes by clicking **Modify Profile**. (See Changing User Passwords).
 - View the privacy policy for the server.
 - Delete the profile by clicking **Delete Profile**. Deleting a user profile removes the user's access to the DesignSync Web UI.

When you click **Delete Profile**, the user profile is removed (after you confirm the deletion). You then get the Deleted User Cleanup panel, where you can remove or reassign references to the deleted user (see Cleaning Up References to Deleted Users).

3. If you prefer to use the command-line interface to delete a user profile, refer to the user delete command in the *ProjectSync User's Guide: ProjectSync Tcl Commands* topic.

If you need to remove or reassign references to an already deleted user, select **Clean up remnants from a previously deleted user**. This link brings up a window where you enter the user name of the deleted user. When you click **Submit**, the main Deleted User

Cleanup panel appears and you can clean up references to the deleted user (see [Cleaning Up References to Deleted Users](#)).

The **Clean up remnants from a previously deleted user** link does not appear if you do not have permission to modify other users' profiles.

Related Topics

[Creating User Profiles](#)

[Changing User Passwords](#)

[Cleaning Up References to Deleted Users](#)

Cleaning Up References to Deleted Users

The Deleted User Cleanup panel lets you remove or reassign references to a deleted user.

To access the Deleted User Cleanup panel:

1. From the **User Profiles** section of the **Admin** menu, select **Edit**.
2. From the Edit User Profile panel (see [Editing and Deleting User Profiles](#)), either:
 - Select an existing user and click **Delete Profile**.
 - Select **Clean up remnants from a previously deleted user** to deal with references to an already deleted user.

The Deleted User Cleanup panel lists all references to the deleted user. You can clean up email subscriptions for the deleted user and references to the user in notes.

Cleaning Up Email Subscriptions

In the **Email Subscriptions Options** section, you can choose:

- **Remove all email subscriptions for this user** - Deletes all of the user's email subscriptions. If you select only this option, deleted users can still get email notifications when their user names are added to any field used to create a recipient list (e.g., **CC List, Responsible**).
- **Add user to Email Suppression list** - Suppresses all email to this user. If you select only this option, the deleted user's email subscriptions are preserved, but no email notifications are sent.

If the user is already on the email suppression list, this field displays **Remove user from Email Suppression list**. If you select this option, the user can receive email notifications.

Cleaning Up References in Notes

Following the email subscription options is a list of all references to the deleted user in notes. Such references include notes where the deleted user is listed as the author, is listed in another role (e.g., responsible), or is on a CC list:

- For notes where the deleted user is listed as the author, you can reassign authorship to another user by:
 - Selecting the **Reassign** checkbox.
 - Choosing the new author name from the **Select a User** pulldown menu.

If you do not select another author, **"(deleted)"** appears following the user's name on the note.

- For notes where the deleted user is listed in a user list field (for example, **Responsible** or **Design Manager**), you can reassign responsibility to another user by:
 - Selecting the **Reassign** checkbox.
 - Choosing a new person from the **Select a User** pulldown menu.
- For notes where the user is on a CC list, you can select the **Remove** checkbox to delete the user's name.

Click **Submit** to remove and modify references to the deleted user.

Related Topics

[Creating User Profiles](#)

[Changing User Passwords](#)

[Editing and Deleting User Profiles](#)

Creating User Authentication Scripts

You can create a custom Tcl script to import users into DesignSync from another user database. In your script, you create users as part of an authentication event that includes the `user create` command.

If your company uses LDAP (Lightweight Directory Access Protocol), you can use built-in LDAP client to import users from an LDAP database. See [Enabling LDAP](#) for details. You need a custom user authentication script only when your user database is not LDAP. If you use both a user authentication script and the LDAP client, the script is called before the LDAP client is invoked.

Note that DesignSync does not allow spaces, accents, apostrophes or quotes in the user profile name (that is, the login name). If a user's login name contains any of these special characters, he or she will not be able to access DesignSync. Before importing a user database, remove these characters or replace them with underscores.

To set up user authentication, you create a trigger that fires whenever a user tries to access an area of DesignSync that requires login information. A special DesignSync event, `externalAuthHook`, is provided to enable user authentication.

Registering Your Trigger Script

To integrate your user authentication trigger script with DesignSync, copy it to one of the following custom directories:

- Server-specific:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/tcl
```

- Site-wide script directory:

```
<SYNC_SITE_CUSTOM>/share/tcl
```

The custom directories are not overwritten when you install new versions of DesignSync.

To execute the script, enter a URL script request in your web browser:

```
http://<host>:<port>/scripts/isynch.dll?panel=TclScript&file=<script>
```

For example, if your user authentication script is called `AuthHook.tcl` and your server is zen on port 2647, you would enter:

```
http://zen:2647/scripts/isynch.dll?panel=TclScript&file=AuthHook.tcl
```

IMPORTANT: In order to use the TclScript panel, you must either design your script to use a security token to validate the authenticity of the request, or you must disable CSRF security by copying the `TclScript.tcl` script to the custom directory and removing this line: `validateDialogToken`.

For information about designing your script to use secure panels, see *ProjectSync Advanced Customization Guide: Securing Custom Panels*.

After you issue this URL, the authentication script executes the trigger when unknown users attempt to log into DesignSync.

You also can execute your script using the `rstcl` command from a DesignSync client. See *Working with Server stcl Scripts* in the ENOVIA Synchronicity stcl Programmer's Guide for more information on setting up and running server-side scripts.

Related Topics

User Authentication Hook

Creating Note Objects Triggers

Enabling LDAP

Making Advanced LDAP Settings

Enabling LDAP

DesignSync includes an optional LDAP (Lightweight Directory Access Protocol) client. You can configure DesignSync's LDAP client to import users from an LDAP database into DesignSync.

The LDAP tab page on the Administer Server Settings panel lets you set up DesignSync to import users from an LDAP database.

Note: DesignSync does not allow spaces, accents, apostrophes or quotes in the user profile name (that is, the login name). If a user's login name contains any of these special characters, he or she will not be able to access DesignSync. Before importing users from an LDAP database, remove these characters or replace them with underscores.

To access the LDAP tab page:

1. In the DesignSync Web UI, click Admin Menu: | **Administer Server**. This menu option appears only when you have permission to perform administrative functions. See the ENOVIA Synchronicity Access Control Guide for details.
2. On the Administer Servers: Choose Operation panel, click **Server Settings**.
3. In the Enter Network Password dialog box, enter your ProjectSync user name and password.
4. On the Administer Server Settings panel, click the **LDAP** tab in the dialog box.

To make optional LDAP client settings (such as setting an expiration time for imported users) or to make settings for an entire site, you can manually edit the registry files. (See *Advanced LDAP Settings* for details.)

If your user database is not LDAP, you can write a custom script to authenticate users. See [Creating User Authentication Scripts](#) for details.

Configuring LDAP

The first section of the LDAP tab page lets you make the following settings for the LDAP client on a DesignSync server:

- **Enabled** - Enable this checkbox to turn on the LDAP client.

When a user attempts to log in, DesignSync first checks whether the user name and password are stored in its internal database. If the LDAP client is enabled and the user is not found internally, DesignSync searches your LDAP database(s) for the user name and password.

If the user name and password are found in an LDAP database, the user is temporarily imported into DesignSync's user database. By default, users are imported for ten minutes. See [Making Advanced LDAP Settings](#) for details on how to change this expiration time.

A user name must be in the same case each time the user logs in. If a user logs in as "Jane" in one instance and "jane" in another, DesignSync will create two distinct user accounts.

When this checkbox is disabled, the other fields on the LDAP tab page are grayed-out.

- **Servers** - Enter one or more LDAP servers to search for user data.

List LDAP cleartext servers in the `<host>:<port>` format using the `ldap://` designation and either the server's DNS name (for example, `ldap://your.company.com:389`) or its IP address (for example, `ldap://127.0.0.1`). The default port 389 is used for `ldap://cleartext` hosts.

List LDAP SSL servers in the `<host>:<port>` format using the `ldaps://` designation and either the server's DNS name (for example, `ldaps://your.company.com:636`) or its IP address (for example, `ldaps://127.0.0.1`). The default port 636 is used for `ldaps://SSL` hosts.

Separate server entries with spaces. If you enter more than one server, the search order is from left to right.

- **Base Query DN** - Enter the DN (distinguished name) for the location in the LDAP server hierarchy tree where the user search should begin.

The server name specified in this field must be in the same form as it was entered in the **Servers** field. For example, if you entered `your.company.com:389` in the **Servers** field, you would enter `o=your_company.com` in the **Base Query DN** field to begin the search at the top of the LDAP server hierarchy tree.

- **Protocol Version** - Select your LDAP protocol version by clicking on one of the following radio buttons (If you don't know which version you are using, click Auto detect):

- Auto detect
- 2
- 3

Click **Apply** to set the appropriate value.

The **Server Query Authentication** section of the LDAP tab page lets you specify how ProjectSync's query to the LDAP server is authenticated. Depending on how your LDAP server is set up, you can select one of the following:

- **Bind Anonymously** - Queries from the ProjectSync LDAP client are accepted without requiring a DN and password to access the LDAP server.
- **Bind Using a Distinguished Name** - Queries from the ProjectSync LDAP client require a DN and password to access the LDAP server. If you choose this option, you must enter:

- **Bind DN** - The DN required for permission to query the server. For example, you could specify a DN like the following:

```
uid=admin, ou=it, o=your_company.com
```

- **Password** - The password for the user specified in the **BindDN** field.

The LDAP tab page sets server-specific LDAP client settings. To enable the LDAP client for all the servers on your site, you need to manually edit the site registry files. See [Advanced LDAP Settings](#) for details.

Related Topics

[Creating User Authentication Scripts](#)

[Advanced LDAP Settings](#)

Advanced LDAP Settings

DesignSync includes an optional LDAP (Lightweight Directory Access Protocol) client that imports users from an LDAP database into DesignSync. You can configure the LDAP client in the registry files.

To make advanced LDAP settings, you manually edit the `PortRegistry.reg` or `SiteRegistry.reg` file. For example, you can create registry settings to specify how long LDAP user authentication remains in the DesignSync user database and to specify how those users are imported.

If your user database is not LDAP, you can write a custom script to authenticate users. See [Creating User Authentication Scripts](#) for details. If you want to move the DesignSync user authentication process to an LDAP server without issuing new passwords, see the section [Using LDAP for User Authentication](#) that follows.

You create registry settings for your entire site or for specific servers by editing the applicable registry file in the installation's custom area:

- Site-wide:

```
<SYNC_CUSTOM_DIR>/site/config/SiteRegistry.reg
```

- Server-specific:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/PortRegistry.reg
```

You use the `SiteRegistry.reg` file to make LDAP settings for an entire site. For a specific server, you can make the most common settings in the LDAP tab page of the [Administer Server Settings](#) panel (see [Enabling LDAP](#)) and make additional settings in the `PortRegistry.reg` file. (See [Overview of Registry Files](#) for more information on registry files.)

In the registry files you can:

- Set up the LDAP client
- Specify filters for user searches
- Map alternate attribute names

Important: After you change a registry file, you must restart your server.

Setting Up the LDAP Client

To set up the LDAP client, you can specify registry values that:

- Enable the LDAP client (`IsEnabled` registry value)
- Specify server hosts (`ServerURL` registry value)
- Specify the start of the search path (`BaseDN` registry value)

- Specify the DN (distinguished name) (`ClientDN` registry value) and password (`ClientPassword` registry value) required for permission to query the LDAP server
- Restrict user import from an LDAP server into the DesignSync database (`IsUserImportEnabled` registry value)
- Set an expiration time for imported users that limits how long the authorization of imported users remains in the DesignSync database (`ImportedUserExpireTime` registry value)
- Set limits on how many imported user entries are held in cache (`UserCacheSize` registry value)
- Set limits on how long entries remain in cache (`UserCacheTTL` registry value)
- Set time-out limits for LDAP server requests (`RequestTimeout` registry value)
- Allow case insensitive matching of login names (`IgnoreUserNameCase` registry value)

These settings are discussed in the following sections.

IsEnabled

Set the DWORD value `IsEnabled` to 1 in the registry to turn on the LDAP client or to 0 to turn off the client. The default value is 0. To enable user import, specify the following in your registry file:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
IsEnabled=dword:1
```

When a user attempts to log into DesignSync, it first checks whether the user name and password are stored in its internal database. If the LDAP client is turned on and the user is not found internally, DesignSync searches your LDAP database(s) for the user name and password. If they are found, the user is temporarily imported into the DesignSync user database.

Note that when you disable LDAP, users whose profiles had been previously imported will still be able to access the server until the time-out period expires. To limit this problem, set the import time-out period to a very low value (default is 10 minutes). See the section in this topic on `ImportedUserExpireTime`.

You can enable or disable LDAP for individual servers using the LDAP tab of the Administer Server dialog box (see [Enabling LDAP](#)).

Important: DesignSync does not allow spaces, accents, apostrophes or quotes in the user profile name (that is, the login name). If a user's login name contains any of these special characters, he or she will not be able to access DesignSync. Before importing a user database, remove these characters or replace them with underscores.

ServerURL

Use the string value `ServerURL` to specify one or more LDAP SSL servers (`ldaps://`) or cleartext hosts (`ldap://`) to search for user data.

For example:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\ServerURL
="ldap://ldap.company.com:1389 ldaps://ldap.company.com:1636"
```

If you do not specify a port number, the default port 389 is used for `ldap://<cleartext hosts>` and the default 636 is used for `ldaps://<SSL hosts>`. Separate server entries with spaces. If you enter more than one server, the search order is from left to right. DesignSync retrieves users from the first server that it can connect to so you can use this list to designate primary and backup servers.

You can make this setting for individual servers using the LDAP tab of the Administer Server dialog box (see Enabling LDAP).

BaseDN

You use the string value `BaseDN` to specify the DN (distinguished name) for the location in the LDAP server hierarchy tree where the user search should begin. This setting is required for successful user searches. For example, to begin the search at the top of the LDAP server hierarchy tree, you might specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\
  BaseDN="o=your_company.com"
```

The server name in the `BaseDN` setting must be in the same form as it was entered in the `Hosts` setting. For example, if you entered `zen.your_company.com:389` as the `Hosts` setting, you would enter `o=your_company.com` as the `BaseDN` value.

You can make this setting for individual servers using the LDAP tab of the Administer Server dialog box (see Enabling LDAP).

ClientDN

You specify the string value `ClientDN` when queries from the DesignSync LDAP client require a DN and password to access the server. The `ClientDN` value is the DN required for permission to query the server. For example:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\
  ClientDN="uid=admin,ou=it,o=your_company.com"
```


If you do not set a `ClientDN` value, DesignSync uses anonymous login, even if `ClientPassword` is specified.

You can make this setting for individual servers using the LDAP tab of the Administer Server dialog box (see Enabling LDAP).

ClientPassword

You specify the string value `ClientPassword` when queries from the DesignSync LDAP client require a DN and password to access the server. The `ClientPassword` value is the password for the user specified in the `ClientDN` setting, for example:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\ClientPassword="admin4it"
```

If you do not set a `ClientPassword` value, DesignSync uses anonymous login, even if `ClientDN` is specified.

Tip: Although you can store the password in either encrypted or plain-text form, you should store it in encrypted form to keep your server secure. If the registry value starts with ":", the client treats it as encrypted and decrypts the value. For example:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\ClientPassword=":WlpsZEZvcC96Wm89"
```

You can encrypt your passwords using the `_encrypt Tcl` command. The `_encrypt` command takes your password in cleartext form as an argument and encodes it for use with LDAP. The command also adds the ':' prefix to the encrypted string, so you can copy the returned value directly to the registry. For example:

```
stcl> _encrypt admin4it
:RkxzYnoxSHZ5Rjg9
stcl>
```

You can make this setting for individual servers using the LDAP tab of the Administer Server dialog box (see Enabling LDAP).

IsUserImportEnabled

When the LDAP client is enabled (see `IsEnabled`) and `IsUserImportEnabled` is not set or is set to 1 in a registry file, users in an LDAP database are imported into the DesignSync user database. You can set the DWORD value `IsUserImportEnabled` to 0 to restrict how users are authenticated:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\
  IsUserImportEnabled=dword:0
```

When `IsUserImportEnabled` is set to 0, only the following users can access Project Sync:

- Users registered normally in the DesignSync user database.
- Users registered in the DesignSync database with a disabled password. In this case, if the user is authenticated on an LDAP server, the user is allowed to access DesignSync.

Note: DesignSync does not allow spaces, accents, apostrophes or quotes in the user profile name (that is, the login name). If a user's login name contains any of these special characters, he or she will not be able to access DesignSync. Before importing a user database, remove these characters or replace them with underscores.

ImportedUserExpireTime

When `IsUserImportEnabled` is enabled (the default setting), you can use the DWORD value `ImportedUserExpireTime` to specify how long the authorization of imported users remains in the DesignSync database. When a user's authorization expires, the user's profile remains in the DesignSync database. If the user logs in again and is authorized, the user profile is updated with a new authorization expiration time; if the user is not authorized, the user profile is deleted from the DesignSync database.

The authorization expiration time is expressed as a number of seconds in hexadecimal. For example, to authorize a user for two minutes, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
  ImportedUserExpireTime=dword:78
```

Instead of a time in seconds, you also can specify one of the following special values:

- 0 - User authorization expires immediately, so that users are always checked on the LDAP server.
- -1 - User authorization never expires.

If you do not specify an `ImportedUserExpireTime` value, the default time is 600 seconds (10 minutes).

The `ImportedUserExpireTime` value applies only to users imported from an LDAP server, not to users entered into the user database via DesignSync.

UserCacheSize

You can use the DWORD value `UserCacheSize` to limit the number of entries in DesignSync's LDAP cache. The cache stores both user data and whether the user was authenticated. When a new entry exceeds the specified cache size, the oldest entry in the cache is removed.

For example, to hold 250 entries in the DesignSync LDAP cache, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
  UserCacheSize=dword:250
```

If you do not specify a `UserCacheSize` value, the default number of entries is 100.

UserCacheTTL

You can use the DWORD value `UserCacheTTL` to specify how long imported user entries remain in LDAP cache. The value is the number of seconds, expressed in hexadecimal. For example, to hold user entries in cache for 20 minutes, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
  UserCacheTTL=dword:4B0
```

If you do not specify a `UserCacheTTL` value, the default time is 300 seconds (5 minutes).

RequestTimeout

You can use the DWORD value `RequestTimeout` to specify how many seconds elapse before DesignSync queries to the LDAP server are timed out. The number of seconds is expressed in hexadecimal. For example, to make server requests time out after 45 seconds, you could specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
  RequestTimeout=dword:2D
```

If you do not specify a `RequestTimeout` value, the default time is 120 seconds.

Using LDAP for User Authentication

If you disable your users' passwords on the SyncServer, you can move the DesignSync user authentication process to an LDAP server without issuing new passwords.

The easiest way to set all your users' passwords to disabled is to use a server-side Tcl script. The script should set the user property "ClearKey" to "Synchronicity-PasswordDisabled".

For example,

```
foreach user [url users "sync:///"] {  
    url setprop $user ClearKey "Synchronicity-PasswordDisabled"
```

}

If this property has a value "Synchronicity-PasswordDisabled" the server will go to LDAP in order to authenticate the user. If the LDAP query is successful and LDAP import is enabled (see the section `IsUserImportEnabled`), the disabled password is replaced with the active one from the DesignSync user database.

If LDAP import is disabled, the user entry is not updated and therefore the password is not replaced.

IgnoreUserNameCase

By default, a user's DesignSync user name is case sensitive and should match an LDAP account. To make the user name case insensitive, specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\
Auth\IgnoreUserNameCase=dword:1
```

When `IgnoreUserNameCase` enabled on the server, the case of the user names for all DesignSync operations will adopt any (case insensitive) matching DesignSync user profile, if one exists, even when using `AuthUnm` authentication. The potential name case differences need to be accounted for in server side custom code and access controls.

When the user name and password authentication (`AuthPwd`) is in use, the DesignSync user name and password must correspond to a DesignSync user profile. For more information on access-control actions for user authentication, see [User Authentication Access Controls](#).

Specifying User Search Filters

You can use the string value `Filter` to restrict how searches of the LDAP database are performed. Only users who meet the filter requirements are imported into the user database. For example, to limit user authentication to members of the IT department, you might specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\
LDAPStrings\Filter="(ou=it) "
```

Mapping LDAP Attribute Names

When your LDAP server uses alternate names for standard attributes, you must set DesignSync to recognize your alternate attribute names. You can map your custom LDAP attributes to the default names (for example, `uid` or `mail`) in the DesignSync registry files.

UID

You can use the string value `UID` to modify the default LDAP attribute name, `uid`, for a user ID. For example, if you use `userid` rather than `uid`, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
LDAPStrings\UID="userid"
```

Email

You can use the string value `Email` to modify the default LDAP attribute name, `email`, for a user's email address. For example, if you use `e-mail` rather than `email`, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
LDAPStrings>Email="e-mail"
```

CN

You can use the string value `CN` to modify the default LDAP attribute name, `cn`, for a user's full name. For example, if you use `name` rather than `cn`, you would specify:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\LDAP\  
LDAPStrings\CN="name"
```

Related Topics

[Enabling LDAP](#)

[Using the Administer Server Panel](#)

Enabling 3DPassport

DesignSync includes a method for optionally using the 3DPassport/Central Authentication Server (CAS) to authenticate users across your entire 3DEXPERIENCE platform. Using 3DPassport, you can log in once, from any type of client on your machine and have access to all the tools you use without having to log into each system separately. For example, if you access your DesignSync server through the DesignSync GUI client, you do not need to login again if you launch DSclipse, or `dssc`, the command line interface; you are already authenticated. This authentication persists until either the session times out, as defined here, or the 3DPassport login times out defined by the 3DPassport administrator.

The Web apps and the DesignSync Web UI also share a single-login, meaning that, although they are not automatically authenticated after using the DesignSync client applications, logging into any 3DEXperience web application, including the DesignSync WebUI, will allow you to access all the web applications without reauthentication.

Note: When 3DPassport is enabled, no other login types (for example, user logins created in DesignSync or available through LDAP) can be used for user authentication, but they are still visible for other user-related activities, such as email subscriptions, and still show up in drop-down lists, etc, such as in Access Control Administrator.

When 3DPassport is enabled, DesignSync imports the username record from the 3DPassport server when the user logs in. The fields imported include Full Name, Username, Email, and Phone Number. The profiles are automatically refreshed on login and session timeout. If a profile imported from CAS has the same username as an internal user profile, all fields of the internal profile are updated (overridden by the imported profile) except the password field, which is retained. While 3DPassport is enabled, the password defined on the CAS server is used to validate a user's credentials. If 3DPassport is disabled, the previously retained DesignSync password becomes active again.

You can enable 3DPassport from the Administer Server Settings panel in the DesignSync WebUI.

TIP: When you enable 3DPassport, ideally your DesignSync administrator username should not match any usernames on the 3DPassport server. Using a name like syncadmin or designsyncadmin or using a special prefix that is unique to the DesignSync server can help avoid confusion.

To access the 3DPassport tab page:

1. In the DesignSync Web UI, click **Admin Menu: | Administer Server**. This menu option appears only when you have permission to perform administrative functions. See the ENOVIA Synchronicity Access Control Guide for details.
2. On the Administer Servers: Choose Operation panel, click **Server Settings**.
3. On the Administer Server Settings panel, click the **3DPassport** tab in the dialog box.
4. Specify your settings and click **Apply**. When you apply the changes, you will be logged out and required to log back in with your 3DPassport credentials.

Configuring 3DPassport

Using the 3DPassport panel, you enter the settings needed for your DesignSync system to use the 3DPassport server.

The screenshot shows a configuration window with four tabs: 'RC Notes', '3DPassport', 'LDAP', and 'Vault Types'. The '3DPassport' tab is active. Below the tabs, there is a checkbox labeled 'Use 3DPassport Authentication'. Underneath is a text input field for '3DPassport URL' containing the text 'https://server:port'. Below that is a spinner control for 'Design Sync Session Idle Timeout (in hours)' set to '12'. At the bottom right of the window are two buttons: 'Apply' and 'Help'.

Use 3DPassport Authentication

Click this checkbox to use 3DPassport Authentication.

3DPassport URL:

Enter the URL, including the port number of the 3DPassport URL. Enter the URL like this:

`https://<servername>:<port>`

For example:

`https://3dpassport.ABCo.com:9943`

DesignSync Session Idle Timeout (in hours):

Enter the idle timeout value. When a DesignSync client session contacts the SyncServer and is authenticated, the idle time is set. If the idle time expires, then the user may need to reauthenticate. The default Idle Timeout value is 12 hours.

Disabling 3DPassport

If you disable 3DPassport, all user profiles imported from 3DPassport server will be unavailable for user authentication. The password fields will be cleared, and the system administrator will need to define new passwords through the Edit User Profile panel if those accounts are going to be used. The profiles will still be available in drop-down menus and for email subscriptions, etc.

Access Control Overview

As an administrator, you can limit access to certain DesignSync operations, or **actions**.

For example, you might not want all users to have the ability to tag versions "Golden", or you might require that all checkins include check-in comments. You can control

access to most DesignSync actions such as checking in files, checking out files, removing locks, tagging files, retiring files, and deleting versions from a vault. In addition, you can specify the level of user authentication required when accessing DesignSync data.

For more information on access controls, see the ENOVIA Synchronicity Access Control Guide.

RevisionControl Notes

RevisionControl Notes Overview

A **RevisionControl note** is a **note** created automatically when a DesignSync revision control operation takes place.

As project leader or DesignSync administrator, you can enable DesignSync to generate RevisionControl notes during revision-control operations such as check in, check out, and populate. These notes have the built-in type **RevisionControl** and contain information about the operation, such as the name of the user, the command used to invoke the operation, and the time of command execution. The RevisionControl note also includes an Objects field that lists the objects affected by the operation. In addition, RevisionControl notes generated for a check-in operation include both the comment specified for the check-in operation and the comment specified when the object was checked out.

DesignSync generates one note for each revision control operation even if the operation affects multiple files, such as a recursive check in.

Once you enable the generation of RevisionControl notes, you can:

- Trigger some other action, such as running a Tcl script. Often, system administrators set up the generation of RevisionControl notes to trigger the execution of a Tcl script, included in the DesignSync distribution, that sends automatic email notification of the operation.
- View the notes and run queries on the information they contain.
- Generate statistics on the notes for use in reports.

You can specify how RevisionControl notes are counted for statistics purposes. For example, if there is a recursive checkin operation that checked in 100 files, you can determine whether that operation is counted as one checkin (and therefore one RevisionControl note) or 100 checkins.

By default, the statistics function counts RevisionControl notes. For information about changing this default behavior, see Revision Control (RC) Notes Options: Count objects operated on when calculating server statistics

Notes:

- DesignSync note generation is available for remote vaults only. You cannot enable note generation for local (client) vaults.
- Adding or editing RevisionControl notes is not enabled by default.

Related Topics

Enabling the Generation of RevisionControl Notes

Troubleshooting Email Notification of RevisionControl Notes

Configuring Email Notifications

Adding the Email Trigger

Revision Control (RC) Notes Options

The **Revision Control Notes** options pane allows you, the DesignSync tools administrator, to enable DesignSync to generate RevisionControl notes during revision-control operations such as check in, check out, and populate. By default, RevisionControl notes are disabled.

When you enable RevisionControl notes, DesignSync generates one note for each revision control command even if the command affects multiple files, such as a recursive checkin.

DesignSync does not attach notes to objects. However, an Objects field on the RevisionControl note lists the objects that have been changed by the revision-control operation and the resulting e-mail notification includes the list. (For information on setting up this e-mail notification, see [Configuring Email Notifications](#)).

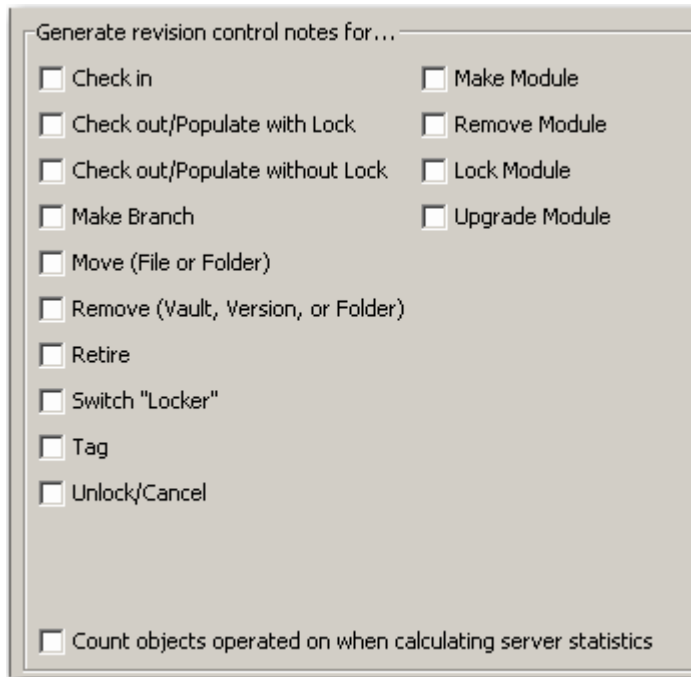
RevisionControl notes generated for a check-in operation include both the comment specified for the check-in operation and the comment specified when the object was checked out. These comments are also included in the automatic e-mail notification of the RevisionControl note.

Notes:

- When you check-in an object with the -reference option and only the state of the object has changed, DesignSync does not add the object to the Objects field on the RevisionControl note.
- DesignSync note generation is available for remote vaults only. You cannot enable note generation for local (client) vaults.

- Generation of revision-control notes is disabled by default because the operations for which notes are generated is a system administration preference.

Click on the image for more information about each field.



Commands

Check In

When an object is checked in, a snapshot of a design object is stored in a vault. The new snapshot is called a version. As a LAN administrator, you can choose if you want a RevisionControl note to be generated for this DesignSync operation. (See DesignSync Data Manager User's Guide: Checking in Design Data for more information.)

Check Out or Populate with lock

When an object is checked out by a check-out or populate operation, the object is available in your local work area. You can check out a locked or unlocked copy of the object, a link to a cache or mirror copy of the object, or a locked or unlocked reference to the object.

When an object is checked out or populated with a lock, only the user who has the lock can check in a newer version of the object on that branch. You should use this option when your project team uses the locking model (as opposed to the merging model) and you intend to make changes to an object. To remove the lock use the 'checkin or cancel command.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when users check out or populate objects with a lock. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: Checking out Design Data and Populating Your Work Area.*)

Check Out or Populate without lock

When an object is checked out by a check-out or populate operation, the object is available in your local work area. You can check out a locked or unlocked copy of the object, a link to a cache or mirror copy of the object, or a locked or unlocked reference to the object.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when users check out or populate objects without a lock.

Note: RevisionControl notes are not generated when checking out unlocked references, or links to the cache or mirror. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: Checking out Design Data and Populating Your Work Area.*)

Make Branch

Separate branches may be created for projects that require multiple lines of development (parallel development).

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a new branch is created. (For more information, see *DesignSync Data Manager User's Guide: Parallel (Multi-Branch) Development.*)

Move (File or Folder)

You can use the `mvfile` command to move or rename a specified local file or collection object (for example, a Cadence cell view). You also can use the `mvfolder` command to move or rename a specified local folder.

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a file, collection object, or folder is moved or renamed. See the `mvfile` and `mvfolder` commands descriptions in the *ENOVIA Synchronicity Command Reference* for more information.

Remove (Vault, Version, or Folder)

You can delete vaults, versions in vaults, and folders, for non-module data. As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a vault, version, or folder is removed.

See the `rmvault` and `rmfolder` command descriptions in the *ENOVIA Synchronicity Command Reference* for information on removing vaults and folders. For information on removing versions, see *DesignSync Data Manager User's Guide: Deleting Versions from a Vault*.

Retire

You can retire design objects when they become obsolete. As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when design objects are retired.

For more information, see *DesignSync Data Manager User's Guide: Retiring Design Data*.

Switch "Locker"

An administrator with permission can use the `switchlocker` command to change the lock owner of a design object. (By default, access controls deny this command to all users.)

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when the `switchlocker` command is used to change a lock owner. See the `switchlocker` command description in the *ENOVIA Synchronicity Command Reference* for more information.

Tag

A tag is an identifier that you attach to a group of design object versions that work together as a configuration. You can then perform operations on the group of objects as a whole. For example, you might tag all versions that went into a release "Rel2_0" so that at a later date another user could fetch all versions tagged "Rel2_0" thereby recreating that release.

As a LAN administrator, you can choose if you want a RevisionControl note to be generated when design objects are tagged. (For more information, see the following topics in the *DesignSync Data Manager User's Guide: What is a Design Configuration and Tagging Versions and Branches*.)

Unlock/Cancel

You can unlock objects that have been locked by you or other team members. You can also cancel a lock of objects that you have locked in your own workspace. (For more information, see *DesignSync Data Manager User's Guide: Unlocking Server Data*.)

As a LAN administrator, you can choose to have a RevisionControl note generated when users unlock design objects.

Make Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module is created on the server.

For more information, see *DesignSync Data Manager User's Guide: Creating a Module*.

Remove Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module is deleted from the server.

For more information, see *DesignSync Data Manager User's Guide: Deleting a Module*.

Lock Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a module branch is locked.

For more information, see *DesignSync Data Manager User's Guide: Locking Module Data*.

Upgrade Module

As a LAN administrator, you can choose whether you want a RevisionControl note to be generated when a legacy module is upgraded.

For more information, see *DesignSync Data Manager User's Guide: Importing Legacy Modules*.

Count objects operated on when calculating server statistics

DesignSync generates a single note for each revision control command, however many objects are operated upon.

As a LAN administrator, you can choose whether server statistic reports are generated based on the number of notes or the number of objects recorded within the notes.

OK

When you click on the **OK** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. (In most cases, this is the site registry file.) After saving any changes, SyncAdmin exits.

Note: There may be situations where you want to implement server-specific settings. See [Enabling/Overriding RevisionControl Note Generation for Specific Servers](#) for information.

Apply

When you click on the **Apply** button, any changes you have made are written to the registry file you selected when you started SyncAdmin. (In most cases, this is the site registry file.) After saving any changes, you can continue using the SyncAdmin tool. Note that the **Apply** button is disabled until changes are made in the pane.

Cancel

When you click on the **Cancel** button, SyncAdmin exits. If you have not clicked on the **Apply** button prior to clicking **Cancel**, any changes you have made are not saved.

Help

When you click on the **Help** button, the SyncAdmin help is displayed.

Related Topics

[Enabling Note Generation](#)

[Registry Files](#)

Enabling Note Generation

As DesignSync administrator, you can enable DesignSync to generate a RevisionControl note each time a specified revision control operation takes place. You can then view and query these notes. You can also use the generation of a RevisionControl note to trigger some other action, such as running a Tcl script.

Note: For more information on RevisionControl notes and their uses, see the [RevisionControl Notes Overview](#).

To enable note generation for an entire site, use SyncAdmin to modify the site settings:

1. On the UNIX server machine, start the SyncAdmin tool:

```
%SyncAdmin &
```

2. Select **Change Site Settings**. Then click **OK**.
3. Select **Site Options=>RC Notes**.

4. Turn on the RevisionControl note generation for the command you want and click **OK**. (See the RevisionControl Notes Options for information on each command.)
5. Restart any SyncServers that share the same SYNC_DIR installation directory to have the settings take effect.
6. Any clients that access the server need to be restarted as well, so they recognize the change in the server's RevisionControl note setting.

Notes:

There may be situations where you want to implement server-specific settings. This is possible if your servers are on Unix machines. For more information, see [Enabling/Overriding RevisionControl Note Generation for Specific Servers](#).

Related Topics

[RevisionControl Notes Options](#)

Enabling or Overriding RevisionControl Note Generation for Specific Servers

In general you would implement note generation on a site-wide basis, however there are situations where you might choose to implement server-specific settings. For example, at your site, RevisionControl note generation is implemented site-wide, but for one project (on its own SyncServer) you don't want to generate notes.

You can use SyncAdmin to modify the server's `PortRegistry.reg` file to enable RevisionControl note generation or override site settings.

You can also use ProjectSync's Server Settings panel to remotely modify the RevisionControl note settings for a server. See [ProjectSync User's Guide: Setting RevisionControl Note Generation](#), for details.

To enable RevisionControl note generation specific to a server using SyncAdmin:

1. On the server machine, start SyncAdmin.
2. On the **Select Which Setting** form, select **Change setting stored in a specified file**.
3. Use the Browse button to navigate to the correct `PortRegistry.reg` file located in `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg`.
4. Click **OK**.
5. Click the **RevisionControl Notes** tab to select the commands you want to generate revision control notes for.
6. Turn on note generation for the commands you want and click **Apply**.

Notes

- RevisionControl notes generated for a check-in operation include both the comment specified for the check-in operation and the comment specified when the object was checked out. These comments are also included in the automatic e-mail notification of the RevisionControl note.
 - Although notes are not attached to objects, an Objects field on the RevisionControl note lists the affected objects and the resulting e-mail notification includes the list. For information on setting up this e-mail notification, see [Setting Up Automatic Email Notification of RevisionControl notes](#) .
 - Populating with references does not generate RevisionControl notes.
7. Stop and restart the SyncServer for the registry changes to take effect.

Note

Any clients that access the server need to be restarted as well, so they recognize the change in the server's RevisionControl note setting.

Related Topics

[RevisionControl Notes Overview](#)

[Enabling the Generation of RevisionControl Notes](#)

Setting Up Revision Control Triggers

A revision control trigger is a server-side trigger that executes a Tcl script when DesignSync generates a RevisionControl note. For example, DesignSync administrators often set up the generation of a RevisionControl note to trigger the execution of a Tcl script (provided with DesignSync) that sends automatic email notification of the operation.

For information on notes and triggers, see the [Activating the Email Trigger](#).

Related Topics

[Enabling the Generation of RevisionControl Notes](#)

[Troubleshooting Email Notification of RevisionControl Notes](#)

[Configuring Email Notifications](#)

Working with RevisionControl Notes

A RevisionControl note is a note created automatically by the SyncServer when a DesignSync revision control operation such as check in, check out, and populate takes place.

A RevisionControl note contains information about the operation, such as the name of the user, the command used to invoke the operation, and the time of command execution. A RevisionControl note also includes an **Objects** field that lists the objects affected by the operation. In addition, RevisionControl notes generated for a check-in operation include check-in comments and may also include the comments entered when the object was checked out. DesignSync generates only one RevisionControl note for each revision control operation, even if the operation affects multiple files.

Using the Administer Server **RevisionControl Notes** tab page from the DesignSync web interface, you can enable RevisionControl note generation for a specific server. (See Setting RevisionControl Note Generation for details.) Using the **RevisionControl Notes** tab page in DesignSync's SyncAdmin tool, you can enable RevisionControl note generation for your entire site. (See Enabling the Generation of RevisionControl Notes for details.)

After you enable the generation of RevisionControl notes, you can:

- Use the generation of a RevisionControl note to trigger some other action, such as sending automatic email notifications.
- Search for and view RevisionControl notes.
- Generate statistics on RevisionControl notes for use in reports.

These features are discussed in the following sections.

Understanding Email Notifications

Email notification of RevisionControl notes is enabled through the DesignSync web interface. You specify what revision control operations generate RevisionControl notes using DesignSync's SyncAdmin tool. For more information, see Enabling the Generation of RevisionControl Notes.

When DesignSync generates a RevisionControl note, DesignSync sends email notification to a distribution list compiled from two sources:

- The subscription list - A user subscribes to receive notes for a project through the DesignSync web interface. This adds the user's name to the subscription list for RevisionControl notes for the project. Subscribers are shown in the **cc:** list on the email.
- The **Author** field, if any - The standard RevisionControl note type includes an **Author** field which stores the name of the DesignSync user who performs the revision control operation. Email notifications of revision control events are not sent to the individual in the **Author** field by default. You can

use the Email Administrator (see Configuring Email Notifications) to enable email notifications for the RevisionControl note **Author** field. If you enable email notification for this field, the author's name is shown in the `To:` list of the email.

You can edit the criteria used to send out email notifications and the content of the email using the Email Administrator. (See Configuring Email Notifications for details.)

See Troubleshooting Email Notification of RevisionControl Notes for information on resolving problems with email notifications.

Viewing RevisionControl Notes

You can access and view RevisionControl notes by:

- Searching for notes using one of the options on the **Queries** section of the **ProjectSync** menu in the DesignSync Web UI.
- Selecting **Data Sheet** from the menu to access the data sheet for a project. The project data sheet links to a form for searching for RevisionControl notes for the project.
- If you know the note ID number of the RevisionControl note, you can access it directly using Quick View. (See *ProjectSync User's Guide*: Using Quick View for details.)

A standard RevisionControl note includes a table that lists the note ID number of the note, the date and time of the operation, and the author (the person who performed the operation). The **Title** row shows the commands used to run the revision control operation:

```
RC(<command>) : [<Project/Module>] <files affected>
```

For example, if a user checked in a file named "samp.asm" from the "code" directory for the "Sportster" project, the following information would appear in the RevisionControl note title row:

```
RC(ci) : [Sportster] code/samp.asm
```

If a user creates a Module in his server `tallis:30106`, the following information would appear in the RevisionControl note title row:

```
RC(hcm mkmod) : sync:///Modules/MyMod/Mod1
```

If a user populates his workspace with module `Task1`, the following information would appear in the RevisionControl note title row:

```
RC(populate no lock) : sync:///Modules/Task1;1.5
```

A standard RevisionControl note also includes the following fields:

- **Project** - The name of the project affected by the operation. You can select the hyperlinked project name to go to the data sheet for the project.
- **Configuration** - The name of the configuration affected by the operation. This field is applicable for legacy modules commands only.
- **Module** - The name of the Module affected by the operation. You can select the hyperlinked name to go to the data sheet for the Module.
- **Attached To** - The name of an object, not under the project or module hierarchy, that is affected by the operation. You can click the hyperlinked URL to go to the data sheet for an attached object.
- **Full Command** - The complete command issued for a DesignSync or Module operation.
- **Tag or HCM sub-cmd** - The tag name for a DesignSync tag operation. For other DesignSync operations, the tag field is blank. If the RevisionControl note is for an module operation, the module command, without an hcm prefix, (for example, "mkmod" from the operation "mkmod") is shown.
- **Command** - The command (for example `ci` for check ins) used for a DesignSync operation.
Note: If the RevisionControl note is for a Module, the **Command** value is `hcm`.
- **Completed** - A true or false value indicating whether or not the client operation finished. A value of false signifies an operation still in progress or a problem such as a client crash, network glitch, or power outage.
- **Objects** - A hyperlinked list of the objects and versions that were affected by the operation. You can click the hyperlinked object name to go to the data sheet for the object. The version number of the object is given following a semicolon at the end of the URL.

If the RevisionControl note contains more than 500 objects, the listed objects are not hyperlinked.

- **Additional Information** - Information, such as hierarchical references, recorded by the server during Module operations.

In most cases, you would not want to create or edit RevisionControl notes. By default, access controls prevent all users from adding or editing RevisionControl notes. However, you can amend the access controls to allow adding or editing. (See the ENOVIA Synchronicity Access Control Guide for information on access controls.) If access is enabled, you can click **Edit RevisionControl** from the bottom of the View mode of a note to go to the Edit RevisionControl panel. On this panel you can edit the fields on the RevisionControl note or click **Delete** to remove the note.

Creating Statistical Reports for RevisionControl Notes

Using **ProjectSync | Queries | Statistics** menu in the DesignSync Web UI, you can generate status reports or historical reports based on RevisionControl notes. The Statistics tool lets you produce tables, bar charts, and area charts broken down by category. You also can export the statistical information in the CSV interchange format, which can be imported into many spreadsheet programs. See *ProjectSync User's Guide: Creating Reports with the Statistics Feature* for help on creating statistical reports.

It is important to note that, by default, the statistics function counts the number of RevisionControl notes generated and not the number of objects that are affected by revision control operations. Using DesignSync's SyncAdmin tool, you can choose whether statistic reports are generated based on the number of notes or the number of objects recorded within the notes.

To calculate statistics according to the number of items listed on the **Objects** field of a RevisionControl note, you select **Count objects operated on when calculating server statistics** on the SyncAdmin **RevisionControl Notes Tab** panel.

Related Topics

ProjectSync User's Guide: Using ProjectSync with DesignSync

Mapping RevisionControl Notes

You can use DesignSync to query for notes on an entire module hierarchy regardless of where submodules reside. Submodule servers can have different note types, containing different field names and field values.

If submodule servers use different note types, the DesignSync Administrator must map the upper-level module note types to the note types used on the submodule servers. Mapping note types ensures the successful operation of not only queries but also subscriptions to notes on a module hierarchy.

Note: Mapping of note types ensures successful queries and subscriptions to a module hierarchy only when the user's login is the same as on the referenced server, or when a login has been stored for the user.

To map note types:

1. Use the **showhrefs** operation to identify submodules that reside on servers different from the upper-level module. For example:

```
stcl> hcm showhrefs sync://tallis:30106/Modules/Task1
Beginning showhrefs operation ...

Showing hrefs of module sync://tallis:30106/Modules/Task1 (1.3) ...
sync://tallis:30106/Modules/Task1: Href mode is normal.
-----
Name   url                                     Selector  Version  Type           Relative Path
-----
Task2  sync://tallis:30106/Modules/Task2      Trunk:    1.1      Module         REF1
230    sync://desprez:30012/deliverable/230  Deliverable REF2
-----
Finished showhrefs operation.
```

2. On the upper-level module's SyncServer, use the NoteType Manager to examine the note type structure. At the NoteType Manager, choose **Modify an active note type** and use information in the first two screens displayed.
3. For each submodule on a different SyncServer, log in to that server. Then use the NoteType Manager to view note types and their fields. (To get information on choice and state machine field values, use the Property Type Manager on the NoteType Manager panel.)

From the information displayed, determine the note types and fields you need to map. To complete this step, you will have to know the semantics of how fields are used.

For example, the **showhrefs** operation for Module `Task1` lists the submodule `230` as residing on an IP Gear Publisher server. You know that the Publisher server uses the Ticket note type (based on the `CustTicket` note type). You want a query based on the `DateFixed` field of the `SyncDefect` note

type to use the ClosedDate field in the CustTicket note type.

View SyncDefect 1

Id	Creation Date	Author
1	2007-01-09 14:20:15	Jhon Smith

Title Problem with Integration

View Ticket

Ticket Id	Date Submitted	Author
29	2007-01-09 14:31:54	Master Administrator

Externally Visible Attributes

Attached To [Interface Specification \[1.3\]](#)

Group/Site [Default Group](#)

Title Ticket1

Ticket State closed

Priority Medium

Closed Date 2007-01-09 14:33:34

Customer CC List

Supporting Data

Resolution Data

Ticket Type Enhancement Request

Analyst Assigned Master Administrator

Waiting On Nobody

Original text by **masteradmin** on 2007-01-09 14:31:54-0500 :
Problem with Integration.

From this information, you know that on the `sync://tallis:30106` server, you must map:

- o The SyncDefect note type to the CustTicket note type
 - o The DateFixed field to the ClosedDate field
4. On the SyncServer where the upper-level module resides (`sync://tallis:30106` in the example), create a configuration file named `hcmNoteMap.conf` to contain note type maps. (**Note:** The upper level module's SyncServer has the SyncDefect note type installed.)

To create the file, copy the file `<SYNC_DIR>/share/config/hcmNoteMap.conf` to the custom area of your DesignSync installation directory.

To apply Copy the `hcmNoteMap.conf` file to this location...
values to...

Your entire site `<SYNC_SITE_CUSTOM>/share/config/hcmNoteMap.conf`

(The default definition of the environment variable `SYNC_SITE_CUSTOM` is `<SYNC_CUSTOM_DIR>/site`)

A specific SyncServer `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/config/hcmNoteMap.conf`

where `<host>` and `<port>` are the server's name and port number

(The default definition of the environment variable SYNC_CUSTOM_DIR is <SYNC_DIR>/custom.)

Note: Always copy the DesignSync-provided configuration files to the custom directory before editing them. Do not edit the <SYNC_DIR>/share/config/hcmNoteMap.conf configuration file. Any changes you make to this file will be lost upon upgrading your SyncServer.

5. Modify the site or server hcmNoteMap.conf file you created to map note types, fields, and values. (The file contains syntax and examples to help you.)

For the example (Step 3) showing that the SyncDefect note type should be mapped to the CustTicket note type, you would edit the hcmNoteMap.conf file on Module Task1's SyncServer (sync://tallis:30106) and add the following lines:

```
lappend SyncNoteTypeMap(sync://desprez:30012) { \  
  localntype SyncDefect \  
  refntype    CustTicket \  
  localfield DateFixed \  
  reffield   ClosedDate}
```

Notes:

- Specify the server URL exactly as it appears in the output of the **showhrefs** operation, minus the module path.
- For additional examples, see the hcmNoteMap.conf file.

6. Reset the server by using **Reset Server** from the **Server** section of the **Admin** menu in the DesignSync Web UI.. For more information, **Resetting the SyncServer**.

The mapping takes effect the next time a query is performed.

Limitations of Note Type Mapping:

- You can map multiple elements (note types, fields, or values) on the origin server to one element on the referenced server. But you cannot map one element on the origin server to multiple elements on the referenced server. For example, you can map the BugReport and SyncDefect note types on the origin server to the ProblemReport note type on the referenced server. But you cannot map the SyncDefect note type on the origin server to both the ProblemReport and TroubleNote note types on the referenced server.
- You cannot map substrings for text searches. For example, suppose that on the sync://cpu.ABCo.com:2647 server you map the SyncDefect note

type's EVALUATED keyword to the REVIEWED keyword of the BugReport note type on `sync://srvr1.ABCo.com:2647:`

```
lappend SyncNoteTypeMap(sync://srvr1.ABCo.com:2647) { \
  localntype SyncDefect \
  refntype "BugReport" \
  localfield KeyWords \
  valuemap {EVALUATED REVIEWED}
```

A Standard Query of SyncDefect for the CPU module, specifying **This object and one level below** or **This object and all levels below**, plus Key Words EVALUATED will not yield any matches.

Related Topics

`showhrefs` command

How Email Notification Works

Email notification of RevisionControl notes occurs by enabling RevisionControl note generation in DesignSync, setting up the email trigger in ProjectSync, and (optionally) setting up the email distribution list through the Email Administrator.

When DesignSync generates a RevisionControl note, the system sends email notification to a distribution list. The distribution list from two sources:

- The subscription list

When a user subscribes to receive notes for a project, the user name is added to the subscription list for RevisionControl notes for the project. Subscribers are shown in the `CC:` list on the email. See *ProjectSync User's Guide: Adding Email Subscriptions*.

- The user list fields on the note (in this case the RevisionControl note type).

The default setting for RevisionControl notes in the Email Administrator defines the User field as the sole user list field on the note. The User is the DesignSync user who performed the revision control operation. However, by default email notifications of revision control events are not sent to the user in the User field. You can use Email Administrator available from the DesignSync web interface to enable email notifications for RevisionControl note authors. If you enable email notification for the User field, the user's name is shown in the `TO:` list of the email.

How do you know which type of email notification you are getting?

Knowing how a user came to receive the email notification can be useful information in troubleshooting problems with email notification. You can tell which type of email notification you are getting by looking at the email you receive:

- Users who subscribe for email are listed as recipients on the **CC**: list of the email.
- Users who are added to the **CC**: list for a note are listed as recipients on the **To**: list of the email.

Related Topics

Enabling the Generation of RevisionControl Notes

Troubleshooting Email Notification of RevisionControl Notes

Configuring Email Notifications

Adding the Email Trigger

Configuring Email

The **Email Administrator** option under the **Admin | Triggers** menu brings you to the Email Administrator wizard. You use the panels in this wizard to customize the behavior of email such that DesignSync users have the ability to:

- Receive email notifications when existing notes are modified
- Append to existing notes by replying to email notifications
- Create new notes using email

All the panels of the Email Administrator wizard have the following buttons:

- **Next>>** - Takes you to the next panel in the wizard. (Use your browser's **Back** button to return to a previous panel.)
- **Review** - Displays the Review Changes panel. It lists the changes you made to the original settings.
If you have finished configuring, click **Commit Changes** to save your changes. The Changes Saved panel confirms the changes you made and you exit the wizard. Otherwise, click your browser's **Back** button to continue using the wizard.
- **Index** - Takes you to the Index of Operations panel, where you can quickly move from one Email Administrator panel to another.

The Index of Operations panel contains a list of all the panels in the Email Administrator wizard. Click on a hyperlinked panel name to go directly to that panel.

- **Help** - Displays the online help for that panel.

Enabling or Disabling the Email Trigger

The Email Administrator Global Email Notification Status panel lets you enable or disable the email trigger.

You must enable the email trigger when you install a new version of DesignSync. If you have upgraded from an earlier product version, your previous email triggers are updated and you do not need to enable the email trigger.

To enable or disable the email notification, select either of the following options:

- **Enabled** - Enables global email notification.
- **Disabled** - Disables global email notification.

Note: The email trigger is automatically enabled when you open the Email Administrator wizard to configure email. You can also enable the trigger manually. See the Activating the Email Trigger for information on how to manually enable the standard email trigger.

Disabling the email trigger halts email notification resulting from creation or updating of notes. Other types of email, like welcome messages sent to newly registered users, remain enabled.

After the first time you enable your email triggers, the **Edit Email Trigger** hyperlink appears. See Editing the Email Trigger for more information on the Edit Trigger Define Attributes panel, where you can specify debug settings for your email triggers.

Specifying Domain and Server Information

The Server Identification panel lets you specify the name of the machine used to access the Internet and send email. The fields on this panel automatically display information for the current DesignSync server:

- **Domain Name** - The domain name of the email server - for example, `your_company.com`. When a username is entered using only a login name as an email address, this domain name is appended to the login name to create the email address.
- **External IP Name** - The complete name of the machine that runs the email server - for example, `wiz.your_company.com`, where `wiz` is the machine name.
- **Server Host and Port** - The machine and port number running the DesignSync server. DesignSync uses the host and port information to send email notifications. For port number 2023, your server host and port is:
`wiz.your_company.com:2023`.

If you use a DNS alias for your server, you can enter the alias with the port number in the **Server Host and Port** field instead of the literal host name.

Note: An encrypted SSL port cannot be specified for email subscriptions; you must specify a cleartext port. To maintain secure communication, use the `NonSSL` access control.

If the SyncServer detects that your communications need a secure port, it redirects the DesignSync client software to a secure communications port, as indicated in the `AccessControl` file. See the ENOVIA Synchronicity Access Control Guide for more information.

Specifying the Server Administrator

The Server Administrator panel lets you specify the name of the Administrator responsible for the email notification system. The Administrator receives an automatic email when errors occur.

To specify a server administrator, select a user from the **Administrator** pull-down menu. Your name is the default selection in this field.

Specifying a Mailer

The Configure Email Method panel lets you specify the method for sending email notifications. To specify a method, select one of the following options :

- **SMTP** - (Simple Mail Transport Protocol)
- **program** - The `sendmail` program (or a compatible program).

In general, UNIX platforms use `sendmail` program. If you do not have a `sendmail`-compatible program to dispatch email, you need to choose the SMTP protocol.

Specifying Email Method Parameters

The Configure Email Method Parameters panel lets you specify the program to use when sending email notifications. The options in this panel differ depending on the option you selected in the previous panel:

- If you selected the SMTP mailer, the options are:
 - **Email server** - The machine that distributes email in the following format - `< email_server > . < domain_name > .` For example, `mail.your_company.com`.
 - **SMTP socket** - The socket on the email server that distributes email. This field automatically displays 25, the default port for the SMTP protocol.

- If you selected the sendmail program, the options are:
 - **Program to Execute** - The command (including arguments) that invokes the sendmail program. DesignSync requires a sendmail-compatible program to send email notifications.
 - **Run as background process** - Whether email should be delivered as a background process. Default is **no**. Select **yes** only when you are certain that your email notifications are performing as expected.

Note: Selecting **yes** makes debugging difficult but improves performance as by not waiting for sendmail to return before confirming the notification.

Specifying the Configure Email Interface Parameters

The Configure Email Interface Parameters panel sets the email server information.

Before you can configure this panel, you must save the name and password of the server administrator and make sure you have an email account for the mailbox. The SyncServer needs this account information in order to communicate with the email account.

Registering the Server Administrator

To register the Server Administrator (usually, also the DesignSync Administrator):

1. Log into the unix account for the SyncServer.
2. Make sure that the SyncServer is running.
3. Enter:

```
dssc password -save <host>:<port>
```

For example:

```
dssc password -save wiz.my_company.com:2023
```

4. At the first prompt, enter the user name of the server administrator.
5. At the second prompt, enter the password of the server administrator.

```
Confirm password: <1>
```

The `password` command returns 0 if the password is accepted.

Setting Up a Mailbox

Ask your IT department to create the email account(s) you need. These accounts should be enabled before users can use email to create notes or append to notes. A SyncServer requires a dedicated email account to handle the incoming mail. One email

account is required for each SyncServer, but multiple accounts can reside on the same mail server.

Configuring Email Interface Parameters

The configurable email interface allows your to configure your email server setting and email append settings.

Note: Email appends are used for notes functionality, they are not applicable to RevisionControl notes. For more information, see *ProjectSync User's Guide: What are Email Notifications*.

The configurable email interface provides the following configuration parameters:

- **Email Interface mail server** - The name of the mail server that will handle the incoming mail for this server. For example: `mail.my_company.com`.
- **Email Interface mailbox** - The name of the email account that will receive the incoming mail for this server. For example: `popmail2`.

Note: The email address should be local to the server's domain, that is, valid on the same domain as the SyncServer. You cannot use any external email address, for example, a hotmail or yahoo email address.

- **Email Interface mailbox password** - The password for the email account.
- **Email Interface polling interval** - How often the SyncServer checks for new mail in the email account in seconds. Default is 10 seconds.
- **Email Interface HTML support** - The **Enabled** radio button allow email appends in HTML, as generated by an email tool. Default is **Disabled**. It allows email appends in plain text only.
- **Mail server protocol** - The protocol to use to communicate with the mail server. Default is **AUTO** mode which attempts to connect to the mail server using the IMAP, POP3, or POP2 protocols, in that order. This default setting should work for most server configurations.
However, if you get timeout error messages in your `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log` file, you may want to set the protocol manually. The best protocol to try is IMAP, followed by POP3, followed by POP2.
- **User profile required to append to notes** - The **True** radio button should be selected when a valid user profile with matching email address is required to create new notes using the email client. Choose **False** if a user profile is not needed to append to notes using the email client.
- **User profile required to create new notes** - The **True** radio button should be selected when a valid user profile with matching email address is required to create new notes using the email client. Choose **False** if a user profile is not needed to append to notes using the email client.

- **Remove email signature** - The **Enabled** radio button strips off any standard email signature, such as a company slogan from any new note that is created. It also strips off any standard email signature from appends to existing notes.

Any errors generated during the setup or operation of the email-reply system are written to:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log
```

Note: The settings on this panel set up the Javamail utility, which is used by the SyncServer to retrieve messages from and send messages to the email account.

Disabling Email Reply and Note Creation

To disable email reply and note creation capability, enter `none` in the **Email Interface mail server** field which disables the Javamail utility. To re-enable, configure the **Email Interface mail server** field again.

Note: Javamail also stops when you stop the SyncServer and starts when you restart the SyncServer.

Specifying General Formatting Settings

The General Email Formatting Attributes panel lets you specify formatting settings that apply to all email notifications:

- **Default To: recipient:** - The default recipient of email when no known user is listed for the `To:` field of an email notification. This situation can occur when RevisionControl notes are generated. (See Working with RevisionControl Notes for details.)
Default is `nobody`. On UNIX systems, this setting can send the email notifications to a throw-away email account. However, you can enter a user name in this field to designate a default recipient.
- **State Based Subject** - Information about a change in the state of a note appearing as part of the subject of an email notification. Default is `State "moved to"`.
For example, for a SyncDefect note, the default states are `new`, `open`, `postponed`, `verification`, or `closed`. With the default value of `State "moved to"` in this field, the subject line of the email notification reads "SyncDefect moved to closed" when a defect is verified as fixed.
You optionally can enter a different string between the double quotes (" ") to prepend text to the state information. Using the preceding example, if you specify "has changed to state" in this field, the subject line of a closed SyncDefect email notification reads "SyncDefect has changed to state closed" when the state changes from verification to closed.

- **Max Short Property Length** - The maximum number of characters defined for a short property value. Default is 70 characters.
When users enter information in the corresponding field on the GUI, this number of characters determines whether the value is treated as long or short. Short property values are grouped and displayed at the top of an email notification, above long property values.
- **Max Property Change Length Display** - The maximum number of characters allowed for displaying changes to a property value. Default is 1,000 characters.
When users enter information in the corresponding field on the GUI, this setting determines the total number of characters that can be included in the email notification for any one property change for both old and new values. If the number of characters exceeds the setting, a condensed version of the value is displayed in the email notification. Users who want to read the entire text must read the note on the server. (This setting does not apply to transcript fields. Depending upon the value of the "Full Include Text" setting, the transcript fields display either the appended text or the full transcript text.)
- **HTML Notification Enabled** - Whether users with email readers that can display HTML will see email notifications in HTML format. You can select:
 - **True** - The specified server will send out both HTML and text email notifications. Users with email readers that can display HTML will see email notifications in HTML format. Users without this capability will continue to see text email notifications.
 - **False** - The specified server will send only text email notifications to all users.
- **ATTN Notification Enabled** - Indicates whether the **ATTN** field searching is active. If enabled, the processing of notes checks each line of text appended to any transcript field. If the line begins with `ATTN: <username or attn: username`, an additional email is sent that user identifying the message as an important notification that should be read as soon as possible. Multiple **ATTN** recipients can may be specified by entering `ATTN: <username username ...>` on a single line in a transcript field in a note. For note types with multiple transcript fields, all **ATTN** recipients is merged into a single notification. You can select:
 - **True** - To enable the **ATTN** field searching.
 - **False** - To disable the **ATTN** field searching. By selecting this option, DesignSync does not recognize the `ATTN:` keyword in a transcript field.

See Sending Email for a User's Attention in *ProjectSync User's Guide: Adding SyncNotes* for details on using `ATTN:` functionality.

- **Suppress Email to these Users** - The names of users who should not receive email notifications. For example, you may not want to send email notifications to users who are no longer in a group. Click **Modify** to open the CC List Helper, which lets you choose users for this field.
- **Audit Trail Entry Format** - The format used to display Audit Trail information when a note type property is configured to do so (see *ProjectSync User's Guide: Configurable Property Types*). By default, the system automatically displays the name of the field that has changed (`$ChangeField`), the fact that it was modified, the person who made the modification (`$SYNC_User`), the modification date and time (`$syncDate`), and the new value (`$NewValue`).
- **Rotate Email Logfiles** - The setting which lets you enable or disable logfile rotation. Default is disabled. When disabled, email trigger messages are logged to file `syncEmailTrigger.log`. When enabled, logging is done to a rotated log file, which is controlled by the settings of the Email Log File Rotation Interval variable.

Note: When determining which rotated log file to write to, DesignSync looks at the last rotated log file created when the trigger first starts. If the current time is within the time frame for the lifespan of the rotated log file, log information is appended to it. Otherwise a new logfile is created, using the current date as the datestamp of the logfile.

- **Email Log File Rotation Interval** - The setting which lets you control the frequency with which log files are rotated. Format of this variable is a string of the form "`# <period>`" where `#` is a positive integer and `<period>` is either days, weeks or months. The default value is "`1 weeks`". When logfile rotation is enabled, email trigger log output is directed to the file `syncEmailTrigger_YYYY-MM-DD.log`.

Formatting Text Email Notifications

The Text Email Formatting Attributes panel lets you specify formatting settings that apply to all text email notifications:

- **Property Set Format** - The format of property values in your email notification when a new note is created. The default is `%-15s=>'%s'`, where the symbol `=>` separates a property name and its value.

In this field, you can enter a valid string that is passed to the Tcl `format` command.

- **Property Append Format** - The format of appended property values in your email notification when note change notifications are sent. The default is `%-15s appended =>%s`, where the word 'appended' appears before the newly appended value.

In this field, you can enter a valid string that is passed to the Tcl `format` command.

- **Property Change Format** - The format of property values in your email notification when a note changes state. The default is `%-15s changed from '%s' => '%s'` where the words 'changed from' are inserted between the current and the previous state.

In this field, you can enter a valid string that is passed to the Tcl `format` command.

- **Property Full Text Format** - The heading used when the full text of discussions is included in email notifications. The default is `%-15s modified (full text follows) =>%s`.

To enable the full text in notifications capability, you must modify the default email settings through the Note Type Specific Email Formatting Attributes panel (see Include Full Text in Notifications in the **Specifying Settings for Individual Note Types** section).

- **Abbreviated Property Change Format** - The message displayed in an email notification for property values that exceed the maximum length (as defined in the Max Property Change Length Display). The default is `%-15s large change - follow link to see`.

When a user enters text that exceeds the maximum length allowed for a property value, the message in this field replaces the user's text. This message normally is used to tell users to read the entire text of the note on the server. (This setting is ignored when `Body` or another transcript field in a note type is configured to include the full thread in the email notification. See Specifying Settings for Individual Note Types for details.)

- **Unchanged Property Format** - The format string used to display unchanged property values when note change notifications are sent. The default is `%-15s = '%s'`.

This new setting is used only for note change modifications, and only for fields that are always included when they have not changed.

Formatting HTML Email Notifications

The HTML Email Formatting Attributes panel lets you specify formatting settings that apply to all HTML email notifications (These settings are applied only when HTML Notification Enabled is set to true):

- **Property Full Text Format** - This is the HTML message displayed in an email notification when the full text of a transcript field is included. It is associated with the Email Administrator setting

`SyncEmailHTMLFormatPropFullText`.

To enable the full text in notifications capability, you must modify the default email settings through the Note Type Specific Email Formatting Attributes panel (see Include Full Text in Notifications in the **Specifying Settings for Individual Note Types** section).

- **Abbreviated Property Change Format** - This is the HTML message displayed in an email notification when there is a property change which exceeds the long property change limit defined in the Max Property Change Length Display field in the **General Email Formatting Attributes panel**. It is associated with the Email Administrator setting

`SyncEmailHTMLLongPropChange`.

When a user enters text that exceeds the maximum length allowed for a property value, the message in this field replaces the user's text. This message normally is used to tell users to read the entire text of the note on the server. (This setting is ignored when `Body` or another transcript field in a note type is configured to include the full thread in the email notification. See Specifying Settings for Individual Note Types for details.)

Specifying Settings for Individual Note Types

The Note Type Specific Email Formatting Attributes panels let you specify settings that apply to email notifications for specific note types. The following fields can be edited on the panel for each note type:

- **Alias** - An alternate name for the note type that is displayed in email notifications. For example, for the SyncDefect note type, you might want to enter Defect Report.
- **Fields Included in Notifications for New Notes** - The properties that you want to include in email notifications of new notes. You can select one or more fields from the pull-down menu to add to the default properties listed. You also can delete properties from the text box.
- **Fields Excluded from Notifications** - The properties that you want to exclude from email notifications. You can select one or more fields from the pull-down menu to add to the default properties listed. You also can delete properties from the text box.

Note: You can get information on the properties included in each note type by choosing the **Install** option of the Note Type Manager. On the installation panel for the note type, click **Describe** to get a list of properties for the note

type. See *ProjectSync User's Guide: Installing Standard Note Types* for more information.

- **Fields Used to Build Distribution List** - A list of properties containing user names that determines which users automatically receive email notifications. For example, if you select **CcList**, all users in the **CcList** field of the note receive email notifications. You can select one or more properties from the pull-down menu to add to the default properties listed. You also can delete properties from the text box.
- **Fields on which to Disable Notification** - Properties for which email notifications are not sent. When one of the specified fields is the only information updated in a note, no email notification is generated. You can select one or more fields from the pull-down menu or delete properties from the text box .
- **Audit Trail** - The property used to store a history of changes to a set of properties and to display changes on the GUI. Select fields from the pull-down menu to automatically transfer the associated property names to the text box. The first entry is the property used to store the audit trail information. The remaining entries are the list of properties for which audit information is kept.

Note: You can modify the existing note type to add a specific property called `AuditTrail`. For more information on the `AuditTrail` property type, see *ProjectSync User's Guide: Configurable Property Types*. The **Audit Trail Entry Format** item on the General Email Formatting Attributes panel defines how the information that is tracked is displayed in the note's **Audit Trail** field.

- **Include Full Text in Notifications** - The properties for which email notifications contain the full text of discussions. By default, when a field on a note is modified, the email notification contains only the new appended text. You can change this behavior so that the email notifications contain all the appends to the specified field. The full text capability applies only to `Body` and other transcript fields. (See *ProjectSync User's Guide: Adding Transcript Fields to Notes* for details on creating transcript fields.)
- To enable full-text notifications, select a transcript field from the pull-down menu. The associated property name is automatically added to the text box.
- **Include View and Edit Links** - Whether or not email notifications include a hyperlink that lets the user quickly view or edit the note information. This option is available only on the panel for the RevisionControl note type; email notifications for other note types automatically include these hyperlinks. Select **True** from the pull-down menu to enable View and Edit hyperlinks on the RevisionControl note type. The default setting is **False**.
- **Fields Always Included in Notifications** -The properties that you would always want to include in any email notifications. You can select one or more fields from the pull-down menu to add to the default properties listed.

Reviewing Your Changes

The Review Changes panel displays a table that summarizes your changes:

- The **Variable** column shows the internal variable name used to hold the specified values.
- The **New Value** column shows the changes you made to the internal variable.
- The **Old Value** column shows the previous value of the internal variable.

Click **Commit Changes** to save your changes. The Changes Saved panel appears to confirm that your changes have been made.

Related Topics

Activating the Email Trigger

Editing the Email Trigger

You can use the Edit Trigger: Define Attributes panel to disable your email trigger or enable debug settings.

To access the Edit Trigger: Define Attributes panel, in the **Triggers** section of the **Admin** menu of the DesignSync Web UI, click **Edit**.

- If the email trigger is the only trigger defined on your server, the Edit Trigger: Define Attributes panel appears.
- If you have defined other triggers, the Edit Trigger: Choose Type panel appears. Click the **Email Notifications** hyperlink to go to the Edit Trigger: Define Attributes panel.

The **Name** field displays the name of the email trigger.

The **TCL File** field shows the name of the Tcl script that defines the trigger. You cannot edit these fields.

Deselect the **Active** field to suspend the email trigger. Email notifications are not sent when this box is not checked.

The **Type** field displays note because the email trigger fires only in response to note activity.

You can use the selections in the **Debug Settings** of the panel to make your email triggers generate detailed debugging output in the `syncEmailTrigger.log` file. (See Troubleshooting Email Triggers for information on the log file.)

You can generate detailed log reports for all trigger phases or for selected phases during the email-generation process:

- Enable the **Full Debug** checkbox to turn on debugging for all trigger phases.
- Enable one or more of the individual phases in the **Trigger Phase** field to turn on debugging only for the specified phase(s):
 - **Init** - Gets all the properties of the note.
 - **GetEventInfo** - Checks basic information about the note and decides whether or not the trigger should continue.
 - **ProcessEvent** - Does some specialized processing not related to generating the email (for example, updating AuditTrail fields).
 - **GatherRecipients** - Builds the To: and CC: lists from the list of people who should receive email notifications..
 - **BuildEmailContent** - Constructs the body of the email from the list of fields to be included in the email notification.
 - **SendEmail** - Dispatches the email.

Click **Submit** to make your changes.

The Operation Successful panel appears with a table that summarizes your changes. In the **Name** row, you can click **Email Notifications** to return to the Edit Trigger: Define Attributes panel.

Related Topics

[Activating the Email Trigger](#)

[Configuring Email Notifications](#)

[Customizing Email Triggers](#)

[Troubleshooting Email Triggers](#)

Activating the Email Trigger

When you first install DesignSync, you must activate the trigger that sends out email notifications. Email notifications are sent to subscribers whenever a note changes or an object is attached to a note.

You specify the behavior and format of email notifications using the Email Administrator. (See [Configuring Email](#) for details.) The email trigger is automatically activated when you use the Email Administrator to set up email notifications.

If you are upgrading from an earlier version of the DesignSync or ProjectSync, your previous email triggers are updated and you do not need to reactivate the email trigger.

Although the email trigger is enabled automatically when you use the Email Administrator, you also can manually activate the trigger.

To manually activate the email trigger:

1. In the **Triggers** section of the **Admin** menu in the DesignSync web UI, click **Add**.

The **Add Trigger: Choose Type** panel appears if you have not yet set up your email trigger. (If you see the **Add Trigger: Define Attributes** panel, your email trigger is already set up.)

2. Click **Email Notifications**.

The **Operation Successful** panel appears and confirms that the email trigger (`syncEmailTrigger.tcl`) is enabled. Click **Email Notifications** in the table on the panel to enable debugging or deactivate the email trigger.

To deactivate the email trigger or enable debug settings, see [Editing the Email Trigger](#).

For more detailed information on triggers, including email triggers, see [Creating Note Object Triggers](#).

See [Troubleshooting Email Triggers](#) for information on solving trigger-related problems.

Related Topics

[Configuring Email Notifications](#)

[Creating Note Object Triggers](#)

[Customizing Email Triggers](#)

[Editing the Email Trigger](#)

[Troubleshooting Email Triggers](#)

Customizing Email Triggers

DesignSync lets you customize email triggers for a particular note type. For example, if RevisionControl note generation for the "tag" command is enabled, you can customize the trigger so that email is sent only when a tag is applied and not when a tag is deleted.

There are six phases to the email trigger, all of which can be customized:

DesignSync Data Manager Administrator's Guide

1. **Init** - Gets all the properties of the note.
2. **GetEventInfo** - Checks basic information about the note and decides whether or not the trigger should continue.
3. **ProcessEvent** - Does some specialized processing not related to generating the email (for example, updating AuditTrail fields).
4. **GatherRecipients** - Builds the To: and CC: lists.
5. **BuildEmailContent** - Constructs the body of the email.
6. **SendEmail** - Dispatches the email.

To overload any or all of the trigger phases, create a file named `syncTrigger_<note type>.tcl` in the appropriate custom area:

- `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/tcl` (**server-specific**).
- `<SYNC_SITE_CUSTOM>/share/tcl` (**site-wide**).

The trigger will look for this file and load it if found. A `<SYNC_CUSTOM_DIR>/servers` file takes precedence over one in `<SYNC_SITE_CUSTOM>`; a `<SYNC_SITE_CUSTOM>` file takes precedence over one in `<SYNC_ENT_CUSTOM>`. (The custom enterprise area is reserved for future development.)

To overload a specific phase of the trigger in this file, create a proc named `syncTrig<name of phase>_<note type>`. For example, to overload the `ProcessEvent` phase for `RevisionControl` notes, you would create a proc named `syncTrigProcessEvent_RevisionControl`.

Note: The server only checks for the existence of this file at startup and system reset. If you add this file while the server is running, reset the server in order to use the custom trigger.

All procs need to return an integer status code. Any return value greater than 1 will cause the trigger to exit.

If you want to extend the functionality of the existing phase in your overloaded proc, the recommended approach is to first call the built-in default handler for that phase (`syncTrig<phase name>`) and then add your own code after it. Continuing the example from above, calling

```
set status syncTrigProcessEvent
```

would run the built-in `ProcessEvent` function, then continue with the custom proc. The `status` variable is the return status the built-in proc would have returned. Save this value and check it to determine whether the trigger should exit.

The packaged `<SYNC_DIR>/share/tcl/syncEmailTrigger.tcl` file is a useful example. You can refer to this file to find global variables utilized by the default proc for a particular phase.

Example

In this example, you want to send email only when a tag is applied, not when it is deleted.

To do this, create a `syncTrigger_RevisionControl.tcl` file in `<SYNC_SITE_CUSTOM>/share/tcl`, with a custom `syncTrigGetEventInfo_RevisionControl` proc, as follows. The proc first runs the built-in `syncTrigGetEventInfo` proc, and checks the result. Then, it determines whether the revision control operation was a "tag -delete". If so, the trigger exits. Otherwise, the trigger continues with the standard trigger processing.

```
#
$SYNC_DIR/custom/site/share/tcl/syncTrigger_RevisionControl.tcl
#
proc syncTrigGetEventInfo_RevisionControl {} {
    variable noteURL
    variable syncNoteProps
    # Run the built-in GetEventInfo function
    set status [syncTrigGetEventInfo]
    if {$status > 1} {
        ::email::log::debug "Exiting based on syncTrigGetEventInfo"
        return $status
    }
    set syncNoteProps(Command) [url getprop $noteURL Command]
    # If this is a tag operation, and a tag is being deleted,
    # then abort the trigger
    if {$syncNoteProps(Command) == "tag"} {
        # Use the shortest accepted representation of "-delete",
        # to match all invocations
        if {[string match "*-d*" $syncNoteProps(Title)]} {
            return 2
        }
    }
    # Otherwise, continue
    return 0
}
```

Related Topics

Activating the Email Trigger

Configuring Email Notifications

Creating ProjectSync Triggers

Editing the Email Trigger

Troubleshooting Email Triggers

Troubleshooting Email Triggers

Files containing information about email subscriptions and email triggers are located in:

`<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/data/EmailMgr`

This directory contains two files:

- `syncEmailDb.dat` - Lists all email subscriptions.
- `syncEmailTrigger.log` - Lists all the events and errors that occur when email triggers generate email notifications. If you do not see this file, your email triggers are not enabled.

The `syncEmailDb.dat` File

The `syncEmailDb.dat` file is an ASCII file that lists each email subscription for each subscribed user. The list is in alphabetical order by user name:

```
heidi, SyncDefect, ALL, State=postponed
heidi, SyncDefect, ALL, State=verification
masteradmin, SyncDefect, ALL, State=closed
masteradmin, SyncDefect, ALL, State=new
masteradmin, SyncDefect, ALL, State=verification
mmf, SyncDefect, ALL, State->closed
mmf, SyncDefect, ALL, State->verification
zaphod_b, ALL, ALL,
```

You can use this list to get a quick overview of the subscriptions registered on your server.

The `syncEmailTrigger.log` File

The `syncEmailTrigger.log` file records all the events that occur during email notifications. These events include each phase of the processing of the email notifications and any error messages generated by the trigger.

To enable recording of events in the `syncEmailTrigger.log` file, you must first enable debugging using the Edit Trigger: Define Attributes panel. (See Editing the Email

Trigger for details.) If your email triggers do not appear to be functioning correctly, you can enable debugging and then check the `syncEmailTrigger.log` file for error messages.

For each email generated, the email trigger performs a series of actions. These actions are listed in the `syncEmailTrigger.log` file as a series of phases. The `syncEmailTrigger.log` file shows the complete sequence for each email notification sent out by the trigger. The six phases are:

1. **Init** - Gets all the properties of the note.
2. **GetEventInfo** - Checks basic information about the note and decides whether or not the trigger should continue.
3. **ProcessEvent** - Does some specialized processing not related to generating the email (for example, updating AuditTrail fields).
4. **GatherRecipients** - Builds the To: and CC: lists from the list of people who should receive email notifications..
5. **BuildEmailContent** - Constructs the body of the email from the list of fields to be included in the email notification.
6. **SendEmail** - Dispatches the email.

See Editing the Email Trigger for more information on using these phases.

Access Control Restrictions

Access controls also govern email notifications. For example, if a user does not have permission to view a note type (the ViewNote access control), the user does not receive email notifications when notes of that type are updated. Similarly, if a user does not have permission to view a project (the BrowseServer access control), the user does not receive email notifications when a note attached to the project is updated.

If your email notifications are not being sent as expected, they may be subject to custom access control restrictions set by your system administrator. See the ENOVIA Synchronicity Access Control Guide for more information.

Related Topics

- Activating the Email Trigger
- Configuring Email Notifications
- Creating Note Type Triggers
- Customizing Email Triggers
- Editing the Email Trigger

Troubleshooting Email Notification of RevisionControl Notes

Setting RevisionControl Note Generation

A RevisionControl note is a note created automatically when a DesignSync revision-control operation, such as check in or check out, takes place (see *Working with RevisionControl Notes* for details). The **RevisionControl Notes** tab page on the Administer Server Settings panel lets you enable RevisionControl note generation for specific revision-control commands. By default, RevisionControl note generation is disabled.

The **RevisionControl Notes** tab page enables RevisionControl note generation for a specific server. Use the **RevisionControl Notes** tab page in DesignSync's SyncAdmin tool to enable RevisionControl note generation for an entire site. For more information, see *Enabling the Generation of RevisionControl Notes*.

Note: Whether or not you plan to use the email feature, you must configure your server for email notifications (see *Configuring Email Notifications*) in order for RevisionControl notes to function properly.

To use the **RevisionControl Notes** tab page:

1. Select **Admin Menu | Server | Administer Server** from the DesignSync Web UI menu.

This menu option appears only when you have permission to perform administrative functions. See the *ENOVIA Synchronicity Access Control Guide* for details.

2. Click the **Server Settings** hyperlink on the Administer Server: Choose Operation panel.
3. If prompted for a login, enter your user name and password.
4. Click the **RevisionControl Notes** tab to get the tab page. This page lists the commands for which you can enable RevisionControl notes.
5. Click on the checkbox next to a command to enable RevisionControl notes for that action. To disable notes for a revision-control action, click on the checkmark. You can enable or disable RevisionControl note generation for the following DesignSync operations:
 - **Check In** - When an object is checked into the vault. (See *DesignSync Help: Checking in Design Files* for more information on the check-in operation.)

Note: For Module operations like adding or removing hrefs, an RC note for 'ci' is generated. For a complete list of Module operations that generate RevisionControl notes, see *Setting Up Email Notification of RevisionControl Notes*.

- **Check Out or Populate with lock** - When a module or DesignSync object is checked out with a lock into a local work area. (See *DesignSync Data Manager User's Guide: Checking out Design Files and Populating Your Work Area* for details on the check-out and populate operations.)

Note: Checkout and Populate with lock operations generate RC notes. For a complete list of Module operations that generate RevisionControl notes, see Setting Up Email Notification of RevisionControl Notes.

- **Check Out or Populate without lock** - When a module or DesignSync object is checked out without a lock into a local work area. (See *DesignSync Data Manager User's Guide: Checking out Design Files and Populating Your Work Area* for details on the check-out and populate operations.)
- **Make Branch** - When a new branch is created for all objects in a project or module. (See *DesignSync Data Manager User's Guide: Creating Branches* for details on the make branch operation.)
- **Move (File or Folder)** - When a specified local file or folder is moved or renamed. (See *DesignSync Data Manager User's Guide: Moving and Renaming Files and Moving and Renaming Folders* for details on the move file and move folder operations.)
- **Remove (Vault, Version or Folder)** - When the specified vault, version or file is deleted. (See *DesignSync Data Manager User's Guide: Deleting Design Objects* for details on the remove vault, remove version and remove folder operations.)
- **Retire** - When an obsolete object's branch is retired. (See *DesignSync Data Manager User's Guide: Retiring Design Objects* for details on the retire operation.)
- **Switch "Locker"** - When the lock owner of a branch is changed. (See *DesignSync Help: Using the Locking Workstyle* for more information on operations that involve locks.)
- **Tag** - When module branches, versions or versions of design objects are tagged as part of a configuration. (See *DesignSync Data Manager User's Guide: What is a Design Configuration and Tagging Versions and Branches* for detail on tags.)
- **Unlock/Cancel** - When locked design objects or modules are unlocked. (See *DesignSync Data Manager User's Guide: Unlocking Branches* for details on unlocking a design object or cancelling a lock.)
- **Make Module** - When a new Module is created. (See *DesignSync Data Manager User's Guide: Creating a Module* for details on creating a Module.)
- **Remove Module** - When a Module is removed from the server. (See *DesignSync Data Manager User's Guide: Deleting a Module* for details on deleting a Module.)

- **Lock Module** - When locking a module. See *DesignSync Data Manager User's Guide: Module Locking* for details on locking a Module branch.)
 - **Upgrade Module** - When upgrading legacy modules. See *DesignSync Data Manager User's Guide: How DesignSync Handles Legacy Modules* for details on upgrading a legacy Module.
6. Click **Apply** when you are satisfied with the new RevisionControl note settings. When you click **Apply**, the changes you have made on any of the tab pages are written to the `PortRegistry.reg` file. This file contains settings for your local SyncServer.

The **Apply** button is disabled until changes are made to one of the tab pages.

Related Topics

Overview of SyncServer Administration

Setting Up Email Notification of HCM RevisionControl Notes

ProjectSync User's Guide: Using ProjectSync with DesignSync

Using the Administer Server Panel

Working with RevisionControl Notes

Setting Up Email Notification of Module RevisionControl Notes

A RevisionControl note is created automatically when a module revision-control operation, such as check in or check out, takes place (see *Working with RevisionControl Notes* for details). The DesignSync Web UI features a method for enabling RevisionControl Note generation for specific revision-control commands. By default, RevisionControl note generation is disabled. For more information on enabling specific RevisionControl note generation, see *Setting RevisionControl Note Generation*.

Note: You cannot enable RevisionControl notes for legacy modules.

The following table lists the RevisionControl notes created for different Module operations.

Command	RC notes generated
mkmod	hcm mkmod
rmmod	hcm rmmod

addhref	
rmhref	ci
mvmember	
remove	
rollback	
mkbranch	mkbranch
tag	tag
populate (lock / nolock)	populate (lock / nolock)
unlock/cancel	unlock/cancel
lock	hcm lock
upgrade	hcm upgrade

The Email Administrator performs these steps to set up email notification of Module RevisionControl notes:

1. Configure email notifications. See Configuring Email for details.
2. From the **Email Administrator Index of Operations** page, select the **Revision Control** link under **Note Type Specific Email Formatting Attributes**. This displays the **Revision Control Specific Settings** panel.
3. For the **Fields Included in Notifications** field, select **Full Command** from the pull-down menu to add the notes type "FullCmd".
4. Click **Review** to review your changes. Then click **Commit Changes** to have the settings take effect.

Users can then subscribe to email for RevisionControl notes, as they would for other note types. The email contains the RevisionControl note's contents and a list of objects (the URL of the relevant modules affected by the operation).

Related Topics

[Setting RevisionControl Note Generation](#)

[Working with RevisionControl Notes](#)

[The CPU Team Subscribes to Email on a Hierarchy](#)

Web Interface Configuration

Controlling HTTP Headers

In certain situations, you may want to return output other than text to users' HTTP requests. For example, if a user requests certain data, you may want the information to be presented in a spreadsheet program.

By default, DesignSync attaches HTTP headers that tell the web browser it is receiving HTML text to the output of your server-side Tcl script. To run a server-side Tcl script that generates anything other than HTML, you must instruct DesignSync to suspend the default headers and output the headers you want. To do so:

1. Add "Sync-user-headers" as the first line of any output being returned to the user. Set the value of Sync-user-headers to "enabled."

This line tells the server to parse the output for all remaining HTTP header lines and to add the results to the response generated.

2. Delimit this and subsequent HTTP header lines with a new line.
3. On the next line, specify the type of HTTP header and value. Use the format

```
<Header-name>: <value>.
```

Note that you cannot set a specific value for the Content-length header because the server sets this based on the actual output.

4. Enter a single blank line to tell the server to terminate the parsing of HTTP headers before continuing with straight text.

Example

The following Tcl Script would return information in a spreadsheet format. Note that tabs are needed between the cell entries for the script to run correctly. The tab character acts as a cell delimiter.

```
puts "Sync-user-headers: enabled"
puts "Content-type: application/xlc"
puts ""
puts "cell 11\t cell 12"
puts "cell 21\t cell 22\t cell 23"
puts "cell 31\t cell 32"
puts "cell 41\t cell 42"
puts "cell 51\t cell 52"
puts "cell 61"
```

For more information on Tcl scripts, refer to the topic *ProjectSync User's Guide: ProjectSync Scripts*.

Note: The very first output of your script must be the "Sync-user-headers: enabled" line or the script will have no effect.

Accessing Apache Document Root

The Apache document root (`htdocs` directory) is closed to everyone by default. This restriction ensures the security of your web site or other public documents that are stored on the same server as the DesignSync web server.

If desired, you can open access to the document root by editing the `httpd.conf` file, which is located in the directory:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/conf
```

The configuration file contains the following settings:

```
#
# Controls who can get stuff from this server.
#
# We protect htdocs from unauthorized users by default.
# Uncommenting 3 lines commented with ## will open
# htdocs directory for GET requests from everybody.
#
## allow from all
## <Limit POST PUT DELETE>
    # Basic Authentication by Synchronicity
    AuthName Synchronicity
    AuthType Basic
    Require valid-user
## </Limit>
```

To open up the `htdocs` directory, remove the `##` comments from the lines:

```
allow from all
<Limit POST PUT DELETE>
    ...
</Limit>
```

After editing the file, you must restart the server for the changes to take effect. Users can then access files in the `htdocs` directory using the URL:

```
http://<host>:<port>/filename.html
```

For additional information on Apache, see their web site at <http://www.apache.org>.

SyncServer Aliases

You cannot use absolute file system paths when specifying locations on a SyncServer. Instead, DesignSync provides a set of aliases that let you navigate the DesignSync installation directory.

These aliases are:

Server Alias	File System Path
/	<SYNC_CUSTOM_DIR>/servers/<host>/<port>/htdocs
/syncinc	<SYNC_DIR>/share/content
/syncent	<SYNC_ENT_CUSTOM>/share/content
/syncsite	<SYNC_SITE_CUSTOM>/share/content
/syncserver	<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/content

Note: The / alias is valid for only Apache servers.

For information on resolving variables in the file system paths, see *ProjectSync User's Guide: ProjectSync Environment Variables*.

For example, to specify the location of a .gif file that your system administrator has added to the images folder, you would specify:

```
"<img src=/syncsite/images/bar.gif>"
```

where /syncsite is an alias for the <SYNC_SITE_CUSTOM>/share/content directory.

Setting up Note Object Triggers

Creating Note Object Triggers

Note Object triggers are agents that perform actions, like running a Tcl script, under circumstances you specify; for example in response to specified note events. Here are some examples of how you might use Note Object triggers in your environment:

- You can set up a trigger so that when a team member attaches a note to a particular project or a DesignSync object, the trigger runs a Tcl script you have developed.
- You can set up a trigger so that when a team member updates an existing note, a script you've developed runs.
- You can configure DesignSync to send email when a design changes. For example, a Note Object trigger can automatically send email to team members when new files are checked in to the DesignSync vault or when a new release is created.

To create a custom trigger you must:

1. Place a Tcl script that defines your trigger in one of the custom areas.
2. Set up your trigger using the Add Trigger panel of the DesignSync web interface.

These steps are discussed in the following sections.

Note: If you are upgrading from an earlier version of DesignSync or ProjectSync, your active triggers are converted to the trigger format used in the new installation. Inactive triggers are not converted. Trigger conversions are recorded in the `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log` file.

Creating a Trigger Script

You write your trigger scripts in stcl which is the combination of the Tcl scripting language and the DesignSync command set. Object Note Trigger scripts should be created as server-side scripts.

See the ENOVIA Synchronicity stcl Programmer's Guide for more detailed information on creating your own server-side scripts. You can also refer to the ProjectSync User's Guide for help writing Tcl scripts for the DesignSync environment. A sample trigger, `SyncDefect_NoteChangeTrigger.tcl`, is in `$SYNC_DIR/share/examples/NoteTypes/SyncDefect/tcl`.

When you have created your trigger script, put it in one of the following areas:

- o `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/tcl` (server-specific).
- o `<SYNC_SITE_CUSTOM>/share/tcl` (site-wide).

Use one of these custom areas rather than the DesignSync Tcl script directory (`<SYNC_DIR>/share/tcl`) to prevent your scripts from being overwritten when you reinstall or upgrade DesignSync software.

The trigger application first searches for Tcl scripts in the server-specific `tcl` directory, then in the site-wide `tcl` directory, then in the enterprise-wide `tcl` directory, and finally in the DesignSync `tcl` directory. (The enterprise-wide custom area is reserved for future development.)

If the script is not found in the search path, an error is written to the `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log` file when the trigger fires. See [SyncServer Error Log File Messages](#) for more information.

Setting Up a Trigger

When you have placed your trigger script in one of the custom areas, you set up the trigger using the Add Trigger panel of the DesignSync Web UI.

To set up a trigger, click **Add** in the **Triggers** section of the **Admin Menu**. The Add Trigger: Define Attributes panel appears.

If you see the Add Trigger: Choose Type panel, you have not yet set up your email trigger. Click the **Note Activity** hyperlink to go to the Add Trigger: Define Attributes panel. See Activating the Email Trigger in Configuring Email Triggers for information on setting up the email trigger.

In the Add Trigger: Define Attributes panel, you can set trigger attributes using the following fields.

Name - Enter a brief name for your trigger. Trigger names must be unique. You can include spaces in trigger names, but do not use quotation marks, equal signs, or backslashes.

TCL File - Enter the path and file name for your trigger script, relative to one of the custom `share/tcl` directories.

Type - This field always shows **note**. Your triggers can fire only in response to events that affect notes.

Command - Select the note action(s) that fire the trigger. You can select one or more of the following commands:

- **create** - The trigger fires when a note is added.
- **delete** - The trigger fires when a note is deleted.
- **modify** - The trigger fires when a note is edited.
- **attach** - The trigger fires when an object is attached to a note. If you choose this command, you also must enable the checkbox in the following **Atomic** field.
- **detach** - The trigger fires when an object is detached from a note. If you choose this command, you also must enable the checkbox in the following **Atomic** field.

If no commands are selected, the trigger fires for all the commands.

When a note is deleted, any links between the note and the objects to which it is attached are removed. Detach events are not generated for the removal of links when a note is deleted.

Atomic - Enable this checkbox to fire the trigger for each command as it occurs (an atomic event); disable this checkbox to fire the trigger at the end of the operation (a non-atomic event).

To understand the difference between atomic and non-atomic events, it is helpful to know how events are generated. When a note operation occurs, each command

generates an individual event. For example, the following pseudo-code contains three events:

```
set noteUrl [note create -type SyncDefect {Title {Test}}]
note attach $noteUrl sync:///Projects/ProjectSync
note setprops $noteUrl State open Class sw-bug
```

This script contains a create command, an attach command, and a modify command (`setprops`) that sets values for note properties. Each individual event created by a command is an **atomic event**.

At the end of the operation, a summary event is generated for each note. This summary event contains information on all the atomic actions that occurred during the operation. This summary event is a **non-atomic event**.

You can use the **Atomic** Boolean to filter the behavior of your triggers:

- If **Atomic** is selected, your trigger fires in response to the atomic events you selected in the **Command** field.
- If **Atomic** is not selected, your trigger fires only in response to the summary event. The summary event contains only summary information for the commands you selected in the **Command** field.

For example, if you select **modify** in the **Command** field and do not enable **Atomic**, a non-atomic summary event is generated that contains only information on modifications made during the operation. Your trigger script will fire in response to the non-atomic summary operation rather than in response to each modification made to the note's properties.

In general, if your trigger script modifies the note that it fires in response to, your trigger should be registered as atomic so that the changes made by your script are included in the summary event at the end of the operation. Otherwise, you should register your script as non-atomic.

If you select **attach** and/or **detach**, your trigger must fire in response to one or both of these atomic events. There are no separate attach and detach summary events because these commands are included in the modify non-atomic event.

Note Type - From the pull-down menu, select the note type that the trigger operates on. Leave the default, **ALL**, to apply your trigger to all note types.

If your trigger fires for one note type, the trigger will perform more efficiently if you select the note type from the **Note Type** pull-down menu rather than specifying the note type in your script. If your trigger fires for more than one note type, use the default **ALL** and specify the note types in your trigger script.

Note Id - Enter the note ID number if you want your trigger to fire in response to a specific note.

Note URL - Enter the `sync:///` URL for a note. You can use wildcards to match multiple notes.

For example, to fire a Tcl script when notes of type SyncDefect change or are attached, enter `sync:///Note/SyncNotes/SyncDefect/*`. The wildcard (*) applies the trigger script to all objects (for example, projects and configurations) to which the SyncDefect is attached. The wildcard can be used only at the end of your string. In most cases, however, you would leave this field blank.

The `sync:///` syntax is shorthand for the host and port of the server that is running your script. Because the script is run on this server, the `host:port` segment of the URL is unnecessary and may prevent your trigger from firing.

Object URL - Enter the DesignSync URL for an object to fire your trigger when notes are attached to or detached from the object. This setting applies only to atomic attach and detach events. For example:

```
sync:///Projects/Project1/*
```

The wildcard (*) causes the trigger to fire when notes are attached or detached from any object in the project. The wildcard can be used only at the end of your string.

Object Type - From the pull-down menu, select the type of object. You can set the trigger to fire when notes are attached to or detached from any of the objects in the pull-down menu. This setting applies only to atomic attach and detach events.

To support note triggers for Module, the Object Type pulldown has the following values: Module and Module Version.

Full Debug - In the email trigger, this setting indicates that debugging is enabled for all phases of the email notification process. In this release, this setting applies only to the email trigger. See Editing the Email Trigger for details.

Trigger Phase - In this release, these settings apply only to the email trigger. See Editing the Email Trigger for details.

Click **Submit** to add your trigger. The Operation Successful panel appears with a table that summarizes the properties of your new trigger. The table includes two hyperlinks:

- In the **Name** row, you can click the name of the trigger to go to the Edit Trigger panel (see Editing Note Object Triggers), where you can amend the properties of the trigger.

- If you have more than one custom trigger, you can click **Define execution order** in the **Set Trigger Execution Order** row of the table. This link takes you to the Setting Note Object Trigger Execution Order panel where you can specify the order in which your triggers fire.

Related Topics

- Activating the Email Trigger
- Editing the Email Trigger
- Editing Note Object Triggers
- Setting Trigger Execution Order
- Troubleshooting Email Triggers

Editing Note Object Triggers

The Edit Trigger: Define Attributes panel lets you modify the behavior of a trigger. To access this panel, you must first select a trigger to edit.

Selecting a Trigger

To edit a trigger, select **Edit** from the **Triggers** section of the DesignSync web menu. The Edit Trigger: Choose Trigger panel appears. Depending on the number of triggers you have defined, you get one or more of the following choices:

- An **Email Notifications** hyperlink in the **Email Notifications** section of the panel. Click this link to activate or deactivate the email trigger and to set debugging behavior. See Editing the Email Trigger for information on modifying the behavior of this trigger.
- A list of your custom triggers in the **Note Activity** section of the panel. The triggers are sorted in priority order, with the trigger that executes first at the top. Click on a hyperlinked trigger name to go to the Edit Trigger: Define Attributes panel and edit the selected trigger.
- If you have more than one custom trigger, a **Set execution order** hyperlink appears in the **Note Activity** section of the panel. Click this link to go to the Edit Trigger: Set Trigger Execution Order panel, where you can define the priority of the triggers.

Editing a Trigger

When you have chosen a trigger to edit from the Edit Trigger: Choose Trigger panel, its attributes are listed in the Edit Trigger: Define Attributes panel. You change the trigger attributes for the following fields.

Name - The brief name for your trigger. Trigger names must be unique. You can include spaces in trigger names, but do not use quotation marks, equal signs, or backslashes.

TCL File - The path and file name for your trigger script, relative to one of the custom `share/tcl` directories.

If the script is not found in the search path, an error is written to the `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log` file when the trigger fires. See SyncServer Error Log File Messages for more information.

Active - Whether or not the trigger is enabled. Deselect this checkbox to suspend your trigger.

Type - This field always shows **note** as the object type. Your triggers can fire only in response to events that affect notes.

Command - The note action(s) that fire the trigger. You can select one or more of the following commands:

- **create** - The trigger fires when a note is added.
- **delete** - The trigger fires when a note is deleted.
- **modify** - The trigger fires when a note is edited.
- **attach** - The trigger fires when an object is attached to a note. If you choose this command, you also must enable the checkbox in the following **Atomic** field.
- **detach** - The trigger fires when a an object is detached from a note. If you choose this command, you also must enable the checkbox in the following **Atomic** field.

If no commands are selected, the trigger fires for all the commands.

When a note is deleted, any links between the note and the objects to which it is attached are removed. Detach events are not generated for the removal of links when a note is deleted.

Atomic - Whether the trigger fires at the action of individual commands or at the end of the operation. Enable this checkbox to fire the trigger for each command as it occurs; disable this checkbox to fire the trigger at the end of the operation.

To understand the difference between atomic and non-atomic events, it is helpful to know how events are generated. When a note operation occurs, each command generates an individual event. For example, the following pseudo-code contains three events:

```
set noteUrl [note create -type SyncDefect {Title {Test}}]
note attach $noteUrl sync:///Projects/ProjectSync
note setprops $noteUrl State open Class sw-bug
```

This script contains a create command, an attach command, and a modify command (`setprops`) that sets values for note properties. Each individual event created by a command is an **atomic event**.

At the end of the operation, a summary event is generated for each note. This summary event contains information on all the atomic actions that occurred during the operation. This summary event is a **non-atomic event**.

You can use the **Atomic** Boolean to filter the behavior of your triggers:

- If **Atomic** is selected, your trigger fires in response to the atomic events you selected in the **Command** field.
- If **Atomic** is not selected, your trigger fires only in response to the summary event. The summary event contains only summary information for the commands you selected in the **Command** field.

For example, if you select **modify** in the **Command** field and do not enable **Atomic**, a non-atomic summary event is generated that contains only information on modifications made during the operation. Your trigger script will fire in response to the non-atomic summary operation rather than in response to each modification made to the note's properties.

In general, if your trigger script modifies the note that it fires in response to, your trigger should be registered as atomic so that the changes made by your script are included in the summary event at the end of the operation. Otherwise, you should register your script as non-atomic.

If you select **attach** and/or **detach**, your trigger must fire in response to one or both of these atomic events. There are no separate attach and detach summary events because these commands are included in the modify non-atomic event.

Note Type - The note type that the trigger operates on. You can select a note type name from the pull-down menu or choose **ALL** to apply your trigger to all notes types.

If your trigger fires for one note type, the trigger will perform more efficiently if you select the note type from the **Note Type** pull-down menu rather than specifying the note type in your script. If your trigger fires for more than one note type, use the default **ALL** and specify the note types in your trigger script.

Note Id - The note ID number. Enter a note ID number if you want your trigger to fire in response to a specific note.

Note URL - The `sync:///` URL for a note. You can use wildcards to match multiple notes.

For example, to fire a Tcl script when notes of type SyncDefect change or are attached, enter `sync:///Note/SyncNotes/SyncDefect/*`. The wildcard (*) applies the trigger script to all objects (for example, projects and configurations) to which the SyncDefect is attached. The wildcard can be used only at the end of your string. In most cases, however, you would leave this field blank.

The `sync:///` syntax is shorthand for the host and port of the server that is running your script. Because the script is run on this server, the `host:port` segment of the URL is unnecessary and may prevent your trigger from firing.

Object URL - The DesignSync URL for an object. Your trigger fires when notes are attached to or detached from the object. This setting applies only to atomic attach and detach events. For example:

```
sync:///Projects/Project1/*
```

The wildcard (*) causes the trigger to fire when notes are attached or detached from any object in the project. The wildcard can be used only at the end of your string.

Object Type -The type of object - either **Project** or **Project Configuration**. You can set the trigger to fire when notes are attached to or detached from either a project or a configuration. This setting applies only to atomic attach and detach events.

Full Debug - In the email trigger, this setting indicates that debugging is enabled for all phases of the trigger. In this release, this setting applies only to the email trigger. See Editing the Email Trigger for details.

Trigger Phase - In this release, these settings apply only to the email trigger. See Editing the Email Trigger for details.

Click **Submit** to make the changes to your trigger.

Click **Delete** to remove your trigger. DesignSync no longer recognizes your trigger, but the trigger Tcl file is not removed from the custom `share/tcl` area.

When you click **Submit**, the Operation Successful panel appears with a table that summarizes the properties of the trigger. The table includes two hyperlinks:

- In the **Name** row, you can click the name of the trigger to return to the Edit Trigger panel (see Editing Note Object Triggers), where you can make further modifications.
- If you have more than one custom trigger, you can click **Define execution order** in the **Set Trigger Execution Order** row of the table. This link takes

you to the Edit Trigger: Set Trigger Execution Order panel where you can specify the order in which your triggers fire.

Related Topics

- Activating the Email Trigger
- Creating Note Object Triggers
- Editing the Email Trigger
- Setting Trigger Execution Order
- Troubleshooting Email Triggers

Setting Note Object Trigger Execution Order

When you have multiple trigger scripts, you can use the Edit Trigger: Set Trigger Execution Order panel to define the priority of the triggers.

To access the Edit Trigger: Set Trigger Execution Order panel:

1. From the DesignSync Web UI Select **Admin Menu | Triggers | Edit**.

The Edit Trigger: Choose Trigger panel appears.

2. In the **Note Activity** section of the panel, click **Set execution order**. This hyperlink appears only when you have multiple note-related triggers.

The Edit Trigger: Set Trigger Execution Order panel appears.

You also can access the Edit Trigger: Set Trigger Execution Order panel from the Operation Successful panel of the Add Trigger panel (see Creating Note Object Triggers) and the Edit Trigger: Define Attributes panel.

The panel contains two fields:

- **Object Type** - The type of object that the trigger applies to. Note is the only object type supported.
- **Trigger List** - A list of all your triggers in execution order. The trigger that fires first is at the top of the list. To change the priority order:
 1. Select the name of the trigger from the list.
 2. Click **Move Up** to move the trigger up the list; click **Move Down** to move the trigger down the list.

All of your triggers probably won't apply to every note operation. Instead, only a subset of your triggers are fired for individual note activities. This subset of triggers fires in the order that they are listed in the **Trigger List** field.

The email notification trigger is not included in the **Trigger List** field because it always fires last.

Related Topics

[Creating Note Object Triggers](#)

[Editing Note Object Triggers](#)

Managing Vault Types

About Vault Types

RCE

DesignSync typically stores your data in a RCE vault type. This vault type always has a `.rca` file extension. Within this vault type, the versions you create are stored as deltas, and each delta contains the changes between the current version and the previous version of the data you checked in. The vault consists of two parts, a header section and a data section. The header section stores the meta information (metadata) about each revision, for example the check-in time, author, comment, and any properties you use for branching and merging data. Any associated tags are stored in the tags database.

The other vault types are supported are as follows.

Copy Vault

Zipped Copy Vault

Smart Vault

Zipped Smart Vault

Copy Vault

When working with large binary files, the Copy Vault enhances performance by skipping the delta computation involved in RCE Vault and copying the entire file into the vault.

The Copy vault also keeps all the functionality that the DesignSync RCE Vault provides. It preserves the header information but, instead of storing the changes for each

version, the Copy Vault stores a complete copy of the version you check in. The Copy Vault type has a `.cpv` extension.

Important: Module member symbolic links should never be checked in to a CopyVault.

Zipped Copy Vault

Copy Vault also provides an optional Zipped Copy Vault feature that allows you to compress files with specified extensions to save disk space.

The DesignSync client send files containing compressed data to the server where they are stored and sent back to the client when a fetch is requested. The client then uncompresses the data content. This enhances performance by saving transfer time.(transferring much less data between the client and server than the actual size of the file).

The data stored in a Zipped Copy Vault has the same limitations as binary files. Keywords cannot be expanded, diff, merge or end-of-line conversion on text files is not applicable. Also, `ci`, `co` and `populate` will always use the `-retain` option (although `-get`, `-keep` and `-lock` files ordinarily has `-noretain` by default).

Smart Vault

Smart Vault provides a vault storage option in which the type of storage is determined by the properties of the object rather than the object name --- even if the properties differ from version to version. Smart Vault has the ability to examine the properties of the object and choose the optimal storage method. Smart Vault will also take into consideration the trade-off between disk space usage and the fetch speed. You can customize Smart Vault to control these parameters.

Smart Vault allows you to choose different parameters for different path name patterns. For example, you can specify compression for `.gds` files, but not for `.gz` files.

Choosing Smart Vault does not slow down your check-in speed. Smart Vault uses a post-check-in processing phase to examine the object and make the best storage type decision. Post-processing parameters can be controlled from the SyncAdmin Vault Types pane.

Zipped Smart Vault

Smart Vault also provides an optional Zipped feature that allows you to compress files with specified extensions to save disk space.

Zipped Smart Vault is ideal for extremely large binary files, to save on transfer time. However, the data stored in a Zipped Smart Vault has the same limitations as binary files. Keywords cannot be expanded,diff, merge or end-of-line conversion on text files is

not be applicable. Also, `ci`, `co` and `populate` will always use the `-retain` option (although `-get`, `-keep` and `-lock` files ordinarily has `-noretain` by default).

Vault Type Selection and Conversion

When an object is first checked in, DesignSync uses the Vault Type registry setting to select a vault type for the object. This vault type selection does not change unless the vault is converted to another type.

A DesignSync administrator can set the vault types for a server by using the **Vault Types** option in **SyncAdmin**. See [Setting Vault Types](#) for information.

DesignSync includes a conversion utility, `convertutil`, to convert one vault type to another. See [Converting Vault Data](#) for information.

Related Topics

[Setting Vault Types](#)

[Converting Vault Data](#)

[Vault Types](#)

Setting Vault Types

As a System Administrator, you can use the **Vault Types** option in **SyncAdmin** to set the vault types that you want to use for your data storage on your server. The vault types you set are recorded in the `SiteRegistry.reg` file.

Note: To set vault types for specific ports on a server, launch the DesignSync Web UI and open **Admin Menu | Server | Administer Server | Vault Types**.

For each vault type you set, DesignSync creates the following three registry entries in the `SiteRegistry.reg` file for the server that is currently running. When an object is first checked in, DesignSync uses the registry entries to select a vault type for the object.

- **Pattern**

The pattern is a string that DesignSync uses to map an object to a vault type. DesignSync compares the pattern to the entire server URL path for each object. For information about how DesignSync uses patterns to map objects to a vault type, see [Add or Edit Vault Types](#).

- **Vault Type**

Currently, DesignSync supports three vault types, RCE Vault (.rca file extension) Copy Vault (.cpv file extension), and Smart Vault. Copy Vault and Smart Vault also provide optional compressed Zipped modes. For a description of vault types, see About Vault Types.

- **Priority**

The priority determines the order in which the patterns are compared to an object's server URL path during the initial check-in of the object. Priority values range from 1 (first priority for comparison) to 10 (last). The first pattern that matches is used to set the vault type for the new checkin. The Priority value is important when two or more patterns could match an object's URL.

More than one pattern can have the same priority. If this condition results in multiple pattern matches for a given object, then DesignSync uses a nondeterministic method to select the match that determines the object's vault type.

Setting the Vault Type

1. Start **SyncAdmin**.
2. Open **Site Options**.
3. Click **Vault Types**.

SyncAdmin displays the **Vault Type** options pane, which lists the vault type settings on your server.

Adding a Vault Type

1. From the **Vault Type** options pane (In the Web UI, open **Admin Menu | Server | Administer Server | Vault Types**), click **Add Association**.
2. In the **Add Vault Association** dialog, in the **Pattern** field, enter a pattern. For information, see Add or Edit Vault Types.
3. Select the **Vault Type**.

You have a choice of **RCE Vault**, **Copy Vault**, or **Smart Vault** (Selecting **Copy Vault** or **Smart Vault** offers optional compressed Zipped modes.)

If you select Smart Vault, you have the option to set the maximum size of additional parameters.

4. Enter a value for the **Priority**.

If you entered an asterisk (*) in the **Pattern** field, SyncAdmin uses a priority of 10. The priority for this pattern cannot be changed. For all other patterns, you can choose a priority between 1 and 9. The lower the number, the higher the priority.

5. Click **OK**.

The setting you added appears in the **Vault Type** options pane.

6. Click **Apply** to write your changes to the `SiteRegistry.reg` file and leave SyncAdmin open. Click **OK** to write your changes to the `SiteRegistry.reg` file and close SyncAdmin.

You must restart the server for these settings to take effect.

Editing a Vault Type

1. In the **Vault Type** options pane ((In the Web UI, open **Admin Menu | Server | Administer Server | Vault Types**), highlight the vault type you want to edit and click **Edit Association**).
2. In the **Edit Vault Association** dialog, make the changes you want.
3. Click **OK**.

The setting you changed appears in the **Vault Type** options pane.

Note: If you edited the pattern, SyncAdmin adds a new vault type to the list.

4. Click **Apply** to write your changes to the `SiteRegistry.reg` file and leave SyncAdmin open. Click **OK** to write your changes to the `SiteRegistry.reg` file and close SyncAdmin.

You must restart the server for these settings to take effect.

Deleting a Vault Type

1. Highlight the vault type in the list that you want to delete and click **Delete**.
2. Click **OK** to write your changes to the `SiteRegistry.reg` file and close SyncAdmin.

Note: If you delete a vault type that was previously modified from a setting in a read only registry file, the entry reverts back to its read only setting.

What happens When You Change Vault Types?

The following two scenarios illustrate the possible outcomes of changing vault types and deleting vaults in relation to when you stop the SyncServer.

Scenario 1: Stopping the SyncServer after Removing a Vault

You have a file `myfile.asm`. When you start the SyncServer, the registry setting is set to recognize `myfile.asm` as a CopyVault. You then perform the following operations:

1. Check in `myfile.asm` with the **Allow check in of new items** option (`ci -new`).

DesignSync creates a CopyVault.

2. Remove the `myfile.asm` vault using DesignSync's `rmvault` command.

DesignSync creates a shell CopyVault.

3. Shut down the SyncServer.
4. Change the vault type, using SyncAdmin, so that `*.asm` files are recognized as RceVaults.
5. Restart the SyncServer so the new registry setting takes effect.
6. Check in `myfile.asm` with the `-new` option.

`myfile.asm` will continue to be a CopyVault type.

Scenario 2: Stopping the SyncServer Before Removing a Vault

You have a file `myfile.asm`. When you start the SyncServer, the registry setting is set to recognize `myfile.asm` as a CopyVault. You then perform the following operations:

1. Check in `myfile.asm` with the **Allow check in of new items** option (`ci -new`).

DesignSync creates a CopyVault.

2. Shut down the SyncServer.
3. Use SyncAdmin to change the vault type so that `*.asm` files are recognized as RceVaults.
4. Restart the SyncServer so the new vault type setting takes effect.
5. Remove the `myfile.asm` vault using DesignSync's `rmvault` command.

DesignSync creates a shell RceVault.

6. Check in `myfile.asm` with the `-new` option.

`myfile.asm` is now a RceVault type.

Related Topics

Vault Types

About Vault Types

Converting Vault Data

Changing Vault Types

The **Vault Types** tab page of the Administer Server Settings panel lets you change (edit or add) vault types on your SyncServer. There are three different types of vaults - **RCE** vault, **Copy** vault and **Smart** vault. The Copy vault and Smart vaults provide an optional **Zipped** option that allows you to compress files with specified extensions to save disk. For example, you may be able to store most of your design files in space-saving RCE vaults with a small subset of binary files in copy vaults.

Note: Before changing your vault type settings, you should be familiar with DesignSync vaults and the available vault types. See DesignSync Data Manager User's Guide: Vaults, Versions and Branches and About Vault Types for details.

DesignSync terminology includes two uses of the term vault. In one instance, a vault is an object that holds an individual file and its revisions. In another instance, a vault is a project vault - a repository for all files associated with a revision-controlled project. The **Vault Types** tab page applies to the vault that holds an individual file and its revisions.

The **Vault Types** tab page associates a vault type with a set of files that are identified using a wildcard expression. The **Vault Types** tab page refers to this wildcard expression as a **Pattern**. When you modify existing vault settings or create new ones using the **Vault Types** tab page, DesignSync writes this information to the `PortRegistry.reg` file. A `PortRegistry.reg` file contains settings for an individual SyncServer.

The SyncServer reads the information about vault types from the following files:

- **Server-specific:** `PortRegistry.reg`
- **Site-wide:** `SiteRegistry.reg`
- **Enterprise-wide:** `EntRegistry.reg`. (The enterprise area is reserved for future development.)

In the case of a conflict, the more-local settings take precedence over the less-local settings. For example, settings in the server-specific registry file take precedence over settings in the site-wide registry file.

Although entries in the `SiteRegistry.reg` or `EntRegistry.reg` files may appear in the **Vault Types** tab page, you cannot modify these settings using the Web UI. Use the DesignSync SyncAdmin utility to set vault types for your entire site. (See Vault Types for details.) SyncAdmin writes **Vault Types** information to the `SiteRegistry.reg` file.

You cannot add vault type settings to the `EntRegistry.reg` file in the `enterprise` area.

To open the **Vault Types** page:

1. From the DesignSync Web UI, select the **Admin Menu | Server | Administer Server**.

Note: This menu option appears only when you have permission to perform administrative functions. See the ENOVIA Synchronicity Access Control Guide for details.

2. Click the **Server Settings** hyperlink on the Administer Server:
3. Click the **Vault Types** tab.

The **Vault Types** page lists the current **Pattern**, **Vault Type**, **Pattern** and **Priority** settings:

Pattern - The **Pattern** is a non-verified string that DesignSync uses to assign objects to their vault types. The Pattern value is used in the following ways:

- If the value of Pattern is `*`, DesignSync maps all objects to the assigned vault type.
- If the value of Pattern is `*.tgz`, DesignSync maps any file name that contains `.tgz` to the assigned vault type.
- Because DesignSync compares the objects's entire server URL path, you can use parts of the URL path other than the file name to assign a vault type. For example, to assign objects associated with the Logic Design project to the Zipped vault type, you would specify the Pattern `*/Projects/LogicDesign/*` and select Zipped as the Vault Type.

Note: If the **Pattern** has been specified for your site using SyncAdmin, you cannot re-enter the value. However, you can edit the vault type and priority of the pattern.

Vault Type - There are three supported vault types; the **RCE**, **Copy**, and **Smart** vault types. The default vault type is RCE. (See About Vault Types for information on these vault types.)

Parameters - The **Parameters** column contain values associated with a Copy or Smart zipped vault.

Pattern	Vault Type	Parameters	Priority
☐ *	RceVault		10
☐ *.gz	CopyVault		9
☐ *.bz2	CopyVault		9
☐ *.Z	CopyVault		9
☐ *.tgz	CopyVault		9
☐ *.*.*	SmartVault	zipped	9

Priority - The **Priority** value determines which vault type applies when a file matches more than one **Pattern**. You can use numbers from 1 to 10. When you use a **Pattern** of *, DesignSync uses the default priority of 10. The lower the number of the priority, the higher the precedence. For example:

- You set **Vault Type**, RCE Vault, to have a **Priority** of 10 and a **Pattern** of *.
- You set **Vault Type**, Copy Vault, to have a **Priority** of 3 and a **Pattern** of *.tgz.

In this case, a file called `configuration.tgz` will match both the RCE and copy vault types. The copy vault takes precedence over the RCE vault because the copy vault's priority is numerically smaller than the RCE vault's priority.

To add a new vault type, click Add Association to bring up the **Add Vault Association** page. To edit an existing vault type, click Edit Association to bring up the **Edit Add Vault Association** page. The **Edit Add Vault Association** page is pre-loaded with the Pattern value.

By default, the **Enable Vault Postprocessing** option is selected. A Smart vault uses a post-check-in processing phase to examine the object and make the best storage type decision. Deselect this option if you do not want post processing for your Smart vaults.

After you have made and reviewed your changes and click **Apply**. When you click **Apply**, your new or modified settings on all the tab pages are written to the server registry file. The **Apply** button is disabled until you enter changes on one of the tab pages. If you are not satisfied, click **Delete** to delete the settings you made. When you click **Delete**, a confirmation opens. Click **Ok** to delete or **Cancel** if you want to save your settings.

Note: Deleting a setting from the server does not remove the row from the table when the same pattern is in the site or enterprise registry. Instead, the table is updated to show the settings that are in effect.

Restart your server. The SyncServer reads your server registry file and applies your new settings.

Related Topics

Using the Administer Server Panel

Adding or Editing a Vault Association

Adding or Editing a Vault Association

The **Vault Association** dialog window lets you change or add a vault association on your SyncServer.

Click on the image for more information about each field.

Add Vault Association

Pattern: *.bz2

Configure Vault Type

RCE Vault

Copy Vault Zipped

Smart Vault Zipped

Maximum version size allowed in multi-version file: 262144 KB

Maximum delta size to version size ratio: 50 %

Maximum multi-version file size: 524288 KB

Maximum versions per multi-version file: 0

Priority: 9

OK Cancel Help

Pattern

Specify the pattern you want to use. The **Pattern** is a non-verified string that DesignSync uses to assign objects to their vault types. In adding or editing a vault type, you must specify a pattern.

The order in which patterns are compared to the object URL depends on the **Priority** value for the pattern. The Pattern value is used in the following ways:

- If the value of Pattern is *, DesignSync maps all objects to the assigned vault type.
- If the value of Pattern is *.tgz, DesignSync maps any file name that contains .tgz to the assigned vault type.
- Because DesignSync compares the objects's entire server URL path, you can use parts of the URL path other than the file name to assign a vault type. For example, to assign objects associated with the Logic Design project to the Zipped vault type, you would specify the Pattern `*/Projects/LogicDesign/*` and select Zipped as the Vault Type.

Note: If the **Pattern** has been specified for your site using SyncAdmin, you cannot re-enter the value. However, you can edit the vault type and priority of the pattern.

Configure Vault Type

Choose between the **RCE** Vault (.rca file extension), **Copy** Vault (.cpv file extension), or **Smart** Vault.

The default vault type is RCE. Copy and Smart vaults provide an optional **Zipped** option that allows you to compress files with specified extensions to save disk space. See [About Vault Types](#) for information on these vault types.

Smart Vault Options

If you select Smart Vault, you have the option to set the maximum size of additional parameters:

- **Maximum version size allowed in multi-version file**

Enter the maximum file size in KB. Enter 0 to indicate no maximum size.

- **Maximum delta size to version size ratio**

Enter a percentage between 1 and 100. To turn this parameter off, set it to 0.

- **Maximum multi-version file size**

Enter a file size in KB. Enter 0 to indicate no maximum size.

- **Maximum versions per multi-version file**

Enter the number of versions you wish to allow for each multi-version file. Enter 0 to indicate no maximum number.

Priority

Choose the priority value from the pull down menu. The **Priority** value determines which vault type applies when a file matches more than one **Pattern**. You can specify a value from 1 (highest priority) to 10 (lowest). For example:

- You set **Vault Type**, RCE Vault, to have a **Priority** of 10 and a **Pattern** of `*`.
- You set **Vault Type**, Copy Vault, to have a **Priority** of 3 and a **Pattern** of `*.tgz`.

In this case, a file called `configuration.tgz` will match both the RCE and copy vault types. The copy vault takes precedence over the RCE vault because the copy vault's priority is numerically smaller than the RCE vault's priority.

Note: The Priority value for the `*` pattern is always 10.

After you have made your selection, click **OK** to save the selection you made or **Cancel**, if you do not wish to save the settings.

Related Topics

Using the Administer Server Panel

Changing Vault Types

DesignSync Data Manager User's Guide:Vaults, Versions and Branches

Add or Edit Vault Association

Converting Vault Data

Depending on the type of data you use, you may want to convert to a different vault type for optimal performance. For example, the default DesignSync vault is an RCE vault that stores versions on the server as deltas. For maximum performance using binary data, convert from the RCE vault type to a Copy Vault type.

To convert CVS or RCS data formats to DesignSync format, follow the steps outlined in [Converting a Vault Repository from CVS/RCS Format to DesignSync Format](#).

There are two UNIX scripts you can use to convert vault data:

- `convertutil`
- `convertvault`

Running the convertutil script

DesignSync provides a UNIX shell script, `convertutil`, that lets you recursively convert any vault folder in your directory between the supported vault types:

- RCE Vault
- Copy Vault
- Copy Vault (Zipped)
- Smart Vault
- Smart Vault (Zipped)

You can use this script to convert your existing data. The script is located in `$SYNC_DIR/bin`.

To convert the contents of a vault repository, follow these steps:

1. Ensure you are working on the machine where the vault repository you want to convert resides.
2. Stop the SyncServer using the `stop_sync_server` utility.

During the conversion, you must prevent write access to the server. If you do not want to stop the SyncServer at this time, you can prevent write access to the server by setting access controls on operations that write to server data. Examples of these operations include `ci`, `co -lock`, `pop -lock`, `cancel`, `unlock`, `retire`, `rmvault`.

3. Run the `convertutil` script.

```
% convertutil [-recursive] -outtype
<destination_vault_type>:[mode],<num1>,<num2>,<num3>,<num4> [-outdir <path_to_outdir> ]
<path_to_vault_or_folder>
```

where:

<code>-recursive</code>	Converts the vault hierarchy. Omitting this switch limits the conversion to the contents of the current directory.
<code>-outtype</code> <code><destination_vault_type></code>	The type of vault you are converting to. The choices are: <code>RceVault</code> , <code>CopyVault</code> , or <code>SmartVault</code> . This parameter is required.
<code>mode</code>	Enter <code>zipped</code> to create a zipped vault.
<code>num1 - num4</code>	Identifies user-defined limits.

	<p>To use the default value for each limit, omit the number in the list. Commas must remain for omitted leading numbers, for example: <code>,, , , , 10</code>.</p> <p>A value of 0 indicates no maximum limit.</p> <p>Note: When converting to Smart Vault, the user defined limits are not immediately applied. The next checkin to the converted vault file performs that optimization, according to the user defined settings were specified to the conversion.</p>
num1	The maximum size of the version file in KB. The default is 262,144 KB.
num2	The maximum percentage ratio of the delta size to the version size. The default is 50%.
num3	The maximum size of multiple version files in KB. The default is 524,288 KB.
num4	The maximum number of versions to allow for each multi-version file. The default setting, 0, means that there is no maximum number of versions..
<code>-outdir <path_to_outdir></code>	The path to the new vault or folder including the extension. If you do not include this path, the new vault is created in the same directory as the original vault. If you specify an <code>-outdir</code> , the directory must exist before you run the conversion script.
<code><path_to_vault_or_folder></code>	The path to the vault you want converted. (If you want to convert an individual vault file, use the <code>convertvault</code> utility.)

The script converts any vaults that reside in the `<path_to_vault_or_folder>`. All versions and branches are converted and all metadata information (for example, author, comments, and check in times) is retained.

Note: You must specify the full option name when running the script. Abbreviations such as `-rec` instead of `-recursive` are not recognized.

4. When the conversion is complete and if you have not already done so, stop the SyncServer using the `stop_sync_server` utility.
5. If RCE Vaults were converted to Copy Vaults, remove the `.rca` vaults from the directory. The `.rca` vaults contain the old RCE data. If Copy Vaults were converted to RCE Vaults, remove the `.cpv` vault folders from the directory. Those folders contain the old Copy Vault data.
6. Optionally, change directories so you are in `-outdir`, and copy the converted files from the `-outdir` folder to your current SyncServer by issuing the following command:

```
% find . -print | cpio -o | (cd  
<path_to_vault_or_folder> ; cpio -idum )
```

This command overwrites the contents of the `<path_to_vault_or_folder>` directory. This step uses the `cpio` utility rather than the `tar` command because `tar` has limitations on the directory path length and can be slower than `cpio`.

7. Start the SyncServer using the `start_sync_server` utility.

Examples of running the `convertutil` script:

The following example converts an RCE Vault, `ASIC_top.gds.rca`, to a Copy Vault.

```
csh> convertutil -outtype CopyVault  
/home/syncmgr/syncdata/<host>/<port>/server_vault/Projects/ASIC/  
tapeout/ASIC_top.gds.rca
```

The conversion creates an `ASIC_top.gds.cpv` folder inside of the directory from which the command was run. The `ASIC_top.gds.cpv` folder contains the CopyVault data. After the conversion, remove the old `ASIC_top.gds.rca` RCE Vault, and move the `ASIC_top.gds.cpv` folder into the vault folder `/home/syncmgr/syncdata/<host>/<port>/server_vault/Projects/ASIC/tapeout`.

The following example converts all RCE Vaults that reside in your `Projects` directory, and places the converted files in a directory `$HOME/converted_data`:

```
csh> convertutil -recursive -outtype CopyVault  
-outdir $HOME/converted_data  
/home/syncdata/<host>/<port>/server_vault/Projects
```

The directory specified as your `-outdir` must exist before you run the script.

Running the convertvault Script

On UNIX, you can use a `convertvault` script that provides more flexibility to the advanced user than the `convertutil` script.

You can run the `convertvault` script to convert a single vault type to another supported type, or you can convert multiple vaults at one time.

To convert the contents of a vault repository, follow these steps:

1. Ensure you are working on the machine where the vault repository you want to convert resides.
2. Ensure you have set the vault types in the registry. See [Setting Vault Types](#) for details.
3. Ensure you have sufficient disk space by setting `TMPDIR` prior to running `convertvault`.
4. Stop the SyncServer using the `stop_sync_server` utility.

During the conversion, you must prevent write access to the server. If you do not want to stop the SyncServer at this time, you can prevent write access to the server by setting access controls on operations that write to server data. Examples of these operations include `ci`, `co -lock`, `pop -lock`, `cancel`, `unlock`, `retire`, `rmvault`.

4. Run the `convertvault` script.

In its simplest form, the following syntax will convert a single vault type to another supported vault type:

```
convertvault -intype <source_vault_type>:[mode],
<num1>,<num2>,<num3>,<num4>
-outtype
<destination_vault_type>:[mode],<num1>,<num2>,<num3>,<num4>
<source_vault> <destination_vault>
```

where:

<code>-intype</code> <code><source_vault_type></code>	The type of vault you are converting from. The choices are: <code>RceVault</code> , <code>CopyVault</code> , or <code>SmartVault</code> . This parameter is required.
<code>-outtype</code> <code><destination_vault_type></code>	The type of vault you are converting to. The choices are: <code>RceVault</code> , <code>CopyVault</code> , or <code>SmartVault</code> . This parameter is required.
<code>source_vault</code>	The path and name of the vault you are converting from. Do not include the file extension

	that indicates the type of vault. For example, if your source vault is a Smart Vault named <code>sgc.tst.smv</code> , specify <code>sgc.tst</code> as the source vault. This parameter is required.
<code>destination_vault</code>	The path and name of the vault you are converting to. Do not include the file extension that indicates the type of vault. For example, if you are converting to a Copy Vault and you specify <code>sgc.tst</code> , the <code>convertvault</code> script generates a file named <code>sgc.tst.cpv</code> . This parameter is required.
<code>mode</code>	Enter <code>zipped</code> to create a zipped vault.
<code>num1 - num4</code>	Identifies user-defined limits. To use the default value for each limit, omit the number in the list. Commas must remain for omitted leading numbers, for example: <code>,,,,10</code> . A value of 0 indicates no maximum limit. Note: When converting to Smart Vault, the user defined limits are not immediately applied. The next checkin to the converted vault file performs that optimization, according to the user defined settings were specified to the conversion.
<code>num1</code>	The maximum size of the version file in KB. The default is 262,144 KB.
<code>num2</code>	The maximum percentage ratio of the delta size to the version size. The default is 50%.
<code>num3</code>	The maximum size of multiple version files in KB. The default is 524,288 KB.
<code>num4</code>	The maximum number of versions to allow for each multi-version file. The default setting, 0, means that there is no maximum number of versions..

In the following example, you are converting an RCE Vault, located in `/home/syncdata/server_vault/Projects/myfile`, to a Copy Vault:

```
%convertvault -intype RceVault -outtype CopyVault
/home/syncdata/server_vault/Projects/myfile myfile
```

When the script runs, it expects to find a `/home/syncdata/server_vault/Projects/myfile.rca` vault. The script converts the vault to a Copy Vault and creates a

/home/syncdata/server_vault/Projects/myfile/myfile.cpv directory to store the revisions. All versions and branches are converted and all metadata information (for example, author, comments, and check in times) is retained.

To convert multiple vaults, or if you want to control the conversion in more detail, you can run the script with the following switches:

```
convertvault [-tmpdir <tmpdir>][-recursive|-norecursive][--maxerrors <maxnumerrors>][--log <-logfile>][--report verbose]-intype <source_vault_type> -outtype <destination_vault_type> [dir1 dir1_out file1 file1_out]
```

-tmpdir	The path and name of a temporary directory to use. You can set the default temporary directory on your system. You may need to use a different temporary directory if you have a large amount of data. The translation process checks out every version that resides in the RCA Vault and checks it back into the new Copy Vault, so enough space has to be allocated to accomplish this process.
-recursive -norecursive	The default behavior is to include all subfolders in the hierarchy during conversion. Use -norecursive to exclude subfolders in the hierarchy during conversion.
-maxerrors	The maximum number of errors allowed during conversion before the script stops.
-log	The path and name of the log file you want to use for output messages.
--report verbose	Specifies the most amount of information about the translation to be displayed. If you do not specify 'verbose', the normal amount of data is displayed.
-intype <source_vault_type>	The type of vault you are converting from. For example RceVault, CopyVault, CopyVault:zipped, SmartVault, or SmartVault:zipped. This parameter is required.
-outtype <destination_vault_type>	The type of vault you are converting to. For example RceVault, CopyVault, CopyVault:zipped, SmartVault or SmartVault:zipped. This parameter is required.
dir1 dir1_out file1 file1_out	The full path and names of pairs of directories or files that you want converted. Each pair consists of the source directory or file, and the destination

directory or file.

The script recursively converts the contents of `dir1` and places the output in `dir1_out`.

If you include files, the script then converts `file1` by looking for the corresponding RCE vault file, `file1.rca`. The `convertvault` script creates a new corresponding vault file for the vault type that you specified. For example, `file1.cpv`.

You can also convert from a Copy Vault format to an RCE Vault format.

The following example shows the syntax for converting an RCE Vault, `/home/syncdata/server_vault/testrcevault`, to a Copy Vault, `/home/syncdata/server_vault/testnewcopyvault`, using the temporary directory `/tmp`, with a total of 20 errors during conversion.

```
convertvault -tmpdir /tmp -maxerrors 20 -intype RceVault -  
outtype CopyVault /home/syncdata/server_vault/testrcevault  
/home/syncdata/server_vault/testnewcopyvault
```

`testnewcopyvault` is the destination directory for the vaults in `testrcevault`. The script recursively converts vaults in `testrcevault` to copy vaults and places the output in `testnewcopyvault`.

Related Topics

[Converting a Vault Repository from CVS/RCS Format to DesignSync Format.](#)

[About Vault Types](#)

[Setting the Vault Type](#)

[Access Control Overview](#)

Converting a Vault Repository from CVS/RCS Format to DesignSync Format

To convert a vault repository to DesignSync data format, use the **convertdata** script. This C-shell script converts vault files from Concurrent Version System (CVS) or Revision Control System (RCS) data formats to the DesignSync format.

To convert a vault repository:

1. Run the **convertdata** script on the computer where the vault repository resides. The script is located in the DesignSync installation directory (<SYNC_DIR>/bin).

Note: Conversion of a vault repository may take quite a bit of time, depending on the vault's size and complexity. For example, a sample CVS vault containing 15,000 files, each with 70-90 branches, took three to four days to convert.

In cases where the CVS files have tags for deleted versions, you get a warning. However, data conversion continues. For example:

```
comp.args P0_1:2.2
           invalid magic revision number + [comp.args]
WARNING: Could not add tag (P0_1) to rev ()
P0_2:2.2
           invalid magic revision number + [comp.args]
WARNING: Could not add tag (P0_2) to rev ()
```

2. Import the converted vault files (located in the directory you specified as `destination_ds_dir`) into the SyncServer. Performing this import of the vault files is necessary because it adds the tag information to the tags database. **Note:** The `destination_ds_dir` argument of the `convertdata` utility corresponds to the `-from staging_area` argument of the `importVaults` utility. For information on importing vault files to the SyncServer, see [Using the Vault Utilities](#).

Syntax for invoking the conversion script:

```
convertdata <vault_dir> <destination_rcs_dir>
<destination_ds_dir> <tmpdir> <systmpdir> [no/yes]
```

Where:

vault_dir is the path to the root of the CVS or RCS directory hierarchy you want to convert. Specify the absolute path to the directory. This argument is required. **Note:** The `convertdata` script parses this directory recursively.

destination_rcs_dir is the path to the directory where files reside after they have been converted from CVS to RCS data format. Specify the absolute path to the directory. This argument is required.

If the directory does not exist, the script asks if you want the script to create it. If you answer yes, the script creates the directory. If you answer no, the script exits.

destination_ds_dir is the path to the directory for the data converted from RCS to DesignSync format. Specify the absolute path to the directory. This argument is required.

If the directory does not exist, the script asks if you want the script to create it. If you answer yes, the script creates the directory. If you answer no, the script exits.

tmpdir is the temporary directory that the script uses to perform some intermediate translation steps. Specify the absolute path to the directory. This argument is required.

If the directory does not exist, the script asks if you want the script to create it. If you answer yes, the script creates the directory. If you answer no, the script exits.

Note: To allow enough space to convert large or complex files (those with many revisions), the tmpdir should be at least ten times the size of the data to be converted.

sysmpdir is the temporary system directory for system files created during the conversion of RCS files to DesignSync data format. Specify the absolute path to the directory. This argument is required.

If the directory does not exist, the script asks if you want the script to create it. If you answer yes, the script creates the directory. If you answer no, the script exits.

[no/yes] is the answer to the question: "Are the vault files to be converted in RCS data format?"

no indicates that files are in a data format other than RCS (most likely CVS).

The script first converts the files to RCS format and then to the DesignSync data format.

yes indicates that data is already in the RCS data format.

The script converts the files to the DesignSync data format.

Notes:

- If your files are in RCS data format but have a ,v extension, specify yes for this argument.
- If you do not specify this argument, the script interprets this as a no response and performs the complete conversion (from CVS format to RCS format, and then from RCS to RCA format).

Guidelines for Conversion

In using the convertdata script, follow these guidelines:

You can redirect results via stdout.

By default the script outputs results via stdout.

Tip: When you redirect the output to a file, you can review the output results, or in the event of any problems, forward them to Customer Support for assistance. To redirect the output to a file, add the redirection command to the script invocation:

```
| tee convert_results
```

Note: No matter how stdout is redirected, the convertdata script stores the intermediate output of converting from CVS to RCS and RCS to DesignSync in a file named results, which is local to that directory. There are two types of stdout data:

- Script output - The time a particular conversion began on a directory, as well as the comparison results on whether a destination file exists for each source file converted
- Conversion executable output - Files that contain all versions processed within files that were converted, as well as errors on converting the file based upon content

A `results` file is created in each directory being converted.

Specify the first six arguments.

At least six arguments in exact order are required for the convertdata script to process data. The seventh argument is optional.

- If you do not specify the first five arguments, the script exits.
- The contents of the `destination_rcs_dir` directory (the second argument) depend on the value for argument six ([no/yes]):

If argument six is yes, then the `destination_rcs_dir` directory will contain symbolic links with the following naming convention: `<Filename>.rcs`. This type of reference will exist for each file in the source directory (argument 1).

If argument six is no, then the `destination_rcs_dir` directory will contain the actual converted files.

- The `destination_ds_dir` directory (the third argument) will always contain converted data. This action is always performed.
- If you do not specify the sixth argument ([no/yes]), the script interprets this as a no response and performs the complete conversion (from CVS format to RCS format, and then from RCS to RCA format).
- The contents of the `tmpdir` (the fourth argument) depend on the value of argument seven ([no/yes]):

If argument seven is yes, then the script will use the `keyterminate` option for the second phase of conversion.

If argument seven is no, then the script will use the `[no]keyterminate` option for the second phase of conversion.

You can delete certain directories after conversion.

After conversion is complete, you can delete these directories: `destination_rcs_dir`, `tmpdir`, `sysmtpdir`.

How the `convertdata` handles the CVS dead property.

In converting a CVS vault repository, the `convertdata` script handles the CVS dead property by retiring the branch that the dead property is found on.

For example, suppose X.1 and X.2 are two revisions in a CVS vault repository. If either revision, X.1 or X.2, has the dead property, the `convertdata` script assigns the X branch the status of Retired.

The `convertdata` script provides help information.

To get help on the script invocation and arguments, enter **`convertdata -help`**, **`convertdata -usage`**, or **`convertdata`** (no argument).

You may want to make certain terminal window settings for working in a C-shell.

The `convertdata` script is a C-shell script. When working in a C-shell, many people like to use Control-c as an interrupt. If this is the case, you may want to check that your terminal window has these two settings:

Backspace = 3D ^H (also can be set with `tset -e^H`)

Interrupt =3D CTRL C (also can be set with `stty intr^C`)

After the data conversion, you must import the converted vault files into the SyncServer.

You must import the converted vault files into the SyncServer so that their tag information is added to the central tags database.

Examples of Conversion

Converting a CVS vault repository to a DesignSync vault repository

This example converts CVS vault files in the `/disk1/Projects/projectA` directory to DesignSync data format and stores the converted files in the `~/projectA_ds_dir` directory. In addition, the results output of the conversion process is directed to a file called `projectA_results`.

```
csh> convertdata /disk1/Projects/projectA ~/projectA_rcs_dir
~/projectA_ds_dir ~/tmpdir no ~/systmpdir no |tee
~/projectA_results
```

In this example:

- /disk1/Projects/projectA is the directory of vault files to be converted.
- ~/projectA_rcs_dir is the directory where the script will put the files converted to RCS data format.
- ~/projectA_ds_dir is the directory where script will put the files converted to DesignSync data format.
- ~/tmpdir is the temporary directory where the script puts intermediate versions of the files during conversion. In this case, normal (cvs2rcs) conversion will occur
- ~/systmpdir is the system temporary directory where the script puts system files created during conversion.
- The no argument indicates that vault files are not in RCS data format and therefore need to be converted to RCS.
- The |tee parameter directs results output to a file called projectA_results.

Converting an RCS vault repository to a DesignSync vault repository

This example converts RCS vault files in the /disk1/Projects/projectB directory to DesignSync data format and stores the converted files in a directory called ~/projectB_ds_dir. In addition, the results output of the conversion process is directed to a file called projB_results.

```
csh> convertdata /disk1/Projects/projectB ~/projectB_rcs_dir
~/projectB_ds_dir ~/tmpdir yes ~/systmpdir yes |tee
~/projectB_results
```

In this example:

- /disk1/Projects/projectB is the directory of vault files to be converted.
- ~/projectB_rcs_dir is the directory where the script will put the files converted to RCS data format.
- ~/projectB_ds_dir is the directory where script will put the files converted to DesignSync data format.
- ~/tmpdir is the temporary directory where the script puts intermediate versions of the files during conversion. In this case, conversion with RCS input data files with a ,v extension will occur.

DesignSync Data Manager Administrator's Guide

- `~/systmpdir` is the system temporary directory where the script puts system files created during conversion.
- The `yes` argument indicates that vault files are already in RCS data format.
- The `|tee` parameter directs results output to a file called `projectB_results`.

Reconverting Data Files

If you need to reconvert (run **convertdata** again on the same files), follow these steps:

1. Delete the directories you specified for: `destination_rcs_dir`, `destination_ds_dir`, `tmpdir`, and `systmpdir`.
2. Make sure resources are available and that there are no duplicate files.
3. Run the **convertdata** script.

Exporting Vaults

To export vault data, use DesignSync's Export Project feature. This method of exporting your vault preserves note attachments, version identifiers, and user-defined properties. See [ProjectSync User's Guide: Exporting Projects](#) for details.

To export data from client vault folders, use the `exportVaults` utility. See [Using the Vault Utilities](#) for details.

To export modules, along with their versions, history, and hierarchical references, see [Module Export and Import](#).

Related Topics

[ProjectSync User's Guide: Exporting Projects](#)

[Moving Vaults](#)

[Importing Vaults](#)

[Converting a Vault Repository from CVS/RCS Format to DesignSync Format](#)

[DesignSync Data Management User's Guide: Moving and Renaming Files](#)

[DesignSync Data Management User's Guide: Moving and Renaming Folders](#)

Importing Vaults

To import vault data, use the Export Project feature. This method of exporting your vault preserves note attachments, version identifiers, and user-defined properties. See *ProjectSync User's Guide: Exporting Projects* for details.

To import data from client vault folders or data converted using the `convertdata` utility, use the `importVaults` utility. See *Using the Vault Utilities* for details.

Related Topics

ProjectSync User's Guide: Importing Projects

Moving Vaults

Converting a Vault Repository from CVS/RCS Format to DesignSync Format

DesignSync Data Management User's Guide: Moving and Renaming Files

DesignSync Data Management User's Guide: Moving and Renaming Folders

Moving Vaults

In most cases, you use the import- and export-vault functionality to move server vaults. This method of exporting your vault preserves note attachments, version identifiers, and user-defined properties. (See *ProjectSync User's Guide: Exporting Projects* for details.) DesignSync also lets you move vault files and vault folders within a SyncServer. The following sections describe these methods of moving vault files and folders:

- Moving vault files and folders within a SyncServer
- Moving a client vault
- Moving CVS or RCS vault folders
- Importing a Vault Object into a Vault Folder

Moving Vault Files and Folders within a SyncServer

- Use the `mvfile` command to move or rename a specified local object.
- Use the `mvfolder` command to move or rename a specified local folder. When the local folder is moved, DesignSync will move the corresponding folder's vault configuration.

Moving a Client Vault

To export data from client vault folders, use the `exportVaults` utility. The following example shows how you use the `exportVaults` and `importVaults` utilities to move client vault folders. (See *Using the Vault Utilities* for details on using these utilities.)

1. Export the vault folder from the client vault repository to a staging area:

```
exportVaults -from  
file:///net/orion/usr1/roger/Projects/Sportster/synth -  
to ~/staging_area -remove
```

The `exportVaults` utility moves the vault folder from the original location to the `~/staging_area` directory, without creating a tag database. No `-root` option is needed when you are exporting from a client vault.

Caution

Ensure you are using the `file:///<local_path_to_export>` notation to export the vault folder from a client repository. It is possible to use the `exportVaults` utility to successfully export a server repository using the `exportVaults -from file:///<local_path_to_export>` notation. This results in the server vault being treated as a client vault and the tags database is lost.

2. Import the vault folder from the staging area into another SyncServer:

```
importVaults -root ~/syncmgr/syncdata/server_vault -from  
~/staging_area -to  
sync://milkyway:2647/Projects/mySportster
```

The `importVaults` utility moves the vault folder from the `~/staging_area` directory into the `milkyway:2647/Projects/mySportster` vault repository. The utility also generates the tag database associated with the vault folder being imported. Use the `-root` option to specify the server data root directory of the `milkyway:2647` SyncServer to which you're importing. **Note:** The SyncServer into which you are importing must be a version 3.0 or higher SyncServer.

Moving CVS or RCS Vault Folders

To move a CVS or RCS vault repository into a version 3.0 or higher SyncServer, you

1. Using the `convertdata` utility, convert the CVS/RCS vault folders into DesignSync format in the staging area. Note: the staging area corresponds to the `<destination_ds_dir>` argument of the `convertdata` utility. See [Converting a Vault Repository from CVS/RCS Format to DesignSync Format](#).
2. Import the DesignSync data from the staging area into the SyncServer.

Note: The SyncServer into which you are importing must be a version 3.0 or higher SyncServer.

Importing a Vault Object into a Vault Folder

Another type of move operation is to import a vault object from an existing vault folder into another vault folder. To perform this type of "switch vault" operation, you

1. Import the vault object from the existing vault folder into your work area using the `import` command.
2. Ensure that your work area is associated with the vault folder into which you want to import the vault object.

Use the `setvault` command to associate the work area with the vault folder.

3. Check the vault object into the new vault folder using the `ci` command.

Related Topics

ProjectSync User's Guide: Importing Projects

ProjectSync User's Guide: Exporting Projects

Exporting Vaults

Importing Vaults

Converting a Vault Repository from CVS/RCS Format to DesignSync Format

DesignSync Data Management User's Guide: Moving and Renaming Files

DesignSync Data Management User's Guide: Moving and Renaming Folders

Using the Vault Utilities

In most cases, you use `import-` and `export-vault` functionality to move server vaults. This method of exporting your vault preserves note attachments, version identifiers, and user-defined properties. See *ProjectSync User's Guide: Exporting Projects* for details. If you need to move a client vault, you can use the `ExportVaults` and `ImportVaults` utilities.

Note: You can also import or export modules. For more information, see *Module Export and Import*.

ExportVaults Utility

The `exportVaults` utility is available from the UNIX command line prompt; the utility is located in `<SYNC_DIR>/bin`. The `exportVaults` utility is also autoloaded as an `stcl` procedure available within `stcl(c)` sessions. Whether you invoke the `exportVaults` script from the UNIX command prompt or within an `stcl(c)` session, the usage, described below, is the same.

Follow these guidelines when using `exportVaults`.

The `exportVaults` utility does not export these forms of metadata:

- Object attachments for notes, such as when adding a note.
- Properties users have added (from the `url setprop` command)

For a discussion of other guidelines for the `exportVaults` utility, see [Moving Vaults Using exportVaults and importVaults](#).

- The `exportVaults` utility does not recognize the `-keepvid` setting. Therefore, if you plan to create new vaults on the same server and in the same location, you should run the `exportVault` utility without the `-remove` option. After you restart the server, use DesignSync's `rmvault` or `rmfolder` command with the `-keepvid` option to remove the original vaults.

ExportVaults Usage

```
exportVaults [-root <server_data_root>]
             -from <url_to_export> -to <staging_area>
             [-remove] [-verbose] [-batch] [-help] [-usage]
```

server_data_root is the local directory path to the server data root. The **-root** option is required only if you're exporting from a server vault, not a client vault. The server data root is the value of the registry setting,

HKEY_LOCAL_MACHINE\Software\Synchronicity\Directories\ServerData Root, in the `PortRegistry.reg` file in the `<SYNC_DIR>/custom/servers/<host>/<port>` directory.

url_to_export is the URL of the vault to be exported, for example, `file:///<client_vault_path_to_be_exported>` or `sync://<host>:<port>/<server_vault_path_to_be_exported>`.

Notes

Remember to use `sync://<<vault_path_to_export>` notation for server vaults, and `file:///<local_path_to_export>` notation for client vaults.

To remove tags be sure to use the `sync://<vault_path_to_export>` notation.

If you use `file:///<local_path_to_export>` notation with the `-root` option, the `exportVaults` utility assumes that you are exporting a client vault and no tags are removed because the tags database only exists on the server. If you subsequently create new files with the same names and check them in, the tags that exist in the tags database on the server are assigned to the new files.

staging_area is a local work area where vaults are copied. Note that vault files must be in a subdirectory of the `staging_area` directory.

-remove removes all exported vault files and tag data (if a tags database exists) from the original vault repository. Any `-keepvid` registry setting is ignored and the vaults are totally removed. You will see a warning message.

-verbose provides additional processing information.

-batch circumvents the warning and prompt that you see when you use the `-remove` option

-help displays this usage message.

-usage displays this usage message.

ImportVaults Utility

The `importVaults` utility is available from the UNIX command line prompt; the utility is located in `<SYNC_DIR>/bin`. The `importVaults` utility is also autoloaded as an `stcl` procedure available within `stcl(c)` sessions. Whether you invoke the `importVaults` script from the UNIX command prompt or within an `stcl(c)` session, the usage, described below, is the same.

Use the `importVaults` utility to import data converted using the `convertdata` utility or data exported from a client vault, .

Follow these important guidelines when using `importVaults`:

- The SyncServer into which you are importing must be a version 3.0 (or higher) SyncServer.

You can export from a SyncServer that is not a version 3.0 SyncServer, but you can only import to a version 3.0 (or higher) SyncServer. The `importVaults` utility operates on a tag database, which is only supported by version 3.0 (or higher) SyncServers.

- The `importVaults` utility does not import these forms of metadata:
 - Object attachments for notes, such as from RevisionControl note generation.
 - Properties users have added (from the `url setprop` command)

For a discussion of other guidelines, see [Moving Vaults Using `exportVaults` and `importVaults`](#).

ImportVaults Usage

```
importVaults -root <server_data_root> -from <staging_area> -to  
<url_to_import_to> [-remove] [-verbose] [-help] [-usage]
```

Where

server_data_root is the local directory path to the server data root. The server data root is the value of the registry setting, `HKEY_LOCAL_MACHINE\Software\Synchronicity\Directories\ServerData Root`, in the `PortRegistry.reg` file in the `<SYNC_DIR>/custom/servers/<host>/<port>` directory.

staging_area is a local work area where vaults are copied. Note that vault files must be in a subdirectory of the `staging_area` directory.

url_to_import_to is the URL of the parent vault folder of the vault folder being imported. For example, suppose that after a vault export your staging area contains `/home/staging/Projects/Sportster` and `Sportster` is the vault folder you want to import. For `url_to_import_to`, you would specify `sync://srvr1:2647/Projects`.

Note: The SyncServer into which you are importing must be a version 3.0 (or higher) SyncServer.

-remove removes the staging area directory (all the vault files and the temporary tags database).

-verbose provides additional processing information.

-help displays this usage message.

-usage displays this usage message.

Backup and Restore

Backing Up Your Server

The Back Up Server panel lets you back up your SyncServer. You can use the backup functionality to generate the data you would need to restore your SyncServer. You can perform both full and incremental backups.

For example, you might want to perform a full backup on a set day each week and do incremental backups on the other days of the week. At the end of the week, you could archive the backups. In addition to routine backups, you should back up your server before you change your database.

To access the Back Up Server panel:

1. Click **Administer Server** in the **Server** section of the **Admin** menu. This menu option appears only when you have permission to perform server operations. (See the ENOVIA Synchronicity Access Control Guide for details.)

The Administer Server: Choose Operation panel appears.

2. Click **Backup** to go to the Back Up Server panel.

You use the backup functionality in conjunction with your standard system backup procedures. To create a backup that lets you safely restore your server, you need to:

1. Use the Back Up Server panel to perform a DesignSync backup operation.

This operation generates backup data in a directory called `Backup.sync` in the `server_vault` area.

You also can create a Tcl script to perform backups using the `backup` command. See the ENOVIA Synchronicity Command Reference: The backup Command help for details on using this command.

2. Perform a standard system backup of your `server_vault` area and your `<SYNC_CUSTOM_DIR>`.

The data stored in the `Backup.sync` directory creates a server snapshot that, in conjunction with a standard system backup, lets you safely restore your server and vault.

Note: If you delete a vault folder (using the `rmfolder` command), any backup files it contains are deleted. If you delete a vault folder before performing a tape backup, your system backup may be incomplete.

The backup operation includes the server vault, metadata, attachments, and notes. The backup preserves the following directories:

- The Postgres database containing the server metadata.
- The entire `server_vault` hierarchy
- The required parts of the `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/data` hierarchy.

Only attachments are copied from the `<SYNC_CUSTOM_DIR>` areas. Note customizations stored in the custom areas are not backed up.

For objects stored in Copy Vault format and for Cadence collection objects, all files associated with a modified file are backed up. In the case of Cadence collections, if any file below a view has been modified since the previous backup, then the entire view directory is backed up.

Backing Up the SyncServer

To back up your server:

1. Select one of the following options on the Back Up Server panel:
 - **Full** - Backs up all vault data, metadata, notes, and attachments.

You must perform a full backup after you upgrade to a new version of DesignSync. You cannot create an incremental backup on top of a full backup from an earlier version of the software.
 - **Incremental** - Based on timestamps, backs up all vault data, metadata, notes, and attachments that have been updated since your last full or incremental backup.

The **Incremental** option does not appear until you have performed the first full backup.
2. If you want an incremental backup, choose whether to back up from the last full backup or from the last incremental backup. The table displays only the last full backup if you have not yet performed an incremental backup.
3. Click **Continue** to run the backup.

The estimated time for a full backup assumes that all data is on the same partition and that your network and disks are not overloaded.

The Operation Successful panel displays the log file, which shows where the data is copied. The data is backed up to the directory:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/Backup.sync
```

This directory contains a subdirectory for each backup. The subdirectory name has the following format:

```
<year><month><day>_<hour><minute><second>
```

The time is in 24-hour format according to the local time zone of the server.

Within the dated subdirectory, the backed-up data is stored in the following directories:

- `Attachments` - Hard links to the note attachment and definition files in the server's `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/data` area.
- `server_metadata/` and all directories below it - Copies of the metadata.
- `server_vault/` and all directories below it - Copies of the tags database are stored in the `server_vault/Partitions` subdirectories. Links to the server vault are stored in the `server_vault` directory.

For example, data in:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/  
Projects/P1/file1.rca
```

might be backed up to:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/  
Backup.sync/20030710_145601/server_vault/Projects/P1/file1.rca
```

The `Backup.sync` directory also contains the backup log file, with the name `<date>_<time>.log`.

The `Backup.sync`, `Import.sync`, and `Export.sync` directories under `server_vault` are not backed up. (See *ProjectSync User's Guide: Importing Projects and Exporting Projects* topics for information about the import and export directories.)

During the backup, `.inprogress` is appended to the dated subdirectory of `Backup.sync`. This extension is removed when the backup completes. You can use this feature to check the status of your backup. If `.inprogress` is appended to the subdirectory name and you cannot write to the server, the backup is still underway. If `.inprogress` is appended to the subdirectory name and you can write to the server, the server crashed and restarted while the backup was underway.

If the server crashes during a backup, all the backup locks are released.

Note: The backup creates symbolic links to vault data if your `Backup.sync` area is on a different partition from your server or if your system does not support hardlinks. If the

vault data is removed (for example, by an `rmfolder` command) the backed-up symbolic links cannot be used to restore your data. To avoid this problem, archive your `Backup.sync` area using a switch to de-reference the symbolic links.

You can use the DesignSync web UI to restore vault data. See [Restoring Your Server Vault Data](#) for details.

To restore all of your server data - vault data, metadata, and note data - you must use the `restoreserver` script. See [Restoring All Server Backup Data](#) for details.

Using the Server During Back-Up Operations

The backup operation does not make the server completely inaccessible. The server is suspended for a short time at the beginning of the backup but after that the server is accessible for all operations. DesignSync users have read access to operations after the initial suspension. For example, after the initial suspension, DesignSync users can populate from a vault during a backup.

If a user is performing a lengthy vault operation (such as `populate`) when the backup is initiated, the user's operation fails during the backup. When the backup is completed, the vault operation can continue. However, data transferred during the backup may be incomplete. If the "Retries to server in maintenance mode" option is enabled, the operation will attempt to resume as defined by the setting. If the retry option is not set, the user may manually retry the operation when the server is accessible. The "Retries to server in maintenance mode" option is on the Performance options pane in SyncAdmin.

Cleaning Up Old Back-Up Data

For each dated backup subdirectory, a `.cleanup` executable file is generated in the `Backup.sync` directory. You can run this executable to remove the corresponding backed-up data. The `.cleanup` file removes itself at the end of the clean-up operation, but does not delete the backup log file.

For example, if your backup directory is:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/  
Backup.sync/20010710_145601
```

You would enter `20010710_145601.cleanup` to remove the data in the `20010710_145601` directory.

To differentiate full backups from incremental backups, look at the `Backup.sync/<date>_<time>.log` file. It will begin with either:

```
[2004-04-01 12:26:23] Full Backup started.
```

or

[2004-04-01 12:27:21] Incremental Backup started.

In a script you can loop over the old files to clean up a group of backup directories. For example, to clean up all the backup directories for May 2001, you could specify a csh script like the following:

```
foreach file (200105*.cleanup)
    $file
end
```

If the server crashes during a backup, the `.cleanup` file may not be created or may be incomplete. In this case, you need to manually clean up the backup directory:

1. Remove all the hard links from the server.

Use the UNIX `find` command to search the `server_vault`, `server_metadata`, and `<SYNC_CUSTOM_DIR>` directories for files that match `.SYNC.*.bck.DATE`. Delete these files.

2. Change the mode of the dated backup subdirectory (`<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/Backup.sync/<date>`) to writable by user, for example:

```
find <SYNC_DIR>/../syncdata/<host>/<port>/server_vault/Backup.sync/20060626_103714 -type d -exec chmod u+w {} \;
```

3. Use the UNIX `rm -rf` command to delete the dated backup subdirectory in `<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/Backup.sync`.
4. Use the UNIX `rm` command to remove the `.cleanup` file, if it exists.

Related Topics

[Restoring All Server Backup Data](#)

[Restoring Your Server Vault Data](#)

Restoring All Server Backup Data

You can use the `restoreserver` script when you need to completely restore your backed up server data. This script lets you restore not only vault folders but also metadata, notes, and attachments. When you restore a Cadence view, all the objects in the view are restored.

Your SyncServer and database server are shut down during the restoration and restarted again when the operation is complete. For this reason, you must be the server owner to run the `restoreserver` script. In order to perform the restore operation, you must enter the database password.

Preparing to Restore

To restore your backed-up vault data, you need to restore the `server_vault` and `<SYNC_CUSTOM_DIR>` data from your system backup and then transfer the data in the `Backup.sync` area to the correct place within the server. To restore your server:

1. Stop the server using the `stop_sync_server` command.
2. Move the most recent `server_vault` directories from

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault
```

to a safe location.

3. Move your `<SYNC_CUSTOM_DIR>` to a safe location
4. If you are storing the backup data on a tape or disk, unpack the `server_vault` data into:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/
```

This step restores the `Backup.sync` area, which is included in the `server_vault` directory. To restore from an incremental backup, all intermediate backups after the last full backup must be available in the `Backup.sync` area.

5. Unpack the `<SYNC_CUSTOM_DIR>` data into:

```
<SYNC_CUSTOM_DIR>
```

Using the restoreserver Script

Run the `restoreserver` script from the command line on the system where you run your SyncServer(s).

1. To invoke the script, enter the following on the command line:

```
restoreserver
```

Before running the `restoreserver` script, DesignSync will prompt you to enter the database password. The script starts and opens a log file, `restoreserver.log`, in your home directory.

2. If your SyncServer has more than one port, the script prompts you to choose the port you want to restore. Enter the number for the port.
3. If you have more than one set of backup data, the script prompts you to choose the backup data that you want to restore from. The script displays a numbered list showing each dated incremental and full backup. Enter the number for the date you want to restore from.

If you choose an incremental backup date, the script first restores the last full backup and then restores each subsequent incremental backup.

Vault data is backed up based on the timestamps of the vault objects. Thus, if a user has a lock on an object when the vault is restored, the user's lock might be lost.

Restoring from incremental backups may restore unwanted files. For example, suppose that a vault object is backed up in a full backup and then deleted before the next incremental backup. When you restore your server back to the last full backup, this deleted vault object will be restored.

Note: If the `restoreserver` script is interrupted in the middle of data restore, the data can be recovered by running the `restoreserver` script again.

When you have restored your server data:

1. Test your restored server until you are satisfied that it works correctly.
2. Delete the copies of the most recent `server_vault` directories from the safe area when you are sure you no longer need them.
3. Delete the backup files created by the `restoreserver` script:

```
$SYNC_DIR/syncdata/<host>/<port>/server_metadata/  
postgresql.saved_by_restoreserver_<date>_<time>
```

```
$SYNC_DIR/syncdata/<host>/<port>/server_metadata/  
data.saved_by_restoreserver_<date>_<time>
```

Related Topics

[Backing Up Your Server](#)

[Restoring Your Server Vault Data](#)

Restoring Your Server Vault Data

The Restore From Backup panel lets you restore individual vault items or small portions of vault data from a back up. You can restore both vault folders or individual vault objects. When you restore a Cadence view, all the objects in the view are restored.

This panel does not restore:

- Backed-up metadata, notes, and attachments. For a full restoration, you must use the `restoreserver` script. See Restoring All Server Backup Data for details.
- Data for the Hierarchical Configuration Manager.
- Vaults backed up from pre-4.0 versions of Developer Suite.

You also can restore vault data using the `restorevault` command. See the ENOVIA Synchronicity Command Reference for details.

Preparing to Restore

To restore portions of your backed-up vault data, you need to restore the `server_vault/Backup.sync` data from your system backup and transfer the data to the correct place within the server. Before you begin:

- If you are storing the backup data on a tape or disk, unpack the `server_vault/Backup.sync` data into:

```
<SYNC_DIR>/../syncdata/<host>/<port>/server_vault/Backup  
.sync
```

When unpacking from a tape or remote disk, fetch only the `server_vault/Backup.sync` areas. Do not overwrite any other section of the `server_vault` area

- If you are restoring from an incremental backup, make sure that all intermediate backups after the last full backup are available in the `Backup.sync` area.

Using the Restore From Backup Panel

You can use the Restore From Backup panel to restore vault data, including vault folders or individual vault objects.

The server is locked while you perform a restoration and any attempts to access the server will hang until the restoration is completed.

To access the Restore From Backup panel:

1. Click **Administer Server** in the **Server** section of the **Admin** menu in the DesignSync Web UI. This menu option appears only when you have permission to perform server operations. (See the ENOVIA Synchronicity Access Control Guide for details.)

The Administer Server: Choose Operation panel appears.

2. Click **Restore** to go to the Restore From Backup panel

To restore your vault data:

1. Under **Enter the backup date to restore from**, select the date of the backup that you want to restore. The text box lists the most recent full or incremental backup.

Click **Browse** to bring up the Select backup date window, where you can choose an earlier backup to restore from. Click the radio button in the **Select** column to enter the date in the text box on the Restore From Backup panel. Click **Close Window** when you have made your selection.

The date you choose depends on what type of data you are restoring:

- If you are restoring a single vault object, you can restore from the last incremental backup (assuming that the object you want to restore was included in that incremental backup).
 - If you are doing a complete restoration and you have made incremental backups, you must first restore the last full backup and then restore each subsequent incremental backup, in order, on top of the last full backup.
2. Under **Enter the vault path to restore**, select the vault or vault object that you want to restore.

Click **Browse** to bring up the Select path window, where you can navigate the vaults on your server. Click a hyperlinked path name to go down one level in the vault hierarchy; click `..` in the top row of the table to go up a level.

When restoring backup data for a Module, navigate to the `/Modules` folder. You can also restore individual module members.

However, you would not need to restore module members under normal circumstances. It is needed only if the member vault file was mistakenly deleted or corrupted.

Click the radio button in the **Select** column to enter the path for an object in the text box on the Restore From Backup panel.

If you want to restore the entire contents of your last full backup, type `' / '` in the text box on the Restore From Backup panel. Note that the best way to completely restore your backed-up server data is to use the `restoreserver` script (see Restoring All Server Backup Data for details.)

3. Optionally deselect the **Overwrite** checkbox if you do not want to overwrite existing vault data. By default, existing vault objects are overwritten. See the Use Models section that follows for more information on how to use this option.

Vault data is backed up based on the timestamps of the vault objects. Thus, if a user has a lock on an object when the vault is restored, the user's lock might be lost.

Restoring from incremental backups may restore unwanted files. For example, suppose that a vault object is backed up in a full backup and then deleted before the next incremental backup. When you restore your server back to the last full backup, this deleted vault object will be restored.

Use Models

When you need to restore from incremental backups, the order in which you restore depends on how you use the **Overwrite** option:

- If **Overwrite** is enabled, you should first restore the last full backup. Then you should restore each subsequent incremental backup, in order, on top of the full backup.
- If **Overwrite** is not enabled, then you should first restore the most-recent incremental backup and then restore each previous incremental backup, in order. You should finish by restoring the last full backup.

The following cases show how to use the **Overwrite** option to restore.

Suppose you have the following situation. A vault directory contains the following files: `dir1/file1.txt`, `dir2/file2.txt`, `dir3/file3.txt`. These files were backed up as follows:

File	April 1: Full Backup	April 2: Incremental Backup	April 3: Incremental Backup
<code>dir1/file1.txt</code>	version 1.3	version 1.4	version 1.5
<code>dir2/file2.txt</code>	version 1.3	version 1.4	
<code>dir3/file3.txt</code>	version 1.3		

The file `dir3/file3.txt` did not change after April 1st, so it was not included in the incremental backups on the following two days. The file `dir2/file2.txt` did not change after April 2nd, so it was not included in the incremental backup on April 3rd.

Later in the day on April 3rd, a user accidentally deletes some vault data and wants to recover it.

Case 1

Suppose that the user deleted `dir2/file2.txt` and `dir3/file3.txt`, but not `dir1/file1.txt`. To restore the files, you need to:

1. Restore the April 3rd backup with the **Overwrite** option disabled. Nothing is restored, because `dir1/file1.txt` exists.
2. Restore the April 2nd backup with the **Overwrite** option disabled. Version 1.4 of `dir2/file2.txt` is restored and `dir1/file1.txt` is not affected.
3. Restore of the April 1st backup with the **Overwrite** option disabled. Version 1.3 of `dir3/file3.txt` is restored and the other files are not affected.

Case 2

Suppose that the user deleted the latest version of the three objects and you want to restore all of the objects from last backup. To restore the files, you need to:

1. Restore the April 1st backup with the **Overwrite** option enabled. All three files are restored to version 1.3 and any existing versions of these files are overwritten.
2. Restore the April 2nd backup with the **Overwrite** option enabled. Versions 1.4 of `dir1/file1.txt` and `dir2/file2.txt` are restored and the previously restored values are overwritten.
3. Restore the April 3rd backup with the **Overwrite** option enabled. Version 1.5 of `dir1/file1.txt` is restored and the previously restored value is overwritten.

Related Topics

[Backing Up Your Server](#)

[Restoring All Server Backup Data](#)

Procedure for Defining Automatic Backups

DesignSync provides a method for backing up your DesignSync data that does not require you to shut down or restart your server when the backups are being created.

The DesignSync backup command allows you to back up the entire SyncServer setup, including vault, metadata, and notes.

After the DesignSync backup has completed and the backup information has been saved to the `server_vault/Backup.sync` area, use your preferred system backup tool to create a system backup that includes the `server_vault` area and the `$SYNC_CUSTOM_DIR`.

This procedure explains the process of defining and configuring automated backups of DesignSync data. Setting up the Backups involves the following steps:

- Identifying your requirements
- Adding Access Controls for Automated Backups
- Configuring automated backups
- Disabling automated backups

Identifying your requirements

A single system backup should expect to be roughly double the size of the server installation + the customization (\$SYNC_CUSTOM_DIR) area, if \$SYNC_CUSTOM_DIR is outside the \$SYNC_DIR area.

DesignSync recommends a backup system that does a full system backup once a week and performs once daily incremental backups at a period when the system is lightly used. Before setting up the backup system, plan your space requirements according to the amount of full and incremental backups you will keep before automatically expiring them, which removes them from system.

After the backups have been written they are immediately available for archival purposes, such as writing to tape for off-site storage until they expire.

Adding Access Controls for Automated Backups

Typically when automated backups are configured, they are run by user, 'nobody.' In order to run the backups, the 'nobody' user must have access rights to suspend the server. By default, all users have access to run suspend. If you have modified the Access Controls to restrict access for suspend, you must add access for user, 'nobody.'

Adding Access Control for User 'nobody'

1. Open the appropriate access control file (enterprise, site, or server) or launch the ACAdmin application,
2. Add access for user, "nobody."

```
access allow Suspend only users {nobody ...} -because  
"Suspend access is reserved for Administrative Use"
```
3. Save the changes and close the access control file, if applicable.

Configuring Automated Backups

Automated backups are configured and maintained using DesignSync registry keys. You can set these keys either server-wide or site-wide. They should not be set in the user registry.

The settings can be made to either the server's

\$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's

installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect.

Note: If you have never used DesignSync automated backups, these registry keys may not be present in your registry files. If the keys do not exist, add them and set the values.

Configuring Automated Backups

1. Open the appropriate registry file in your file editor or use the `sregistry` command.
2. Set the following key to the appropriate time in the future.

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupEarliestTime="MM/DD/YYYY HH:MM"
```

For example:

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupEarliestTime="02/09/2014 21:55"
```

starts the first backup at 9:55 PM local time on February 9th, 2014.

Important: You must set the Earliest Backup Time registry key to a time in the future in order for automated backups to begin.

3. Set the backup frequency to the number of hours, in hexadecimal, between backup beginning times. This is typically one a day (a hexadecimal value of 18).

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupFrequency=dword:18
```

Important: This key must be set to control the frequency of the automated backups. If it is not set, the backup will only run at the earliest backup time.

4. Determine whether to run full backups on a set schedule or perform a certain number of incremental backups between each full backup. Regardless of the method, DesignSync recommends a backup strategy that runs a full backup every week and daily incremental backups. Pick one of the following methods:

- **Scheduling Full Backup by Days of the Week:**

Enter a numeric value for each day (1=Monday, 2=Tuesday, etc.) on which a full backup should be performed in a comma separated list. This example shows a backup schedule of one full backup every week on Sunday.

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupFullBackupDay="7"
```

Notes:

If `BackupFullBackupDay` is set, but full backup has not been executed for more than 3 weeks, DesignSync forces a full backup regardless of the day of the week.

The `BackupFullBackupDay` key requires that the `BackupFrequency` be set to 24 hours, or multiple of 24 hours.

- **Scheduling Full backups after a certain number of incremental Backups:**

Set the number of incremental backups between full backups, in hexadecimal. By default, this is set to 6 meaning that a full backup runs one day a week. This example shows a backup strategy that runs a full backup every week and daily incremental backups (hexadecimal value 6).

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupNumberOfIncrementals=dword:6
```

5. By default, backups expire after 169 hours (a hexadecimal value of `a9`). This is based on optimal scheduling of daily incremental and weekly full backups, using this formula:

$(\text{BackupNumberOfIncrementals} + 1) * \text{BackupFrequency} + 1$.

You can change this value using the following key:

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupRetentionTime=dword:<value>
```

6. By default, backup logs expire after 121 hours (a hexadecimal value of `79`). This value is based on the optimal scheduling of nightly incremental and weekly full backups, using this formula:

$\text{BackupFrequency} * 5 + 1$

You can change this value using the following key:

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupLogRetentionTime=dword:<value>
```

7. Save the registry file and restart the server.

Disabling Automated Backups

To disable automated backups, set the backup frequency time to 0.

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupFrequency=dword:0
```

Related Topics

Backing up your Server

Restoring All Server Backup Data

Restoring Your Server Vault Data

Earliest backup time (`BackupEarliestTime`)

Backup Frequency (`BackupFrequency`)

Schedule for Full Backups (`BackupFullBackupDay`)

Length of time to retain backup logs (`BackupLogRetentionTime`)

Number of incremental backups (`BackupNumberOfIncrementals`)

Length of time to retain backups (`BackupRetentionTime`)

Moving SyncServers

As a DesignSync administrator, you can move an existing SyncServer to a new host and port.

Important: The procedure described here is intended for moving an existing SyncServer to a new location. Two cautions apply:

- You cannot use this method for moving a SyncServer to the location of an existing SyncServer. This procedure overwrites the `syncdata` hierarchy of the server to which you are relocating.
- The vault and custom areas of the old and new server must not overlap.

Before you move a SyncServer:

- Make sure that the file systems for the original and new server are accessible from each other's installations so that you can transfer files between the installations.
- Make sure that the old and new server owners have their `$$SYNC_DIR` and `$$SYNC_CUSTOM_DIR` environment variables set to their own installations. (See [Using Environment Variables](#) for information.)
- If you are moving a server that supports mirrors, you must ensure that your mirror definitions are updated. If you are not using an alias for your server name, you must disable your mirrors before moving your server. See the section [Moving Servers with Mirrors](#) for details.
- If cache space is of concern, tell users to remove links from their work areas to DesignSync caches. (Users can create new cache links after you move the SyncServer.) If cache space is not of concern, team members do not need to remove caches links from their work areas. After you move the SyncServer, they can use the `refreshcache` command to re-establish cache links.

Failure to remove cache links does no harm to users' work areas, but it can cause cache space to double for a time. The file name for a cache object is created from a hash of the object's URL, which includes the server's host and port. After you move the SyncServer, new file fetches use the new URL to construct the cache file hash and create a target in the cache for the new

fetch. Old file copies stay in the cache until the last user reference is gone. The result can be two cache copies (one with the old URL and one with the new URL), causing cache space to double during the transition period.

To remove cache links, users can use either of two methods:

- If they don't want to keep unmanaged files, users can delete their work areas using the `rmfolder -recursive` command.
- If they want to keep unmanaged files in their work areas, users can use the `populate` command with the `-version` and `-force` options. For example:

```
dss> pop -ver nosuchversion -force -rec
```

If the client installation does not use the server host and port in calculating cached files' hashed names, then this step is not necessary. See the [Cached file uniqueness](#) topic for details.

Steps to Take on the Original SyncServer

You must be the owner of the server to perform this procedure.

1. Shut down the old SyncServer:

```
% stop_sync_server
```

2. Run the script `sync_mvpgdata` to export data from the server:
 - At the **What do you want to do?** prompt, choose **1) Dump data from the database into file (export)**.
 - At the **Enter database password** prompt, enter the password for your PostgreSQL database.

This script creates a database file named `syncdb_dump.<date>_<time>` in your home directory. You must supply the path to this file when the new server is ready to import data.

`sync_mvpgdata -help` explains how the program fits into the flow of moving a server.

Steps to Take on the New SyncServer

1. On the new server, create the vault directory structure:

```
% mkdir -p syncdata/<newhost>/<newport>/server_vault
```

In a default installation, the `syncdata` directory is under `../$SYNC_DIR`.

2. Change directory to `server_vault` on the original server.
3. Copy the vault data from the original server to the new server. The method for copying depends on your platform. If the vault contains files smaller than 2 GB, use:

```
o % find . | cpio -pduL {new-server-vault-dir}
```

On Linux, `cpio` does not handle files larger than 2 Gb. If the vault contains files larger than 2 Gb, use:

```
% tar chf - . | (cd {target-directory}; tar xf -
)
```

However, `tar` has a limitation for the maximum path name and leaf name length. If the vault contains long paths, use `gtar` instead of `tar`, with the same command line options as shown for `tar`.

4. Change directory to `$SYNC_CUSTOM_DIR/servers/host/port` on the original server.
5. Copy the custom files from the original server to the new server.

```
% find . | cpio -pduL {new-server-port-dir}
```

Important: If the owner of the server is changing, you should adjust the permissions on the `PortRegistry.reg` file, which stores the mirror definitions and the server customizations, to make it readable by the new owner before you copy the files. Otherwise `cpio` may fail to copy the file to the new location and the information stored in the file would be unavailable to the server.

The `database_log` files are also tied to the specific username. If you want to preserve the logs as part of the server move, you should make them readable by the new owner.

6. If the original SyncServer has site-wide custom files that you want to install in the new server's installation, copy those files to the new server's custom site area. For example, you may want to copy `AccessControl` files, HTML templates, or other files that you have customized.
7. If the original SyncServer had site-wide registry customizations, append the specific registry customizations to the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file.
8. Create the directory:

```
syncdata/<host>}/<port>/server_metadata/postgresql
```

This empty directory is needed to validate the existence of the tags database when you run `sync_setup`.

9. If the owner of the server is changing, edit `SYNC_CUSTOM_DIR/servers/<host>/<port>/sync_server.cfg` and change the values of the variable `SYNC_APACHE_MGR`.

Note: You may also want to change the values of `SYNC_SERVER_PORT`, `SYNC_APACHE_ADMIN`, and `SYNC_APACHE_SERVER` as provide correct defaults when running the `sync_setup` procedure.

10. Run the client installation to unpack the files, install the client, and the server files and directory structure.
11. Use `sync_setup` to configure the server. For more information on running `sync_setup`, see [Configuring the SyncServer](#).

Do not accept the default values for the `server_vault` and `server_metadata`, which default to the settings for the original server. (If the server owner did not change, in which case `cpio` copied the old server's `PortRegistry.reg` file.) Specify the `server_vault` and `server_metadata` paths for the new server. You also may need a new license file.

12. Run the `sync_mvpgdata` script to import the database that you exported from the original server. Specify the path to the database file `syncdb_dump.<date>_<time>` that was created by the original server.
13. Start the new SyncServer:

```
% start_sync_server
```

After You Move the SyncServer

1. From the DesignSync Web UI, select **Email Administrator** from the **Admin** menu.
 - o At the Server Identification panel, change the Server Identification to the new host and port location.
 - o Configure email interface parameters.

For information, see [Configuring Email Notifications](#).

2. If they did not remove links from their work areas to DesignSync caches, tell users to use the **refreshcache** command to refresh their work areas. (For more information, see the `refreshcache` command.)

If the client installation does not use the server host and port in calculating

cached files' hashed names, then users should not run the **refreshcache** command, as that would recreate cache links, which is not necessary. See the Default cache (WebObjectCache) topic for details.

Notes:

- If you are not using aliases, moving a SyncServer from one host and port location to another breaks hierarchical references to or from modules residing at the old SyncServer host:port location. After you move a SyncServer, notify project leaders to remove each hierarchical reference specifying the old SyncServer host:port and add a new hierarchical reference for the new host:port.
- If the move of a SyncServer changed its host identifier and the SyncServer was not associated with a DNS alias, or if the move changed the SyncServer's port number, each work area associated with the old SyncServer `<host>:<port>` location must be changed to associate with the new SyncServer host:port. Users should use the `setvault` command to associate their work areas with the new SyncServer `<host>:<port>`.
- To make vault moves transparent to users, use aliases for SyncServers.
- Ideally, the original SyncServer has an alias, such as `dessyncserver.mycompany.com`, associated with it. Using an alias makes the SyncServer move transparent to DesignSync clients, who would have set their vault to the alias, and not a particular SyncServer `<host>:<port>`. If no alias is set up, then DesignSync clients must set vault to the new location (`setvault -rec sync://<newhost>:<newport>`) in their working directories for each Project whose SyncServer has changed.

Moving Servers with Mirrors

If you are moving a SyncServer that supports mirrors, you must use a different procedure depending on whether the server is a Repository Server (RS) or a Mirror Administration Server (MAS). In either case, moving the server is simpler if you use an alias for your server name.

See Mirroring Overview for information on DesignSync mirrors.

Moving an RS

If you are using a server alias for your Repository Server name, then you can move your RS and the associated mirrors will update automatically.

If you are not using an alias, you must update your mirrors manually:

1. On the **DesignSync** menu, click **Show RS Status** to go to the Mirror Status panel. Using the table on this panel, make a note of all the mirrors registered on your RS. (See [Displaying Mirror Status](#) for details.)
2. Disable all of the mirrors registered on the RS.
3. Move the server.
4. Edit each mirror and change the vault location to point to the new RS.
5. Re-enable all of the mirrors.

Moving an MAS

If you are using a server alias for your Mirror Administration Server name, then you can move your MAS and the associated mirrors will update automatically.

If you are not using an alias, you must update your mirrors manually:

1. Disable all of the mirrors on your MAS.
2. Move the server.
3. If your MAS includes primary mirrors, update the secondary mirrors to point to the new location of the primary mirrors.
4. Re-enable all of the mirrors on your MAS.

Related Topics

[Moving Vaults](#)

Reducing the Size of the Transaction Log

DesignSync generates a server transaction log to keep track of recent revision control operations. By default, this log retains records of all revision control operations performed in the past seven days. DesignSync uses this log to improve the response time for searches of recent revision-control activities. This log file is:

```
$SYNC_DIR/../../syncdata/<host>/<port>/server_vault/Partitions/Default/tlog.db
```

If your company performs many revision-control operations, the `tlog.db` file can become quite large. You can reduce the size of the log file by limiting the number of days for which records are retained. For details, see [Size of the transaction log \(TransactionLogRetainRecords\)](#).

Related Topics

[Size of the transaction log \(TransactionLogRetainRecords\)](#)

Setting Up Enterprise Design User Mappings

The Enterprise Design system requires the administrator to create a mapping between the usernames that exist on the Enterprise Application servers and the DesignSync servers. The Enterprise Design system offers several different ways to perform the mapping:

Username to Username - where the usernames on both systems are the same.

Email to Username - where the email address is used as the username on one of the two systems and usernames(with or without email) are used on the other system.

Email to Email - where the email addresses are the same, but the usernames may or may not be different

Custom - where your system requirements are not met by any of the provided options, you can design a custom function that allows you to specifically or individually map usernames/email addresses across the systems.

The type of mapping is set using a DesignSync Registry key as described in Enterprise User Identify Mapping (enterpriseUserMapType).

Setting up User Name Mapping

This section explains how to configure user name mapping. You may either configure the mapping from DesignSync or from the Enterprise Application server.

Configuring User Name Mapping From DesignSync

1. Using the Enterprise User Identify Mapping (enterpriseUserMapType).registry key, select the type of mapping being performed.
2. If you are using a custom type of mapping, create and load the TCL process onto the DesignSync server.
TIP: For optimal processing, the TCL process should be autoloaded by the server during server startup.

Configuring User Name Mapping from the Enterprise Application Server

On the Enterprise Application server, you will need to define the appropriate JPO function:

- For user name mapping of data going from the DesignSync server to the Enterprise Application server, use the JPO function:

```
'String  
emxSCCEDA:mapUserNameFromDesignSync (Context  
context, String userName)'
```

- For user name mapping of data going from the Application Server to the DesignSync server use the JPO function::
JPO function `HashMap<String, String>
emxSCCEDA:mapUserNameEmailToDesignSync(Context
context, HashMap<String, String>users)`.

Enterprise Design Synchronization Queue

DesignSync maintains synchronization between DesignSync modules and related Enterprise Design Objects. The data being synchronized can be automatically pushed to the Enterprise server or manually synchronized.

Before synchronization occurs, the system maintains a list of transactions queued for processing on the Enterprise Server. This list is displayed on the Enterprise Design Synchronization Queue in the DesignSync WebUI. Transactions in the queue are listed in the order in which they were created, which is also the order in which they will be processed.

If you are performing manual synchronization, or require certain transactions to be processed sooner than the automatic synchronization, or want to synchronize in a particular order, you can select transactions and launch synchronization of those transactions directly from this page.

When synchronization is automatically pushed to the Enterprise server, DesignSync uses the Alarm Trigger Maintenance Timeout setting to determine how often the data is pushed to the Enterprise server.

Viewing the Enterprise Design Synchronization Queue

From the DesignSync Web UI **Admin Menu**, select **Server | Enterprise Queue**. This displays a table containing the list of all pending transactions. If there are no pending transactions, the list is empty. The table is not sortable. It will always display the list of transaction in order of processing.

Beneath the table is a list of operations.

Note: To select all transactions, click the CheckBox in the header row.

Enable

Places the selected transactions in an active state which allows them to be synchronized when the next synchronization action occurs.

Disable

Places the selected transactions in an inactive state so they will not be synchronized when the next push occurs. Transaction are automatically disabled, by default, after 10 unsuccessful attempts to process the transaction.

Delete

Removes the selected transaction from the queue so it will never be synchronized. This action cannot be undone.

Synchronize

Synchronizes all selected, enabled transactions, clearing them from the queue.

Related Topics

Enterprise Servers

Site Options

Alarm Trigger Frequency (MaintenanceTimeout)

Enterprise DesignSync Administration User's Guide: Enterprise Design Synchronization

Privacy Policy and GDPR Compliance

Privacy Policy

In compliance with the European Union General Data Protection Regulation, all users have the right to easily access the privacy policy information for any website that stores personal information. The DesignSync web interface provides an easy interface for the administrator to publish and maintain the privacy policy and for users to view and accept the privacy policy.

When the privacy policy is initially created or updated, the users are prompted to review and accept the privacy policy immediately or, if they are not logged in, during their next login. The user can also review the existing privacy policy at any time on their user information page.

The privacy policy must be created in the form of a PDF file. You may create and format the file using any editor and then generate a PDF file. The PDF file can contain any formatting desired and will be viewed with the built-in browser PDF viewer unless the user has defined an alternate default viewer.

Tip: If you wish to track policy changes, revisions, or updates, use DesignSync to store the policy file source. When you upload the policy PDF file, if it doesn't correspond with the date of the last source checkin, you can do a force checkin to indicate the privacy policy has changed. This allows you a comprehensive picture of the privacy policy at any given time.

Creating a Privacy Policy

After the privacy policy PDF has been created, use the DesignSync Web interface to upload the PDF to the server.

Note: You may only serve one privacy policy per server.

Uploading the privacy policy

1. Click **Privacy Policy** in the **Admin Menu** of the DesignSync Web UI.
2. Click the **Browse** button to select the PDF file containing the privacy policy. Select the PDF file and click OK to accept.
3. Click **Submit** to register the privacy policy on the server. After the privacy policy has been submitted, the users are prompted to view and accept the privacy policy. You do not need to restart the server.

Updating a Privacy Policy

The administrator may update the privacy policy, if one has already been created. This removes the existing policy and replaces it with the updated policy.

After the privacy policy PDF has been created, use the DesignSync Web interface to upload the PDF to the server.

Note: Each server only has a single privacy policy in effect at any time and it applies to all users of that server.

Uploading the privacy policy

1. Click **Privacy Policy** in the **Admin Menu** of the DesignSync Web UI.
2. Select the **Update New Privacy Policy** radio button.
3. Click the **Browse** button to select the PDF file containing the privacy policy. Select the PDF file and click OK to accept.
4. Click **Submit** to register the privacy policy on the server. After the privacy policy has been submitted, the users are prompted to view and accept the privacy policy. You do not need to restart the server.

Removing a Privacy Policy

The administrator may remove a policy.

Removing the privacy policy

1. Click **Privacy Policy** in the **Admin Menu** of the DesignSync Web UI.
2. Select the **Remove Current Privacy Policy** radio button.
3. Click Submit to remove the policy. There is no confirmation dialog.

Viewing a Privacy Policy

Users and administrators may view the privacy policy at any time.

Users View the Privacy Policy by selecting **Edit User Profile** from the Me Menu, and clicking the View Privacy Policy link.

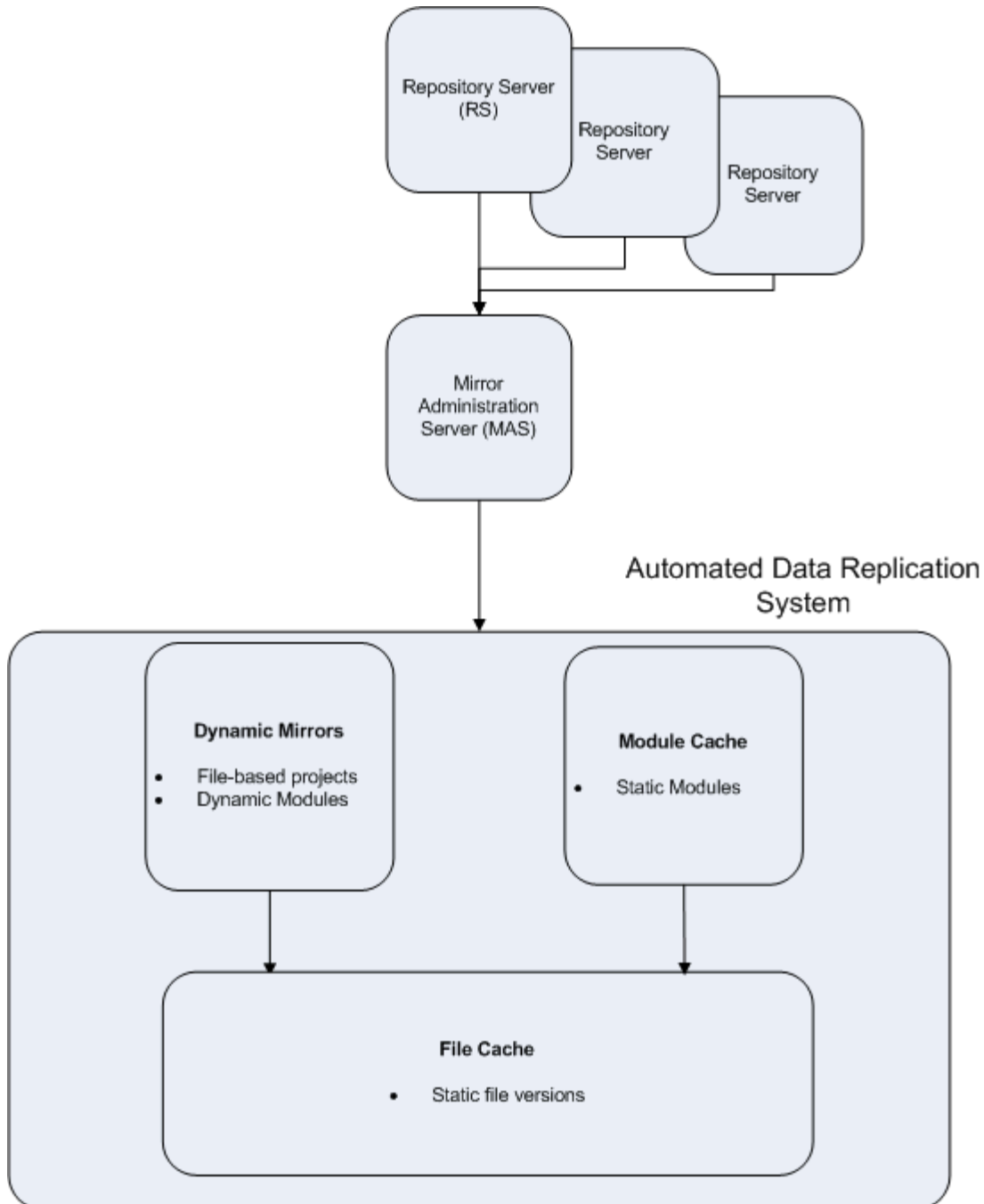
Administrators can view the privacy policy either from **Privacy Policy** in the **Admin Menu** of the DesignSync Web UI. or from **Edit**, in the **Admin Menu | User Profiles**.

Data Replication

Introduction to Data Replication

Members of a project team often need read-only access to a common set of data, such as common core components or development tools or scripts. DesignSync users on UNIX all have access to the same data, with that data maintained in a common area by DesignSync. If users each fetch local copies of the same data into their own work areas, your project team will require additional disk space per user and additional time for the data to be transferred from the server. Instead, your team can share common data in either a enterprise-wide or site-local server.

DesignSync solves this problem by providing a repository containing non-editable copies of the files that users can view in their workspaces. This repository consists of data replication components in a layered structure.



When data is replicated, DesignSync creates a mirror on the mirror administration server to manage the dynamic content. The dynamic content is also fetched into the file cache. File-based vault based users can connect to either the mirrored content when they fetch in mirror mode, or to the cached content when they fetch in share mode. Module-based data users can connect to the data in the file cache.

When static module data is replicated, DesignSync creates an mcache which, in turn, stores the individual module members into a file cache to take advantage of the performance optimization of file caching as well as the resource savings of mcaching. Module-based data users of static modules should connect to the data in the mcache.

Benefits of Using Data Replication

Historically, the first user to use the cache bore the performance cost of waiting while the cache was populated. With the data replication system, the system itself populates the cache, resulting in no performance hit to the initial user.

Additionally, for each mirror, it creates the appropriate caches. Users using static modules can link to the mcache reaping the benefits in the system resource savings that mcaches afford. The mcache itself uses a file cache and so reaps the benefits of the performance savings of file caches, ultimately offering both speed and optimal system resource utilization to the user.

Using data replication, an entire module hierarchy can be replicated and automatically updated to remain current for both dynamic and static module data.

The differences between mirrored and cached data and how to determine which usage is appropriate for your user base is discussed in *Mirrors Versus Caches*.

Understanding Data Replication

Working within a data replication root (DRR), the administrator creates a data replication for the file-based vault, module, or module hierarchy being replicated and the system maintains the replication automatically. When the data is updated, the replication is updated appropriately as well. The administrator does not need to explicitly manage the underlying mirrors, caches, and mcaches separately.

Note: Administrators can implement and manage mirrors, caches, and module caches separately if desired.

File Vault-Based Replication

When you replicate a file-based vault, you create both a static (cache) version and a dynamic (mirror) version.

- The file or folder contents of the replication are fetched to the local LAN in a mirror. This mirror is explicitly named when the replication is created.
- The file or folder contents are fetched recursively into a file cache. Users can link directly to these static file caches. The caches are automatically registered, so populating (or checking out) the sync URL of the vault in "Links to cache" mode uses the cache appropriately.

- Any changes made to the source are dynamically pushed to the mirror so users using the mirrored version will see changes immediately and file cache users will see the changes when they perform another populate (or checkout) on the workspace.

Note: Linking to the mirror shows all the files that have been changed immediately. To see any new files or remove any retired files, you will need to re-populate the workspace.

Module-Based Replication

When you replicate a module, you replicate the entire data hierarchy recursively from the replicated module. The system creates the mirrors and mcaches needed to support the replication automatically and maintains them.

- The entire module and submodule content of the module is fetched to the local LAN
 - The dynamic modules and submodules are fetched into the dynamic data area. Because mirrors have limited support for modules, the dynamic data area cannot be used by the user, however they are fetched into a file cache which is available to the user.
 - The static submodules are fetched and placed into a static mcache, as module instances. Users should have the mcache path (<DRR>/module_cache) set in the "Module Cache Paths" field for the Populate command. For more information on using module caches, see the *ENOVIA Synchronicity DesignSync Data Manager User's Guide: Populating Your Work Area*
- Note:** The mcache fetches the module contents into the file_cache area to improve mcache performance, but users linking directly to the files in the file cache lose the benefits of system resource handling that mcaching provides.
- Any changes made to the source are dynamically pushed to the data replication area so file cache and mcache users will see the changes when they perform another populate (or checkout) on the workspace.

Related topics

Mirrors Versus Caches

Fetching Files from the Mirror or Cache

Setting up a Module Cache

DesignSync Help: Using a Module Cache

Mirrors Versus Caches

The data replication system uses both mirrors and caches to provide a complete replication solution, but it can be difficult to determine which type of replication you should use. Mirrors and caches provide two related but different modes for sharing a workspace. Both mirrors and caches let users access shared files instead of requiring each user to keep local copies. There are, however, several differences in how these two workspace sharing modes work.

Mirrors...	Caches...
Contain only one version of each object (file)	Can contain multiple versions of each object (file).
Are automatically updated to contain the data set specified by the project lead who defined the mirror. For example, the latest versions of files on the main Trunk branch. Or file versions tagged bugfix on the Alpha branch.	Are updated with specified or new versions of files only when a user uses the -share option with the checkout (co), checkin (ci),cancel (cancel), or populate commands (or chooses Keep a link to cache (share) when performing these options.
Have links that are dynamic. When a mirror is updated with a new file version, the link to that mirrored file automatically points to the resolved version.	Have links that are static. A link from your work area to a particular version in the cache does not change until you request an update using the checkout, cancel, or populate operation, or a new version is created with the check-in command.
Usually take less disk space because only one version of a file resides in the mirror directory.	Can potentially require more disk space because multiple versions of files can exist in the cache. Note: Versions in a cache that are no longer linked to by a user are automatically purged from the cache.
Uses symbolic links to files on the mirror. Each symbolic link may require disk space and an inode.	Uses symbolic links or hard links. Each symbolic link created to a file may require disk space and an inode. Hard links may use less disk and use only a single inode per cache file regardless of how many links are created to the file.

Note: Mirrors and caches accomplish sharing of data through use of UNIX links, which limits their use to UNIX platforms.

Related Topics

[Mirroring Overview](#)

How DesignSync Manages Caches

Data Replication System

Setting Up Data Replication

This topic provides the steps necessary to set up data replication from initial system configuration, to setting up a data replication root, to adding replicated data. Data replication provides an easy way to replicate data using the full spectrum of available DesignSync data replication tools: mirroring, file caching, and module caching. The data replication system, once setup, will automatically maintain the data for you, reducing the administrative burden of mirrors, while providing the benefits of using replicated data. For more information understanding the data replication system, see [Introduction to Data Replication](#).

Data Replication supports HTTP Proxy header filtering. For more information, see [Working with HTTP Proxy Header Information](#).

Prerequisites for the Data Replication System

In order to use data replication, you must enable one or more servers as a Mirror Administration Server (MAS). This MAS hosts the mirrors and manages the data replication. You can designate a server as a MAS during install time as described in [Setting Up a SyncServer](#) or any time after installation as described in [Setting Up a Mirror Server: Enabling a Mirror Administration Server](#).

Each server that hosts the source data that will be replicated must be designated as a Repository Server (RS). You can designate a server as an RS during install time as described in [Setting Up a SyncServer](#) or any time after installation as described in [Setting Up a Mirror Server: Enabling a Repository Server](#).

All the RSEs and MASEs that are part of your data replication system must have the same username/password defined for use by the data replication system. This common username/password combination is used to authenticate the channel for RS<->MAS communication. This username/password must also be registered as the default MAS username. For information on setting the default MAS username, see [Setting up a Mirror Server](#).

To provide automated cleaning of the file cache within the data replication, reference counting must be on. By default, reference counting is enabled. For information on enabling reference counting, see the [Enable cache optimizations on the SyncAdmin Performance Options](#).

Any user who is creating or managing data replications must have the access rights to use Data Replication and access to any specific Data Replication commands desired.

For more information setting up access permissions for Data Replication, see [Access Control for Replication](#).

Setting Up the Data Replication Root

The data replication root (DRR) is the central maintenance point for management of data replication. From an administration standpoint, all maintenance on the data replications is done through the DRR. From a physical standpoint, the DRR is composed of three directories which support the three types of data replication supported by the system.

- `dynamic` - this directory supports the mirrors for the data replications.
- `module_cache` - this directory supports the mcaches for the data replications, when the replicated data is a module.
- `file_cache` - this directory supports the LAN or file cache for the data replications.

These physical directories are located within the directory path specified as the root path when the DRR is created. For information on creating the DRR, see [Add Root Settings](#).

Setting up a Data Replication

The data replication consists of the mirror, and the cache(es) that make up the complete replication. For more information on how data replication works, see [Introduction to Data Replication](#). For information on creating a data replication, see [Add Data to Replicate](#).

Related Topics

[Cleaning and Removing Data Replication](#)

Cleaning and Removing Data Replication

DesignSync provides maintenance for active data replication roots by automatically maintaining the file cache and mirror scrubbing to remove outdated references and unused objects, but you also have the ability to manually scrub the mirror or cache or remove a data replication from the system entirely.

Cleaning Data Replications

The data replications are automatically cleaned using a set of registry keys to determine how long a replication should go untouched before scrubbing and at what time the scrubber should run. The primary task of the scrubber is to remove any old, no-longer referenced submodules in both the `dynamic` directory and the `module_cache` directory. This prevents the system from wasting resource maintaining copies of data that is no

longer used. Removed dynamic submodules are also scrubbed from the file_cache directory automatically when reference counting is enabled. For information on file cache scrubbing, see Cleaning File (LAN) Caches.

By default, the scrubber runs nightly at 1:00 AM and cleans any replications that have not been touched in 7 days. For information on modifying these settings, see Replicate Scrub Time (ReplicateScrubTime).

Cleaning Mcaches

Data in the mcache directory controlled by data replication is not automatically cleaned. The administrator should implement mcache scrubbing for the data replication directory using the mcache scrub utility. For more information on implementing an mcache scrub strategy, see Cleaning a Module Cache.

Note: The mcache scrub command works by examining a set of user workspace paths looking for module cache links to module instances within a list of provided module cache paths. In order to work on the replicated data, the path to the module_cache on the DRR must be included on workspace path list.

Cleaning File (LAN) Caches

When reference counting is enabled, the file cache is entirely self-cleaning. When the reference count for an object (file) reaches 1 (no links), the object is removed from the file cache. For information on enabling reference counting, see the Enable cache optimizations on the SyncAdmin Performance Options.

If reference counting is not enabled, the administrator will need to perform regular file cache maintenance on the DRR file_cache directory. For more information on scrubbing the file cache, see Cleaning a Cache.

Removing a Data Replication

When a data replication is no longer needed you can either disable or remove the replication from the system.

To disable or delete a data replication:

1. Launch the DesignSync web interface to your MAS, if it is not already running.
2. Navigate to the Replicate section and select Show Roots.
3. Click on the name of the DRR hosting the data replication.
4. Click the checkbox next to the name of the data replication to disable or delete.
5. Select the appropriate action button. There is no confirmation screen for either action.

For more information in removing or disabling data replication, see Show Replicated Data Hierarchies.

Note: When you remove a replication from the system, it is removed entirely from the system to free up disk spaces and system resources, so this operation cannot be undone. You can recreate the data replication.

Related Topics

ENOVIA Synchronicity Command Reference: mcache scrub

ENOVIA Synchronicity Command Reference: cachescrubber

General Replicate Settings

The General Replicate Settings panel lets you set the properties of your data replication server. The data replication server must be configured as a MAS server.

To access the General Mirror Settings panel, click **General Settings** in the **Replicate** section of the **DesignSync** menu. Click **Submit** when you have made your changes. Click **Reset** to remove your changes from the panel.

You also can work with data replication using the command-line interface. See the `replicate` command descriptions in the ENOVIA Synchronicity Command Reference for details.

Click on the image for more information about each field.

The screenshot shows a web interface for 'Data Replication' with a sub-section for 'General Replicate Settings'. The interface includes a 'Default User' section with input fields for 'Username' (containing 'rsmith') and 'Password' (masked with 10 dots). Below this is a 'Disable Reference Counting' section with an unchecked checkbox and the text 'Disable reference counting in the file cache'. At the bottom of the form are three buttons: 'Reset', 'Submit', and 'Help'.

Default user

Specify the default username and password for the replication host. The default user must be added as a user profile on the RS and MAS.

Note: The default user for data replication can be set up independently of the default user for mirrors. If no default username and password is set up for data replication, then the data replication will use the default username and password that is set up for mirrors. For more information on setting up the default username and password for mirrors, see [Setting up the Mirror Server](#).

Disable Reference Counting

Specifies whether reference counting is enabled or disabled for the system. By default, reference counting is enabled. Enabling reference counts provide automatic reference maintenance.

If you disable reference counting on your system, it should also be disabled site-wide by disabling the SyncAdmin Performance Optimization, "Enable Cache optimizations." These values should always be the same to ensure data integrity within file cache system.

Note: This option is not retroactive. Any existing data continues to have the reference count setting that was in use when the data was fetched. Any subsequently fetched data uses the newly set mode.

Add Root Settings

The Add Root panel lets you create a new data replication root (DRR) on a MAS. Once the DRR has been created on the MAS, you can add data replications to the DRR.

To access the Add Root panel, click **Add Root** in the **Replication** section of the **DesignSync** menu on the MAS server. Click **Submit** when you have made your changes. Click **Reset** to remove your changes from the panel.

Click on the image for more information about each field.

Data Replication | Add Root

Name	<input type="text" value="ChipNZ214"/>
Root Path	<input type="text" value="/home/developments/DesignSync/DRR"/>
Read Mode	<input type="text" value="All"/>

Name

Logical name of the DRR. This name must be unique with respect to all other DRRs defined on the MAS.

Note: For optimal function, you should avoid using any reserved characters in the DRR name.

Root Path

Specifies the path to the DRR being added. If the path does not exist, and is creatable, the command creates it.

Note: The path cannot contain a module cache or dynamic folder, which is a sub-directory of a DRR. It can contain an existing file cache.

Read Mode

The read permissions set on the DRR directory. This option is valid only when SUID mode is enabled.

- All sets the read permission on the DRR directory to be readable by all users, the primary group of the MAS owner, and the MAS owner. (Default)
- Group only sets the read permission on the DRR directory to be readable by the primary group of the MAS owner, and the MAS owner.

Note: If SUID is not enabled, and the **Require SUID** option is not enabled as described in Setting Up a Mirror Server, the system will enable read and write modes for all. If the "**Require SUID**" option is enabled, the command will fail.

Submit

Sends the form to the server for processing.

Reset

Clears the form.

Help

Launches this page.

Add Data to Replicate

The Add Data panel adds the desired data to the specified data replication root (DRR).

To access the Add Data panel, click **Add Data** in the **Replicate** section of the **DesignSync** menu. This menu item is available only after you have set up a Mirror Administration Server (MAS) on the server hosting the DRR and then restarted the server. Click **Submit** when you have made your changes. Click **Reset** to remove your changes from the panel.

You also can create a data replication from the command-line interface using the `replicate data` command. See ENOVIA Synchronicity Command Reference: `replicate data` Command for details.

Replicating Module Data

When you add a module to DRR, the replicate system automatically registers the required mirror setting with the repository server (RS) hosting the top-level module and all RSeS hosting referenced submodules.

Note: The RS must support scripted mirrors, introduced in V6R2012.

After the scripted mirrors have been registered with the MAS, the command creates a module instance corresponding to the module specified with the appropriate selector. This module instance will be created in the 'dynamic' folder.

Replicating Files-Based Data

For files-based vault data, the `replicate data` command calls the `mirror create` command in order to create the data replication mirror.

Click on the image for more information about each field.

Data Replication

Add Data

Replication Root	<input type="text" value="ChipNZ214"/>
Vault To Replicate	<input type="text" value="sync://ABCo.com/Projects/Chip"/>
Selector	<input type="text" value="Trunk:Latest"/>
Name	<input type="text" value="required for file-based"/>

Replication Root

Select the logical name of the DRR from the list of the Replication Roots on the MAS server.

Vault to Replication

Enter the URL of the module or files-based project.

Selector

Enter the selector, or selector list. If no selector is specified, the command will use the default selector, 'Trunk:'. To specify multiple selectors, separate the selectors with a comma.

Name

A user friendly name for a file-based project. This option is not applicable to modules data, which is automatically named with a unique name based on the module instance name.

Submit

Sends the form to the server for processing.

Reset

Clears the form.

Help

Launches this page.

Show Roots

This Show Roots panel lists the data replication roots (DRRs) and their properties registered on the MAS.

To access the **Show Roots** panel, click **Show Roots** in the **Replicate** section of the **DesignSync** menu on the Web UI. This menu item is available only after you have set up a Mirror Administration Server (MAS) on the server hosting the DRR and then restarted the server.

Click on the image for more information about each field.

Data Replication												
Show Roots												
<input type="checkbox"/>	Name	Root Path	Read Mode	Disk Usage								
<input type="checkbox"/>	ChipDev	/home/ReplicationHome/repdata	All	<table border="1"> <thead> <tr> <th>Filesystem</th> <th>Size</th> <th>Used</th> <th>Avail</th> </tr> </thead> <tbody> <tr> <td>flood:/vol/home/tmarci2</td> <td>3.3T</td> <td>1.4T</td> <td>1.8T</td> </tr> </tbody> </table>	Filesystem	Size	Used	Avail	flood:/vol/home/tmarci2	3.3T	1.4T	1.8T
Filesystem	Size	Used	Avail									
flood:/vol/home/tmarci2	3.3T	1.4T	1.8T									
<input type="checkbox"/>	ChipNZ214	/home/developments/DesignSync/DRR	Group	<table border="1"> <thead> <tr> <th>Filesystem</th> <th>Size</th> <th>Used</th> <th>Avail</th> </tr> </thead> <tbody> <tr> <td>flood:/vol/home/tmarci2</td> <td>3.3T</td> <td>1.4T</td> <td>1.8T</td> </tr> </tbody> </table>	Filesystem	Size	Used	Avail	flood:/vol/home/tmarci2	3.3T	1.4T	1.8T
Filesystem	Size	Used	Avail									
flood:/vol/home/tmarci2	3.3T	1.4T	1.8T									
<input type="checkbox"/>	NewDev	/home/development2/DesignSync/DRR	Group	<table border="1"> <thead> <tr> <th>Filesystem</th> <th>Size</th> <th>Used</th> <th>Avail</th> </tr> </thead> <tbody> <tr> <td>flood:/vol/home/tmarci2</td> <td>3.3T</td> <td>1.4T</td> <td>1.8T</td> </tr> </tbody> </table>	Filesystem	Size	Used	Avail	flood:/vol/home/tmarci2	3.3T	1.4T	1.8T
Filesystem	Size	Used	Avail									
flood:/vol/home/tmarci2	3.3T	1.4T	1.8T									

Name

Logical name of the DRR. This link allows you to click through to see what data is replicated by the DRR. For more information, see [Show Replicated Data Hierarchies](#).

Root Path

Root path to the DRR. This link allows you to click through to see how much disk space the data replication is using. For more information, see [Show Disk Usage for Root](#).

Read Mode

Readmode specified when the DRR was created. The possible options are 'all' and 'group.' For more information see [Add Root Settings: Read Mode](#).

Disk Usage

Disk usage information including the name of the partition containing the root, the size of the partition, how much of the partition is used and how much is available for use.

Reset

Resets the panel, clearing any selections that you have made.

Delete

Removes the data replication root. This removes the replication and deletes all data within the replication. This operation cannot be undone. The DRR can be recreated with the same name.

Help

Launches this page.

Show Replicated Data Hierarchies

The Show Data panel lists the data replications and their properties defined on the DRR and allows you to enable, disable, or delete the replications.

To access the **Replicated Data Hierarchies** panel, click **Show Roots** in the **Replication** section of the **DesignSync** menu on the MAS server. From the **Show Roots** page, click the name of the DRR to open the **Replicated Data Hierarchies** page that lists the replications on that DRR.

To perform an operation on the replicated data click the checkbox next to Name and select the appropriate operation. The operations are described after the data replication properties.

Click on the image for more information about each field.

Root Name: ChipDev
Root Dynamic Path: /home/replication/dynamic

<input type="checkbox"/>	Name	Enabled	Status	Vault URL	Selector	Base Directory
<input type="checkbox"/>	CPU%0	✓	●	sync://data.ABCo.com:2647/Modules/Components/CPU	Trunk:	9b/1e/9b1e794175a7c24bd5a329b3a5bd8d7a/CPU/Trunk/basedir
<input type="checkbox"/>	Chip%0	✓	●	sync://data.ABCo.com:2647/Modules/ChipDesign/Chip	Trunk:	ef/ea/efea1173a39a7df9e42e3e8d821fd580/Chip/Trunk/basedir
<input type="checkbox"/>	ROM%0	✗	●	sync://data.ABCo.com:2647/Modules/Components/ROM	Trunk:	ee/f5/ee/f5d1b4b3eab3c9ec3f95b82b1b90dc/ROM/Trunk/basedir

Enable Disable Delete Reset Help

Data Replication Properties

Root Name

Logical name of the DRR on which the data replications are hosted.

Root Dynamic Path

Path to the "dynamic" directory within the DRR.

Name

Logical name of the data replication.

Enabled

Shows whether the data replication is enabled or disabled. Enabled is indicated by a checkmark. Disabled is indicated by an X. To change the status of the data replications, select the desired replication and click **Enable** or **Disable**.

Status

Status of the data replication indicating whether the replication is functioning normally, or there are errors. Normal function is indicated by a green dot. Errors are indicated by a red dot. If you see a red dot, the administrator click the red dot to view the error log to determine and fix the error.

Vault URL

URL of the source data.

Selector

Selector of the source data.

Base Directory

Location of the replicated data within the dynamic folder.

Data Replication Actions

Enable

Turns the data replication on.

Disable

Turns the data replication off.

Delete

Removes the data replication. This deletes all data within the replication. This operation cannot be undone. The replication can be recreated by readding it. For more information on creating a data replication, see Add Data to Replicate.

Reset

Manually forces an update on the data in the data replication.

Help

Launches this page.

Related Topics

ENOVIA Synchronicity DesignSync Command Reference: replicate enable

ENOVIA Synchronicity DesignSync Command Reference: replicate disable

ENOVIA Synchronicity DesignSync Command Reference: replicate rmdata

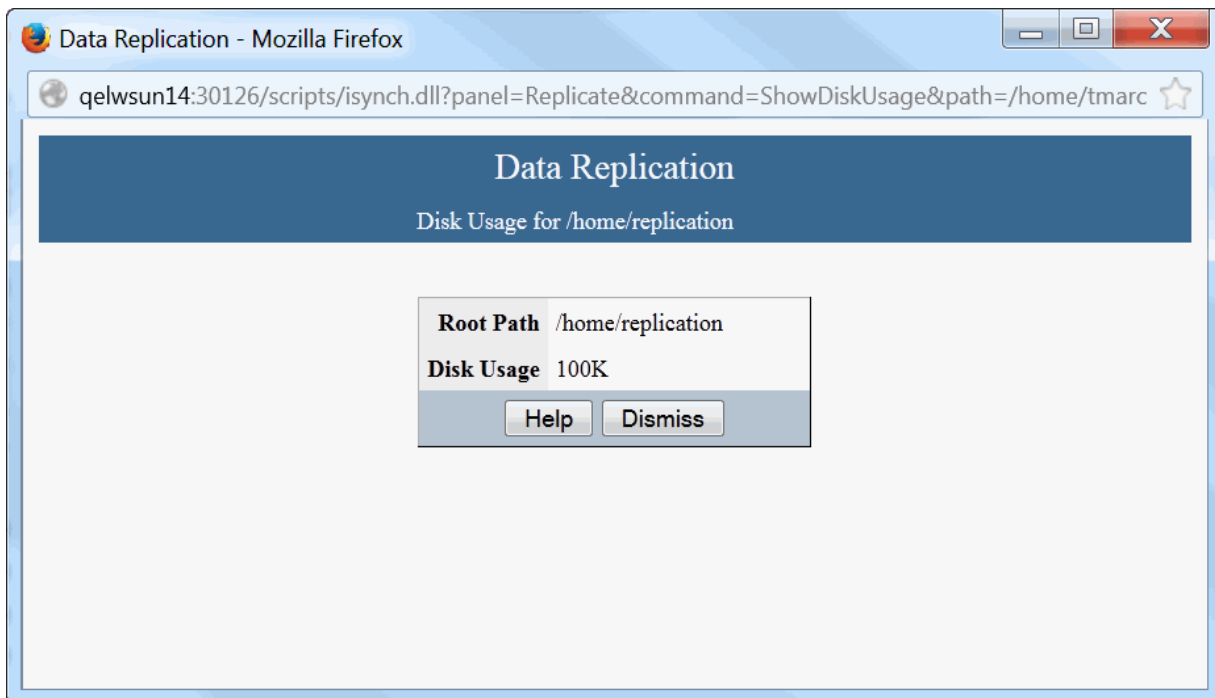
ENOVIA Synchronicity DesignSync Command Reference: replicate reset

Show Disk Usage for Root

The Show Disk Usage for Root panel shows the root directory for the DRR and the amount of space on disk the DRR is using.

Note: There may be a delay displaying this information as the system calculates the disk usage.

Click on the image for more information about each field.



Root Path

Path to the DRR directory.

Disk Usage

Total disk usage of the DRR.

Help

Launches this page.

Dismiss

Closes this page.

File Caches

How DesignSync Manages Caches

As a DesignSync administrator, you can set up a file cache so that your team can access shared copies of object versions. By doing so, you save disk space on the LAN, as well as the time for individual fetches of updated files. Another option for managing design data on a LAN is to set up a mirror. See [Mirrors Versus Caches](#) for a comparison of the two methods.

Note: Using the data replication system takes advantage of the functionality of file caches and mirrors while reducing the administrative overhead required to maintain them. For more information, see [Introduction to Data Replication](#).

DesignSync provides two methods for creating cache links:

- hard links - hard links can be used when the workspace and cache share the same file partition and mount point. Using hard links, rather than symbolic links saves inode resources because only a single inode is required for each file in the cache. It may also save disk space. For information on enabling or disabling hard links, see [Links Options](#).
- symbolic links - symbolic links can be used regardless of whether the cache and the workspace are located on different file partitions or at different mount points. If hard links are enabled, but cannot be used because the cache doesn't meet the hard link criteria, symbolic links are automatically created to the cache.

The following scenario illustrates how DesignSync manages the links that implement a UNIX file cache. For an introduction to DesignSync LAN caches, see [DesignSync Data Management User's Guide: What is a LAN Cache?](#).

A design team has a file cache set up by its DesignSync administrator. The cache is stored in the default location of `< SYNC_DIR>/../ sync_cache` (parallel to the `< SYNC_DIR>` installation directory). (See [Setting Up a LAN Cache](#) for the methods

DesignSync administrators can use to set up a default or project cache.) Until one of the team members performs a check-in, check-out, cancel, or populate with the `-share` option, the `sync_cache` directory remains empty. The first time a team member performs one of these operations with the `-share` option upon a managed object, DesignSync:

1. Fetches the requested file or module member version into the cache.
2. Creates a link from the user's workspace to the file or module member in the cache. Depending on the system setup, this link will be either a hard link or a symbolic link.

DesignSync tracks the objects in the cache such that when all team members relinquish their links to an object, that object is removed from the cache. To perform this tracking, with symbolic links, DesignSync uses hard links with unique names, based on a hash of the user's workspace path, to track the number of references to each file in the cache. When using hard links, the workspace hard link is used for tracking.

See the Tips for Administering LAN Caches topic, Ensure that team members use the same vault URL to prevent multiple copies of cache objects, for a description of how the filenames are constructed.

Note: Because caching maintains a readable copy of the object being cached, you may choose to not cache particularly sensitive IP. For information on removing sensitive IP from the cache, see Adding or Removing an Object from Caching. For information on understanding which objects should be included or removed from the caching functionality, see Understanding Excluding IP from the Cache.

The symbolic link reference tracking mechanism is illustrated in the following scenario.

Action 1:

The team's first user, Joe, creates a file, `top.v`, and checks in the new file with the following command line:

```
dss> ci -new -share top.v -nocomment
```

This command creates version 1.1 of `top.v` in the vault. DesignSync generates a 3-character name associated with the vault object's folder using a hashing algorithm, in this example, `sa6`. DesignSync creates a subdirectory by this name under the `sync_cache` directory. Within this subdirectory, DesignSync fetches a copy of `top.v`, version 1.1, and generates a name for this base version of the file using a hash of the vault-url, the version number, and the check-in time of the version. In this case,

```
sa6fff3241622d4c8-1.1-982002979-0.
```

At the same time, DesignSync creates a hard link reference to the base version in the `sa6` subdirectory of the cache. The hard link has the same root as the base version, but is a unique name, with a suffix of a hash of Joe's workspace path. In this case the name is

```
sa6fff3241622d4c8-1.1-982002979-b59954423b46bcec.
```

The file, `top.v`, in Joe's workspace, is a symbolic link to the corresponding hard link in the cache.

User Action/Result	What the Cache contains:
<p>Joe checks in a new file, <code>top.v</code>, with the <code>-share</code> option.</p> <p>Joe sees a symbolic link, <code>top.v</code>, in his work area to version 1.1 in the cache.</p>	<pre>/sa6 sa6fff3241622d4c8-1.1-982002979-0 base version (1.1) of top.v, fetched from the vault. sa6fff3241622d4c8-1.1-982002979- b59954423b46bcec unique hard link to base version 1.1 in cache (Joe)</pre>

Action 2:

Next, a second team member, Marie, populates from the vault folder:

```
dss> populate -rec -share
```

As in the case of Joe, DesignSync creates a hard link, in the `sa6` subdirectory of the cache, to the base version. This hard link has the same root name as the hard link created for user 1, but is a unique name, with a suffix of a hash of Marie's workspace path. In this case the name is `sa6fff3241622d4c8-1.1-982002979-f216a783154b31cc`.

The file, `top.v`, in Marie's workspace is a symbolic link to the hard link in the cache.

User Action/Result	What the Cache contains:
--------------------	--------------------------

<p>Joe's work area:</p> <p><code>top.v</code> (symbolic link to cache)</p>	<p><code>/sa6</code></p>
<p>Marie populates her work area using the - share option.</p> <p>Marie sees a symbolic link, <code>top.v</code>, in her work area to version 1.1 in the cache.</p>	<p><code>sa6fff3241622d4c8-1.1-982002979-0</code> base version (1.1) of <code>top.v</code>, fetched from the vault</p> <p><code>sa6fff3241622d4c8-1.1-982002979-b59954423b46bcec</code> unique hard link to base version 1.1 in cache (Joe)</p> <p><code>sa6fff3241622d4c8-1.1-982002979-f216a783154b31cc</code> unique hard link to base version 1.1 in cache (Marie)</p>

Action 3:

Next, a third team member, Bob, populates from the vault folder:

```
dss> populate -rec -share
```

DesignSync creates another unique hard link reference, `sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868`, in the `sa6` subdirectory of the cache. As in the case of Joe and Marie, DesignSync creates another unique hard link reference in the `sa6` subdirectory of the cache. DesignSync also creates a symbolic link named `top.v` in Bob's workspace to the unique hard link in the cache.

User Action/Result	What the Cache contains:
<p>Joe's work area:</p> <p><code>top.v</code> (symbolic link to cache)</p>	<p><code>/sa6</code></p> <p><code>sa6fff3241622d4c8-1.1-982002979-0</code> base version (1.1) of <code>top.v</code>, fetched from the vault</p> <p><code>sa6fff3241622d4c8-1.1-982002979-b59954423b46bcec</code> unique hard link to base version 1.1 in cache (Joe)</p> <p><code>sa6fff3241622d4c8-1.1-982002979-f216a783154b31cc</code></p>
<p>Marie's work area:</p> <p><code>top.v</code> (symbolic link to cache)</p>	
<p>Bob populates his work area using the - share option.</p> <p>Bob sees a symbolic link, <code>top.v</code>, in his work area to version 1.1 in the cache.</p>	

	<p>unique hard link to base version 1.1 in cache (Marie)</p> <p>sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868</p> <p>unique hard link to base version 1.1 in cache (Bob)</p>
--	---

Action 4:

Next, Marie checks out `top.v` with a lock:

```
dss> co -lock top.v -nocomment
```

This action creates an actual copy of `top.v` in Marie's workspace and removes the `top.v` symbolic link to the `sa6fff3241622d4c8-1.1-982002979-f216a783154b31cc` hard link reference. DesignSync also removes this hard link from the cache area.

User Action/Result	What the Cache contains:
Joe's work area: <code>top.v</code> (symbolic link to cache)	<p>/sa6</p> <p>sa6fff3241622d4c8-1.1-982002979-0 base version (1.1) of top.v, fetched from the vault</p> <p>sa6fff3241622d4c8-1.1-982002979-b59954423b46bcec unique hard link to base version 1.1 in cache (Joe)</p> <p>sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868 unique hard link to base version 1.1 in cache (Bob)</p>
Marie's work area: <code>top.v</code> (locked copy)	
Bob's work area: <code>top.v</code> (symbolic link to cache)	

Action 5:

After Joe finishes using the `top.v` file, he removes the file using the DesignSync `rmfile` command:

DesignSync Data Manager Administrator's Guide

```
dss> rmfile top.v
```

This action removes the `top.v` symbolic link to the hard link reference in the cache, `sa6fff3241622d4c8-1.1-982002979-b59954423b46bcec`. DesignSync also removes the hard link reference from the cache area.

Important: The user removes the file using the DesignSync `rmfile` command rather than the UNIX `rm` command. If a user removes an object using the UNIX `rm` command, the corresponding hard link reference will not be removed from the cache and then the number of DesignSync's references to the object will always be greater than the number of links to the cached objects, leading to unnecessary objects in the cache.

User Action/Result	What the Cache contains:
Joe removes the file using DesignSync's <code>rmfile</code> command. No references to the file remain in Joe's work area.	<code>/sa6</code> <code>sa6fff3241622d4c8-1.1-982002979-0</code> base version (1.1) of <code>top.v</code> , fetched from the vault <code>sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868</code> unique hard link to base version 1.1 in cache (Bob)
Marie's work area: <code>top.v</code> (locked copy)	
Bob's work area: <code>top.v</code> (symbolic link to cache)	

Action 6:

Next, Marie checks in the changes to the `top.v` file creating version 1.2 in the vault:

```
dss> ci -share top.v
```

This action creates a copy of `top.v` version 1.2 in the cache and generates a new name for the base version, in this example, `sa6fff3241622d4c8-1.2-982003200-0`. DesignSync also creates a new hard link reference in the cache to this copy, `sa6fff3241622d4c8-1.2-982003200-f216a783154b31cc`. In Marie's workspace, DesignSync creates a `top.v` symbolic link to the new hard link reference (`top.v` version 1.2) in the cache.

User Action/Result	What the Cache contains:
Joe's work area:	
Marie checks in changes to top.v 1.1 creating top.v 1.2.	
Marie sees a symbolic link, top.v, in her work area to version 1.2, in her cache.	<p>sa6/</p> <p>sa6fff3241622d4c8-1.1-982002979-0 base version (1.1) of top.v, fetched from the vault</p>
Bob's work area:	
top.v (symbolic link to version 1.1 in cache)	<p>sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868 unique hard link to base version 1.1 in cache (Bob)</p> <p>sa6fff3241622d4c8-1.2-982003200-0 base version (1.2) of top.v, fetched from vault</p> <p>sa6fff3241622d4c8-1.2-982003200-f216a783154b31cc unique hard link to base version 1.2 in cache (Marie)</p>

Action 7:

Finally, Bob populates from the vault folder:

```
dss> populate -rec -share
```

This action creates a symbolic link top.v in Bob's workspace and in version 1.2 in the cache. DesignSync creates a corresponding hard link in the cache, sa6fff3241622d4c8-1.2-982003200-5e29e3477fcc2868, that references the new 1.2 base version, sa6fff3241622d4c8-1.2-982003200-0.

Because Bob is no longer referencing top.v version 1.1 (sa6fff3241622d4c8-1.1-982002979-5e29e3477fcc2868), DesignSync removes the associated hard reference link. There are no more reference links to top.v version 1.1, so DesignSync removes the top.v version 1.1 base version (sa6fff3241622d4c8-1.1-982002979-0) from the cache.

User Action/Result	What the Cache contains:
Joe's work area:	

<p>Marie's work area:</p> <p><code>top.v</code> (symbolic link to version 1.2 in the cache)</p>	<p><code>sa6/</code></p>
<p>Bob populates his work area using the <code>-share</code> option.</p> <p>Bob sees a symbolic link, <code>top.v</code>, in his work area to version 1.2 in the cache</p>	<p><code>sa6fff3241622d4c8-1.2-982003200-0</code> base version (1.2) of top.v, fetched from the vault</p> <p><code>sa6fff3241622d4c8-1.2-982003200-f216a783154b31cc</code> unique hard link to base version 1.2 in cache (Marie)</p> <p><code>sa6fff3241622d4c8-1.2-982002979-5e29e3477fcc2868</code> unique hard link to base version 1.2 in cache (Bob)</p>

This scenario illustrates the proper use of a DesignSync cache. See Tips for Administering LAN Caches to prevent potential problems with your file cache.

Related Topics

DesignSync Data Management User's Guide: What is a LAN Cache?

Setting Up a LAN Cache

Tips for Administering LAN Caches

Setting Up a LAN Cache

The designation of a cache occurs as follows:

- During DesignSync installation, the DesignSync administrator runs the **sync_setup** script and selects the default cache. During the client-configuration stage of the installation, the script prompts for a cache directory name; the default is **<SYNC_DIR>/../sync_cache** (`sync_cache` and `<SYNC_DIR>` are parallel directories).
- Using the SyncAdmin tool's **General** tab, a LAN administrator can specify a file cache that replaces the default cache specified during installation. See SyncAdmin Help for more information.
- Using the SyncAdmin tool's **Projects** tab, a project leader can specify a project cache, which defines a cache/vault folder association. A project cache overrides the default file cache if the vault folder set on a workspace matches the project cache/vault folder association. See Project Configurations for details.
- Using the SyncAdmin tools' **Links** tab, a project leader can enable using hard links or symbolic links to the cache. See Links Options and Tips for Administering LAN Caches for more information.

- Using data replication, when a data replication root (DRR) is created, the replication root creates a `file_cache` directory to use for the caches associated with the replicated data managed by the DRR. For more information on `file_caches` in the DRR, see [Setting Up Data Replication](#).

Cache permissions must be defined such that all intended users of the cache can successfully read from and write to the cache.

Tip: Enable SUID to avoid accidental cache modification. SUID allows you to always fetch files into the cache with a designated user, usually `syncmgr`, so that the users recognize that the files come from the cache.

Note: Cache directories should not be nested -- you should not define a cache directory that is a subdirectory of another cache directory.

Related Topics

[Setting Permissions on the Cache](#)

[ENOVIA Synchronicity Administrator Tool Overview](#)

[How DesignSync Determines the Correct Project Cache](#)

Setting Permissions on the Cache

A cache directory must have UNIX permissions that grant full access to the users of the cache. Even though cached files are read-only, a user must have write access to the cache because the user's process updates the cache directory (when checking in or checking out a version that doesn't yet exist in the cache).

Important: The DesignSync recommended method for managing permissions and enforcing a read-only methodology is described in the *ENOVIA DesignSync Installation* in the Program Directory.

Setting permissions on the cache when SUID is not used

DesignSync creates cache directories during installation or during the creation of a new project. By default, DesignSync creates cache directories with wide-open permissions:

```
2777 -- set-group-ID bit set, read/write/execute privileges for owner/group/world
```

You can limit cache access by changing the permissions of the cache directory from your UNIX shell as described in the following bulleted items. Another method of limiting cache access is to create a project and limit the users of a project. See [Project Configurations](#) for details.

- Set the permissions. The valid permission settings are:

777 (wide open) -- default setting

770 (no world access) -- typical setting

755 (no write access for group or world) - safe setting preventing unauthorized write access. (**Note:** This setting is the default when SUID is enabled.)

750 (no write access for group, or read write access for world) - safe setting prevented unauthorized write access and restricting viewing access to user and group only. (**Note:** This is an acceptable setting when SUID is enabled to restrict access to only group members.)

700 (owner access only) -- not useful in a cache methodology

- Set the set-group-ID bit, which causes files and folders created in the cache to inherit the group ID of the cache directory, not the group ID of the user creating the files or folders. Users of the cache must be members of the same group, but need not have the same primary group.
- Set the group ID for the cache directory to the group whose members will be accessing the cache.

For example, if you want only the 'cae' group to access the cache called "sync_cache", do the following:

1. Change the default permissions so that the world cannot access the cache:

```
% chmod 770 sync_cache
```

2. Set the set-group-ID bit:

```
% chmod g+s sync_cache
```

3. Set the group ID to 'cae':

```
% chgrp cae sync_cache
```

4. Ensure that all users on the project belong to the 'cae' group. Because the set-group-ID bit is set, 'cae' does not need to be each user's primary group. For example, user 'kim' does not have 'cae' as her primary group, but she can access the cache because 'cae' is a secondary group:

```
% groups kim
layout cae mgr
```

Note: UNIX environments vary. For example, some versions of **chmod** do not accept an absolute mask (for example, 2770) to set the set-group-ID bit but instead require a symbolic specification (g+s). Consult your system documentation for information on setting permissions.

Additional information about permissions:

- Cached files have read access for everyone, write access for the owner only, and they may have execute permission depending on whether the file had execute permission when it was checked in.
- All folders in the cache inherit their permissions and group IDs (assuming the set-group-ID bit is set) from their parent folders, which ultimately reflect the permissions and group ID of the top-level cache directory.

Moving a Cache

As DesignSync administrator or project leader, you might find it necessary to move a file cache that is in use. For example, you might need to move the cache to a machine with more disk space or better performance. You might need to move a cache to ensure data security, as well.

To move a cache, follow these steps:

1. Create the new cache location using the SyncAdmin **General** tab.

If you want to set up a project-specific cache, use the **Projects** tab instead. See SyncAdmin Help for details.

2. Adjust cache permissions accordingly.

See Setting Permissions on the Cache.

As users subsequently perform revision control operations using `-share` mode, DesignSync creates new links to the new cache location. Over time as users access the objects in the cache, all of the links will point to the new cache location. To refresh a user's workspace at once so that all the links point to the new cache location, you can use the `refreshcache` command, which is available from `dss/dssc` and `stcl/stclc`.

Cleaning a Cache

If you are using a DesignSync file cache on your project with symbolic links, you may need to periodically clean out the cache to remove unused files and free disk space. Some typical scenarios that produce extraneous files in the cache include:

- Team members on your project used the UNIX `rm` command instead of the DesignSync `rmfile` and `rmfolder` commands; now there are orphaned files left in the cache.
- You upgraded your version of DesignSync, changed all cache links to the new cache file name format, and old cache files still remain.
- You notice extra files left in the cache after running Advanced Diff.

Notes:

- All users have access to the `cachescrubber` command, although it should only be run by the owner of the cache directory. It is possible for users to run it on a cache directory and remove files that should not be removed. To prevent users from inadvertently removing files from the cache, enable SUID for caches (as described in the *ENOVIA Synchronicity DesignSync Data Manager Installation* in the Program Directory.)
 - The `cachetouchlinks` script does not consider hard links during processing. When you use `cachescrubber` on a system that has hard links enabled, you should use the `-noref` option.

To clean a cache, follow these steps:

1. Run the `cachetouchlinks` script on all selected workspaces.
2. Run the `cachescrubber` script.

Both scripts are UNIX command line scripts written in Tcl. You can view an example of a shell script running `cachetouchlinks` and `cachescrubber`.

Marking Files for Cleaning with the `cachetouchlinks` script

The `cachetouchlinks` script searches through all workspaces and when it finds a cache-type object, it reads the value of the link and "touches" the cached file that is referenced.

For a collection object, it will touch each cached member file.

For a hard link object, it performs no operation.

For a description of the command and its available options, refer to the DesignSync Command Reference topic `cachetouchlinks`.

Cleaning with the `cachescrubber` script

When run in conjunction with `cachetouchlinks`, the `cachescrubber` removes all versions in the cache that have not been touched by the `cachetouchlinks` script, effectively removing any version in the cache that is not being used.

Important: When running the `cachescrubber` on a cache that uses hardlinks, use the `-noref` option as described in the following section, Removing versions in the cache that have no references using `cachescrubber`.

The `cachescrubber` is run with an "age in days" argument of 1 or more days.

Note: The "age in days" argument should be greater than 1 day if the `cachetouchlinks` script runs for more than 1 day. If `cachetouchlinks` runs for 2 days, then the `cachescrubber` should be run with 2 or more days. Use only integer values.

Removing versions in the cache that have no references using `cachescrubber`

The `cachescrubber` script can be used with the `'-noref'` option, without first using `cachetouchlinks` to remove versions in the cache that have no references.

Tip: The `-noref` option is recommended for systems on which hard links are enabled.

DesignSync tracks the objects in the cache such that when all team members relinquish their links to an object, that object is removed from the cache. To perform this tracking, DesignSync uses hard links with unique names to track the number of references to each file in the cache. Generally for each version in the cache, there will be at least one hard link reference. However, there are some cases where a version might not have a reference, such as when temporary files are fetched into the cache.

See [How DesignSync Manages Caches](#) for more information about references.

Note: When cache reference counting is disabled, there are no references in the cache. To clean the cache in this scenario, use the `cachetouchlinks` script first, and then `cachescrubber`. When cache reference counting is disabled in a cache containing only hard links, do not use the `cachetouchlinks` script, but instead use the `cachescrubber` script with the `-noref` option.

If you unintentionally remove a file that is still being referenced, refer to the DesignSync help topic [Troubleshooting Cleaning Caches](#).

Example

The following example indicates how a script could be created to clean up a cache.

```
#!/bin/csh -f
#
# Touch all cache files in all workspaces listed in " ws_list_file"
```


DesignSync Data Manager Administrator's Guide

```
cachetouchlinks -file /home/syncmgr/ws_list_file
if ($status != 0) then
echo "### cachetouchlinks failed when running with"
echo "          /home/syncmgr/ws_list_file ###"
exit 1
endif
echo "### successfully touched all cache files from workspaces listed
in"
echo "          /home/syncmgr/ws_list_file ###"
# If cachetouchlinks was successful, run the cachescrubber on
all cache directories
cachescrubber /home/syncmgr/caches/sync_cache 1
if ($status != 0) then
echo "### cachescrubber failed while cleaning sync_cache ###"
exit 1
else
echo "### successfully cleaned sync_cache ###"
endif
cachescrubber /home/syncmgr/caches/ASIC_cache 1
if ($status != 0) then
echo "### cachescrubber failed while cleaning ASIC_cache ###"
exit 1
else
echo "### successfully cleaned ASIC_cache ###"
endif
cachescrubber /home/syncmgr/caches/CPU_cache 1
if ($status != 0) then
echo "### cachescrubber failed while cleaning CPU_cache ###"
exit 1
else
echo "### successfully cleaned CPU_cache ###"
endif
exit 0
```

Note: If this shell script is run once a week, then the "age in days" argument could be an integer greater than 1. If the `cachetouchlinks` script ran for more than 1 day, then this integer must be greater than 1. If 5 "days" were used and this script was run once a week, this would cover the situation where the `cachetouchlinks` script ran for more than 1 day (and less than 5 days). Since `cachetouchlinks` was run 7 days earlier, then the "age in days" should be less than 7. You should record the time that `cachetouchlinks` ran so that the `cachescrubber` could be run with an "age in days" argument that is greater than the number of days used with `cachetouchlinks`.

Related Topics

[cachetouchlinks command](#)

[cachescrubber command](#)

How DesignSync Manages Caches

Setting Permissions on the Cache.

Files Missing from the Cache

Tips for Administering LAN Caches

The following tips describe how to administer DesignSync file caches effectively. These tips will ensure that team members carry out operations on the cache area correctly.

The discussions of some of these tips build on the scenario described in How DesignSync Manages Caches.

The following tips are described in this topic:

- Using hard links to preserve system resources
- Making sure your team uses the DesignSync `rmfile` command rather than the UNIX `rm` command
- Removing unused links and nonexistent files from your cache area
- Ensuring that team members use the same vault URL to prevent multiple copies of cache objects
- Using a project-specific registry file to restrict cache access to a project group

Using hard links to preserve system resources

Using hard links, rather than symbolic links, whenever you can, saves inode resources and can save disk space.

Hard links use a single inode per cache file regardless of how many links are created to the cache. Symbolic links use one inode per cache file connection. For example, if you have five users who have a workspace with symbolic links to the cache, each file within the cache will use 6 inodes. With hard links, each file will use one inode.

Symbolic links on some UNIX systems also require a small amount of additional space if the full path name of the cached object is longer than 64 characters. Hard links do not have the same requirement so there may be some disk space savings using hard links.

You can only use hard links if your setup meets the following conditions:

- Both file cache and workspace are on the same disk file partition.
- Both file cache and workspace have the same mount point.
Note: When automount is used, for example, with home directories, the mount points may be different even though the workspace and cache are on the same file partition. You may need to consult your system administrator to find out how the directories are mounted.

- Hard links must be enabled using the Links panel in SyncAdmin, or the `PBFCEEnabled` registry key.

Using SUID with your cache

DesignSync recommends enabling SUID when using file caches. For more information about the benefits of using SUID and the instructions to enable SUID, see the "SUID Configuration - UNIX only" section of the ENOVIA Synchronicity DesignSync Data Manager Installation file.

Make sure your team uses the DesignSync `rmfile` command rather than the UNIX `rm` command.

For a cache area containing symbolic links to be managed effectively, all team members should use the DesignSync `rmfile` command rather than the UNIX `rm` command to remove files from their workspace. If users remove files from their workspaces using the UNIX `rm` command, DesignSync's reference count will always be greater than the number of links to the cached objects, leading to unnecessary objects in the cache. If users at your site routinely use the `rm` command instead of the DesignSync `rmfile` command to remove files, you can manually clear your cache area of unused files and links. See [Cleaning a Cache](#) for more details.

Caches using hard links can remove their files with the UNIX `rm` command, but it is not recommended for a different reason. Using hard links, when the last linked workspace version is removed, the cached version remains in the cache and you must manually clear your cache area of unused files and links. See [Cleaning a Cache](#) for more details.

Remove unused links and nonexistent files from your cache area.

If users at your site routinely use the `rm` command instead of the DesignSync `rmfile` command to remove files from their work area, DesignSync cannot automatically remove unused files from the cache. You can manually clear your cache area of unused files by following these steps:

1. Have all users refresh their links to the cache area by populating with the `-get` option:

```
dss> populate -rec -get
```

Important: Be sure that all links to cache objects have been removed from your users' workspaces prior to step 2.

2. Remove the `sync_cache` contents:

```
% cd sync_cache
```

```
% rm -rf *
```

3. Have all users populate their workspace with the `-share` option:

```
dss> populate -rec -share
```

Note: This method should only be done if all users have the latest version in their workspaces. .

Ensure that team members use the same vault URL to prevent multiple copies of cache objects.

DesignSync generates cache file names for objects it copies into the cache area, as well as reference links to track objects in cache. See [How DesignSync Manages Caches](#) for details. DesignSync generates a cache file name using a hash of the entire vault URL, including the server name and server port. The name also includes version number and check in time of the version.

If users specify the vault URL using different conventions, DesignSync may generate different names for the same cache object. In this case, DesignSync generates multiple files for each object in cache, thus wasting disk space. For example, if user1 references a version of `top.v` with the URL

```
sync://myhost:2021/Projects/proj1/top.v;1.1, but user2 references top.v with the URL
```

```
sync://myhost.mydomain.com:2021/Projects/proj1/top.v;1.1,
```

DesignSync generates two different cache file names such as `s5e29e3477fcc2868-1.1-982002979-0` and `sa387c9833vgf9835-1.1-982002979-0`. The benefits of using a cache and sharing disk space can only be gained if users adopt the same conventions for referencing the vaults on your LAN.

If you suspect users have been using differing conventions to reference vault objects, you can verify this and remove the extra copies of cache objects as follows:

1. Check users' vault names for objects using this command:

```
dss> url vault top.v
```

2. Decide on a convention such as using `sync://<hostname>:<portnumber>/Projects/<projectdir>/<filename>` to reference vault objects.
3. Instruct users who did not reference the vault by this method to reapply the `setvault` command with the `-recursive` option to their project hierarchy. See the `setvault` Command for details.
4. Instruct users to run the `refreshcache` command. If the users already have the latest files, instruct them to `populate -get` to remove the links to those versions.
5. If the users performed the `populate` in the prior step, instruct them to `populate` with the `-share` option to replace their links to the cache.

Note: this step is not necessary if the refreshcache command was run in the prior step.

See [Cleaning a Cache](#) for more details.

Note: You can omit the server's host and port from the algorithm used to determine the cached filename. See the [Cached file uniqueness \(CacheUseHostPost\)](#) topic for details.

Use a project-specific registry file to restrict cache access to a project group.

you can set group permissions to limit access a cache area as described in [Setting Permissions on the Cache](#). You can also create a project and limit the users who can access that project. See [Project Configurations](#) for details.

Related Topics

[How DesignSync Manages Caches](#)

[Setting Permissions on the Cache.](#)

[Cleaning a Cache](#)

How DesignSync Determines the Correct Project Cache

There are two possible methods by which DesignSync determines project cache information:

- **Managing Project Cache Definitions with a TCL Script** -Project cache definitions are stored in a TCL script read by the system at DesignSync client (dssc, stcl, syncd, DesSync) startup time. When the client performs an operation that uses caching, it loads the project cache definition into memory.
- **Managing Project Cache Definitions with Registry Keys** - Project cache definitions are stored in registry files directory. When the DesignSync client starts, it generates and stores the list of vault or module URL/ project cache directory pairs into resident memory which is used whenever the client performs an operation that uses caching.

Benefits of Using a TCL Script to Define Project Properties

When the project properties are stored directly in the registry files, they are loaded at client startup and maintained in resident memory while the client is running. When there are many project definitions, this can result in a longer startup time for each client. When this information is stored in a script, the client only accesses and loads the definitions during the initial operation that access the cache. This means only the definitions that are used during the client session are loaded into the client.

Depending on how the script is constructed, the script can provide portability. For example, the script can be constructed to support local caching to geographically based server farms. If the user has a set of projects being managed across a geographically diverse infrastructure, the admin can create a single TCL file that includes project definitions that are not absolute links to specific servers, but include a variable that can be parsed to the local cache server or based on the vault naming convention that can be parsed to derive the cache directory.

Managing Project Cache Definitions with a TCL Script

A DesignSync project contains the vault location used to store project files and a cache directory used for sharing project files (UNIX only).

The project definition is managed by the DesignSync client (dssc, stcl, syncd, DesSync). The client sources the definitions as needed from a TCL script specified in the client registry. For more information on specifying the TCL script, see Project Registry Keys (ProjectCacheTclScript) and (Projects).

Creating the TCL Script

DesignSync includes a sample TCL script in the examples directory:

```
$SYNC_DIR/share/examples/doc/stclguide/getProjectCacheLocation.tcl
```

The script and the primary TCL proc in the script must have the same name. The example uses the filename `getProjectCacheLocation.tcl`, with `"getProjectCacheLocation"` as the primary TCL proc name. If you save the script to a different name, you must also rename the TCL proc.

For each project added to the script, you must know:

- The sync Module URL or files-based vault folder URL
- The cache directory path

Using the TCL Script

Save the TCL script into the site custom directory area:

```
<SYNC_CUSTOM_DIR>/site/share/client/tcl
```

Add the `ProjectCacheTclScript` registry key to the appropriate registry file. For more information on adding the registry key, see Project Registry Keys (ProjectCacheTclScript) and (Projects).

Restart any active clients to take advantage of the project mapping defined in the TCL script.

Managing Project Cache Definitions with Registry Keys

A DesignSync project contains the vault location used to store project files and a cache directory used for sharing project files (UNIX only).

The project definition is managed by the DesignSync client (dssc, stcl, syncd, DesSync). The project definitions are created through SyncAdmin or using registry keys, and loaded during client startup. For information on defining the project settings using SyncAdmin, see [Add or Edit a Project](#). For information on defining the project settings using registry keys, see [Project Registry Keys \(ProjectCacheTclScript\)](#) and [\(Projects\)](#).

The SyncURL that is specified in the pairs is a vault folder or module. DesignSync does not match on a file.

For example, when you configured your projects, you might create the following vault to cache mapping:

This Vault...	Maps to this Cache...
<code>sync://holzt:2647/Projects/CPU</code>	<code>/home/adams/Projects/cache/CPU</code>
<code>sync://holzt:2627/Modules/Chips/VR30</code>	<code>/home/adams/modules/cache/chip/VR30</code>

During the matching process, the DesignSync client adds a trailing slash to the vault URL, for example:

```
sync://holzt:2647/Projects/CPU/
```

```
sync://holzt:2647/Modules/Chips/VR30/
```

The trailing slash confines the matching process to vault folders in the cache and not files. Because DesignSync does not perform pattern matching the correct vault folder is found.

When you use a `-share` operation, DesignSync compares the SyncURL of the object to be put into the cache with the list of SyncURL / project cache directory pairs, matching the longest vault URL. DesignSync then uses the associated cache directory. If, at the time of the `-share` operation, and after the match is done, the associated cache cannot be reached or is not writable, DesignSync creates a reference to the object.

Example

You might have the following vault folders:

```
sync://holzt:2647/Projects/CPU
```

```
sync://holzt:2647/Projects/CPU22
```

When you perform a -share operation on a vault folder, DesignSync matches against the longest vault URL or lower in your hierarchy, such as `/home/adams/Projects/cache/CPU/`.

In this example, any -share operation you use on `sync://holtz:2647/Projects/CPU22` will not use the `/home/adams/Projects/cache/CPU/` cache directory. Any vaults in the `/CPU22` directory will use the default cache.

If the cache directory associated with the longest matched vault URL is not reachable, DesignSync does not use any other cache directory. DesignSync does not use the default cache, neither does it use a cache directory that might be associated with a shorter matched vault URL.

Note: If a project cache directory is unreachable when you start a DesignSync client, the URL/project cache directory pair is still used when looking for a match. If a match is found during a -share operation, DesignSync creates a reference to the desired object instead of a link to that project cache directory. The reference remains for the life of the client, even though the project cache may become reachable after the client starts. DesignSync still uses the longest match, even if a shorter match exists that is reachable. When the project cache can be reached, you must stop and restart your client to carry out successful -share operations.

Related Topics

[Setting Up a LAN Cache](#)

[Project Configurations](#)

[Project Registry Keys \(ProjectCacheTclScript\) and \(Projects\)](#)

[ENOVIA Synchronicity Administrator Tool Overview](#)

Understanding Excluding IP from the Cache

In order to help protect and restrict access to sensitive intellectual property, you can exclude objects from being cached, minimizing their possible exposure and providing a finer control over where the objects are stored.

This allows you to protect extremely sensitive IP while still taking advantages of the features of caching, particularly in a distributed system. This is a particularly valuable feature when working on joint collaborations with other companies. Shared information can be cached, and private corporate information can be excluded from the cache.

Rules for Determining Cacheability

Whether an object is cached is determined by the lowest level at which caching is disallowed. For example, if caching is disallowed at a server level (sync://serv1.ABCo.com:30126), no data below that point is able to be cached.

The following table shows the result if caching is disabled for the object:

Object	Description
sync://<host>:</port>	No objects on the server can be cached.
sync://<host>:</port>/Modules	No module objects on the server can be cached.
sync://<host>:</port>/Projects	No project configurations on the server can be cached.
sync://<host>:</port>/Projects/<object>[...]	The specific object or folder specified and if the object was a folder, objects contained within the folder recursively, can not be cached.
sync://<host>:</port>/Module/<category>[[...<category>]][<module>]	The specific category or module specified and the contents of the specified object recursively can not be cached.

Note: Objects in the sync://<host>:</port>/Modules/SYNC category are always cacheable, even that category is below a non-cacheable directive.

Related Topics

Adding or Removing an Object from Caching

Synchronicity Command Reference: caching disable

Synchronicity Command Reference: caching enable

Synchronicity Command Reference: caching status

Access Control Guide: Access Controls For Object Caching.

Adding or Removing an Object from Caching

To protect your intellectual property, you may wish to finely control what objects are placed in the cache.

Adding or Removing an Object from Caching

To prevent objects from being cached:

1. Launch your preferred DesignSync Client (dssc, stcl, DesSync).
2. From the command line, use the caching disable command to remove one or more objects from caching.
caching disable <SyncURL> [...<SyncURL>]
3. To verify that caching is disabled, you can view the caching status command.
caching status <SyncURL>

Note: By default, the ability to modify the caching status is disabled. To enable the ability to modify the caching status, see *Access Control Guide: Access Controls For Object Caching*.

To allow an object into the cache:

1. Launch your preferred DesignSync Client (dssc, stcl, DesSync).
2. From the command line, use the caching enable command to remove one or more objects from caching.
caching enable <SyncURL> [...<SyncURL>]
3. To verify that caching is enabled, you can view the caching status command.
caching status <SyncURL>

Related Topics

Understanding Excluding IP from the Cache

Synchronicity Command Reference: caching disable

Synchronicity Command Reference: caching enable

Synchronicity Command Reference: caching status

Module Caches

Procedure for Setting Up a Module Cache

This procedure explains the process of setting up a module cache for your team members to access over the LAN. Setting up the cache involves the following steps:

1. Create the module cache directory.
2. Populate the module cache.
3. Provide the module cache directory to your team members.

Note: If a user's request to link to the module cache is disallowed, DesignSync fetches the module from the server instead.

When using the data replication system, the mcache directory is created and populated for you. The mcache path directory you provide to your team members is, `<DRR_path>/module_cache`. For more information on setting up data replication, see [Setting up Data Repication](#).

Create the module cache directory

The system on which you create the directory must be a UNIX system accessible to your team members. You cannot create a module cache on a Windows system.

1. Log into the syncmgr account, or the account of the user who owns the SyncServer.
2. Create the module cache directory, if it doesn't already exist.

Note: Windows clients can use a legacy module cache, populating a copy of the legacy module from the legacy mcache instead of directly from the server, using the option **Module Cache Mode: Copy from the module cache** `-mcachemode copy`. This still uses the performance optimization characteristic of the mcache by reducing the load on the server. This option is not available for non-legacy mcachees.

The module cache directory should be writable only by the syncmgr account.

Populate the module cache

Use the `populate` command or the Populate dialog box to fetch modules from a server into the module cache . This section lists the populate options you may use to create the module cache as well as the options you cannot set.

Populate GUI option/Command-Line option	Availability	Description
Links to cache/ <code>-share</code>	Recommended	This forces the module contents to get fetched into the DesignSync cache (different from an mcache). Symlinks are

		created in the mcache to point to these files in the DesignSync cache. Alternately, you may populate with Unlocked copies / <code>-get</code>
Recursive / <code>-recursive</code>	Optional	This allows you to recursively fetch an entire module hierarchy. Users' links to modules in the module cache mimic that recursion. If you fetch a module recursively into a module cache, then users will only be able to link to that module if they fetch the module recursively. If you fetch an individual module into a module cache (non-recursively), then users will only be able to link to that module when fetching the module non-recursively.
Filter / <code>-filter</code> Exclude / <code>-exclude</code>	Prohibited	Do not filter the fetched module data. If a filter is used when fetching a module into a module cache, users will not be able to link to that module.
Href mode: Static mode / <code>-hrefmode static</code>	Required	This specifies populating a static version of a module into the cache. Users can only link to statically fetched module versions (modules fetched with a static href mode). This restriction prevents the use of outdated module versions in the module cache.

Note: Do not fetch overlapping modules into the module cache. Users are not allowed to link to overlapping modules in the module cache.

Provide the module cache directory to your team members

Provide the cache directory to users. Users enter that directory as the value for the **Module cache paths**/`-mcache_path` option.

Using SyncAdmin, you can define the default module cache paths that users' populate operations will search, when linking to a module cache. For more information, see Module Options.

Note: Users should not be able to write to the mcache. The module cache directory should be writable only by the syncmgr account.

Related topics

Updating a Module Cache

DesignSync Data Manager User's Guide: What is a Module Cache?

DesignSync Help: Using a Module Cache

DesignSync Data Manager User's Guide: Deleting a Module Cache

DesignSync Data Manager User's Guide: Populating Your Work Area

ENOVIA Synchronicity Command Reference: populate

Module Caches and Legacy Modules

A legacy module cache that contains legacy module releases cannot also be used for non-legacy modules. Similarly, a module cache containing non-legacy modules cannot also be used for legacy module releases.

A non-legacy module can have hierarchical references to legacy modules. DesignSync fetches the referenced legacy modules into the module cache, as part of the module hierarchy. However, users cannot link to or copy from the legacy submodules in the module cache.

Note that the `populate -mcachemode` option to copy data from a module cache only applies to legacy modules. Attempting to copy a non-legacy module from a module cache will fetch the module from the server.

Related topics

Setting up a Module Cache

DesignSync Help: Using a Module Cache

Updating a Module Cache

A module cache owner is responsible for updating the module cache with newer module versions and hierarchical references and removing old module versions that the design team no longer uses.

Note: When using the data replication system, the DRR maintains the mcache so it is always in the current state.

To update a module cache, use the `populate` command or the Populate dialog box to fetch a module from a server into the module cache.

Note: You should populate with the hrefmode static as users can only link to statically fetched module versions (modules fetched with a static href mode). You should populate in "share" mode, to take advantage of DesignSync's file caching. Do not filter the module data that is fetched. If a filter is used when fetching a module into a module cache, users will not be able to link to that module.

A module version that is no longer needed by your design team should be removed using the mcache scrubber. For more information, see [Cleaning a Module Cache](#).

When an mcache is updated, links to modules or hrefs that are removed in the mcache are broken. To remove broken links and update their workspace, team members should update their work areas by using **populate –recursive** to fetch the contents of the module cache.

Related topics

[Setting Up a Module Cache](#)

[DesignSync Help: What is a Module Cache?](#)

[DesignSync Help: Using a Module Cache](#)

[Module Options](#)

Cleaning a Module Cache

The mcache commands enable an administrator to remove unused module instances from module caches. Instances may be candidates for removal because they are too old or not currently used. For example, if a set of known user workspaces don't contain any mcache links to module cache instances.

You may have a work flow that pre-populates a module cache. It is therefore possible for a newly fetched module cache instance to have no references from a user workspace when a scrub operation is run on the mcache. As a result, new module cache entries would be prematurely removed before having the opportunity to be referenced in a user workspace. For example, suppose that an administrator has set up a weekly touch/scrub cron job. If a new module release is populated to a module cache and then scrubbed before any users have populated their workspace with the new release, it will be removed from the mcache. To prevent the new release from being scrubbed, the administrator can mcache touch the release.

You must run mcache scan or mcache touch prior to running mcache scrub, to ensure that active module cache instances are not inadvertently removed by mcache scrub.

The mcache commands are directly callable from a UNIX shell, similar to how the cachescrubber and cachetouchlinks commands are made available. In particular, this allows the mcache scan and mcache scrub commands to be run as cron jobs, to support automating module cache maintenance.

mcache scan searches a given list of user workspace paths looking for module cache links to module instances within a list of provided module cache paths. When a link to a module instance in the cache is found, a property attached to the module cache instance is updated with a timestamp of the current date/time. This property reflects the last time a module cache instance was known to be referenced, and is referred to as the "touch time".

mcache touch sets the "touch time" of a list of module cache instances to the current time. "mcache touch" updates a module cache instance timestamp property when it finds a module cache link to the instance within one or more provided workspace paths, similar to cachetouchlinks for file caches.

mcache scrub removes module cache instances in a list of module cache paths with a "touch time" older than a given age, similar to cachescrubber for file caches. Note that a module cache instance with parents cannot be removed regardless of the "touch time" unless all parents are also removed. This restriction is necessary to prevent module hierarchies from being damaged during a scrub.

Notes

There is no recognizable difference between a module cache and a workspace. As a result the mcache commands cannot distinguish between them and therefore assume that the workspace on which they are directed to operate is in fact a module cache.

The mcache commands are intended for administrators who have write access to the module caches.

The mcache commands are only applicable to modern module caches. They cannot be used to maintain legacy module caches.

No SUID functionality is provided.

Related Topics

Procedure for Setting Up a Module Cache

Understanding Excluding IP from the Cache

In order to help protect and restrict access to sensitive intellectual property, you can exclude objects from being cached, minimizing their possible exposure and providing a finer control over where the objects are stored.

This allows you to protect extremely sensitive IP while still taking advantages of the features of caching, particularly in a distributed system. This is a particularly valuable feature when working on joint collaborations with other companies. Shared information can be cached, and private corporate information can be excluded from the cache.

Rules for Determining Cacheability

Whether an object is cached is determined by the lowest level at which caching is disallowed. For example, if caching is disallowed at a server level (`sync://serv1.ABCo.com:30126`), no data below that point is able to be cached.

The following table shows the result if caching is disabled for the object:

Object	Description
<code>sync://<host>:</port></code>	No objects on the server can be cached.
<code>sync://<host>:</port>/Modules</code>	No module objects on the server can be cached.
<code>sync://<host>:</port>/Projects</code>	No project configurations on the server can be cached.
<code>sync://<host>:</port>/Projects/<object>[...]</code>	The specific object or folder specified and if the object was a folder, objects contained within the folder recursively, can not be cached.
<code>sync://<host>:</port>/Module/<category>[[...<category>]][<module>]</code>	The specific category or module specified and the contents of the specified object recursively can not be cached.

Note: Objects in the `sync://<host>:</port>/Modules/SYNC` category are always cacheable, even that category is below a non-cacheable directive.

Related Topics

Adding or Removing an Object from Caching

Synchronicity Command Reference: caching disable

Synchronicity Command Reference: caching enable

Synchronicity Command Reference: caching status

Access Control Guide: Access Controls For Object Caching.

Adding or Removing an Object from Caching

To protect your intellectual property, you may wish to finely control what objects are placed in the cache.

Adding or Removing an Object from Caching

To prevent objects from being cached:

1. Launch your preferred DesignSync Client (dssc, stcl, DesSync).
2. From the command line, use the caching disable command to remove one or more objects from caching.
caching disable <SyncURL> [...<SyncURL>]
3. To verify that caching is disabled, you can view the caching status command.
caching status <SyncURL>

Note: By default, the ability to modify the caching status is disabled. To enable the ability to modify the caching status, see *Access Control Guide: Access Controls For Object Caching*.

To allow an object into the cache:

1. Launch your preferred DesignSync Client (dssc, stcl, DesSync).
2. From the command line, use the caching enable command to remove one or more objects from caching.
caching enable <SyncURL> [...<SyncURL>]
3. To verify that caching is enabled, you can view the caching status command.
caching status <SyncURL>

Related Topics

Understanding Excluding IP from the Cache

Synchronicity Command Reference: caching disable

Synchronicity Command Reference: caching enable

Synchronicity Command Reference: caching status

Updating a Legacy Module Mcache

A legacy module cache owner is responsible for updating the module cache with new releases and removing releases no longer referenced in configurations that your design team uses.

To update a legacy module cache, use the `populate` command or the Populate dialog box to fetch the release from a server. Specify a base directory (with the `-dir` option) that has a unique name and is located directly below the cache directory. See Example Script for Updating a Module Cache. For a custom script determining the releases that need to be fetched to the module cache and then fetches those releases.

A release that is no longer referenced in a configuration's hierarchy should be removed using the DesignSync `rmfolder -recursive` command. See DesignSync Data Management User's Guide: Deleting Design Objects for information.

When an mcache is updated, links to releases that are removed from a configuration in the mcache are broken. To remove broken links and fetch a new configuration, team members should update their work areas by using `populate -recursive` option.

A nonrecursive fetch of a release to the module cache must have a different base directory from the same release fetched recursively. Suppose you have the entire hierarchy of a release in the module cache (fetched with `populate -recursive`) but the team wants to create links to just the upper-level module of the release, . When you fetch the release to the module cache (using `populate` without the `-recursive` option), you must specify a base directory that is different from the one you specified when you fetched the release recursively.

When you fetch a release hierarchy to the module cache (with `hcm get -recursive`), you should guard against overwriting one release with another.

Overwriting can occur because the get operation allows you to fetch a different configuration of the same module into an existing directory. For example, if you fetch a configuration containing the submodule ALU@R3 to the module cache and the ALU directory already contains ALU@R1, the get operation overwrites ALU@R1 with the contents of ALU@R3. This behavior (the default for the get operation) lets users update their work areas just by refetching a module configuration. In addition, in fetching to the module cache, it is possible for one or more of the release's submodules to be fetched to the top of the cache directory or even to a directory outside of the module cache directory. (This situation can occur because the `hcm addhref` command allows

specification of any relative path, even one that falls outside of the directory hierarchy of the upper-level module.) For example, some teams create module hierarchies that, when fetched to a work area, organize directories such that all of the configurations in the design are peers in a single, flat directory. (The hierarchical references all contain relative paths of the form `./<submodule>`.) Because all releases must reside as at the top level of the module cache, a fetch of such a module to the cache might overwrite releases in the cache with submodule releases being fetched to the same directory.

To identify releases with the potential of overwriting each other in the module cache, use the **hcm showhrefs** operation to review a configuration's hierarchy before fetching it to the module cache. If a configuration looks as though it may overwrite another, fetch it into a different module cache.

Example Script for Updating a Module Cache

This example script fetches the releases of a module into a module cache. The script fetches only those releases that are not in the module cache, are not available, or were not fetched to the module cache hierarchically.

You can use this example script to help you write your own module cache update script. To use this example script:

1. Write a script to call this procedure for each module for which all releases should be fetched.
2. Start a cron job to call the script.

Example Script:

```
#####
# Fetch the releases of a module into a module cache. Only
# those releases that are not in the module cache, are not
# available, or were not fetched to the module cache
# hierarchically are fetched.
#
# To use:
# 1. Write a script to call this procedure for each module
#    for which all releases should be fetched.
# 2. Start a cron job to call the script.
#####

proc populateMCache {mcachepath moduleUrl} {
    global errorInfo

    #
    # Get a set of the releases for the module by fetching the
    # configurations for the module from the server and filtering
    # out all but those with a configuration type of "Release".
    #
    if [catch {hcm showconfs -target $moduleUrl -report script} configList] {
        error "Failed to get configuration list from server." $errorInfo
    }
}
```

```

}
foreach {configName configPropList} $configList {
    array unset configPropArray
    array set configPropArray $configPropList
    if {[string equal "Release" $configPropArray(type)]} {
        set releaseArray($configName) $configPropList
    }
}

#
# If there are no releases for the module, we're done.
#
if {![info exists releaseArray]} {
    puts "No releases to fetch to the module cache."
    return
}

#
# Now get a list of the releases of the provided module that
# are available in the module cache and have been hierarchically
# fetched. These releases do not need to be refetched. Any
# release of the module that is not in the cache, not available,
# or non-hierarchical needs to be fetched to the module cache.
#
if [catch {hcm showmcache -mcachepaths $mcachePath -report script}
cacheEntries] {
    error "Failed to read module cache." $errorInfo
}

set availableReleases ""
foreach cacheEntry $cacheEntries {
    array unset cacheEntryPropArray
    array set cacheEntryPropArray $cacheEntry
    if {![string match "$moduleUrl*" $cacheEntryPropArray(target)]} {
        continue
    }
    if {!$cacheEntryPropArray(hierarchical) ||
!$cacheEntryPropArray(available)} {
        continue
    }
    lappend availableReleases $cacheEntry
}

#
# Get a list of the releases that need to be fetched to the module
# cache. This is done by removing from the list of releases
# available on the server, every release that is already available
# in the cache. What is left is a list of releases that are not
# available in the cache.
#
foreach cacheRelease $availableReleases {

    #
    # Since we're dealing with releases of a single module, it is
    # sufficient to compare the release names only. The release
    # names will be unique within the context of a single module.
    # So, get the release name for the cached release and look for

```

DesignSync Data Manager Administrator's Guide

```
# it in the list of releases from the server.
#
array unset cacheEntryPropArray
array set cacheEntryPropArray $cacheRelease
set release [file tail $cacheEntryPropArray(target)]
set releaseName [lindex [split $release @] 1]

unset releaseArray($releaseName)
}

#
# The releaseArray array now contains the releases that need to be
# fetched.
#
if {[array size releaseArray]} {
    puts "No new releases to fetch to the module cache."
    return
}

#
# Start fetching the releases that need to be fetched. Place
# each in a directory name of <module-name>@<release-name>
#
set failed 0
foreach {releaseName releasePropList} [array get releaseArray] {
    set target [join [list $moduleUrl $releaseName] @]
    set pathTail [file tail $target]
    set path [file join $mcachePath $pathTail]
    puts "\nFetching release to module cache."
    puts "\ttarget: $target"
    puts "\tpath:    $path\n"
    if [catch {hcm get -target $target -path $path -recursive -mcacheMode
server} result] {
        puts "Failed to fetch release."
        puts "\ttarget: $target"
        puts "\ncontinuing ..."
        set failed 1
    }
}

if {$failed} {
    error "Failed to fetch all releases."
}

return
}
```

Related Topics

[Setting Up a Module Cache](#)

[Updating a Module Cache](#)

The Mirror System

Mirroring Overview

A mirror exactly mimics the data set defined for your project vault. Mirrors provide an easy way for multiple users to point to the file versions that comprise their project's data. The file versions in the mirror belong to the configuration defined by the project lead. For example, the configuration could be the Latest version of files on the main Trunk branch. A mirror for a development branch may be defined to always contain the file versions on that branch with a specific tag. When the file versions in the configuration change, the mirror directory is automatically updated with the new version; for example, if Latest versions are mirrored and a new version of a file is checked into the vault. Without mirroring, users would need to frequently update their work areas using the `populate` command to reflect the project's current data set.

The `setmirror` command associates a workspace with a mirror directory. A mirror will always have accurate metadata because any action that writes to a mirror directory updates the local metadata in the mirror directory. When you use the `setmirror` command to associate a mirror directory, checking in an object will:

- create the new version in the vault,
- update the file in the associated mirror associated, and then
- update the metadata.

Mirror can be updated and administered automatically. See the topic [Administering Mirrors](#) for details.

Note: Any user who is creating or managing mirrors must have the access rights to use the mirror system and access to any specific mirror commands desired. For more information setting up access permissions for mirrors, see [Access Control for Mirrors](#).

The mirror system supports HTTP Proxy header filtering. For more information, see [Working with HTTP Proxy Header Information](#).

DesignSync Mirrors

When users make changes to objects in a repository, update notifications are transmitted to the associated mirrors. The mirror sites then respond to the notification by updating their mirror directories. Updates include changes made to the configuration of the objects such as tags, branches, and references. The mirror updates using a transaction queue that sends the updates to the server in a batch of transactions. The transactions are processed until the queue is empty and the mirror is synchronized.

This batch queuing of transactions prevents the mirror processes from getting overloaded and slowing down the server.

You can set up the following types of DesignSync mirrors:

- Normal mirror - Fetches design objects directly from the repository server. The mirror directory to project vault url (with selector) is explicitly mapped. The Mirror Administration Server loops thru these mirrors and processes a change set by matching the url in the change against the project vault url for that mirror.
- Autogen Mirror Script - Fetches objects directly from the repository server to a list of mirrors contained in a tcl script. The MAS will loop through these mirrors and, for matching URLs, invokes the script to locate the actual mirror directory. Mirrors generated by a script show up in the mirror status and reset pages with the name "Script Generated Mirror." For more information on the setup and configuration of the tcl script to auto-generate a mirror, see Using Scripted Mirrors.
- Primary mirror - Fetches design objects directly from the repository server and serves them to secondary mirrors. A primary mirror must be on a UNIX host that supports hard links.
- Secondary mirror - Fetches design objects from a primary mirror instead of directly from the repository server.

Secondary mirrors help reduce network traffic at the repository server. A secondary mirror communicates with the Repository Server to determine what objects need to be updated in the mirror. The secondary mirror fetches the updated contents from the primary mirror instead of from the Repository Server.

Note: Primary and Secondary mirrors are not supported for modules, scripted mirrors or generated mirrors.

When you create a mirror, submirrors for referenced data (referenced modules, configurations or DesignSync references) are created automatically. See Adding a Mirror for details.

You also can create and work with mirrors using the command-line interface. See the `mirror` command descriptions in the ENOVIA Synchronicity Command Reference for details.

You should specify the `populate` command `-incremental` option for mirror updates. When you specify the `-full` option, mirror updates traverse the entire vault hierarchy for the mirrored project and may take a long time to complete. See the `Co` options for all mirrors on the MAS (`CoOptions`) and `Populate` options for all mirrors on the MAS (`MUPNewVersForPopulate`) for details on setting `co` and `populate` options for mirrors. See the for details on `Populate Full/Incremental` (`PopulateUseIncremental`) for details about the `PopulateUseIncremental` registry key.

Mirror Servers

Mirrors are managed by a SyncServer. You can use DesignSync web access to create and administer two types of DesignSync mirror servers:

- Mirror Administration Server (MAS) - The SyncServer at a mirror site that manages the mirrors. You can create mirrors only on servers with an MAS.
- Repository Server (RS) - The SyncServer that manages a vault repository that is mirrored at one or more mirror sites.

When objects change in a vault repository associated with an MAS, the RS that manages the vault notifies the MASs associated with the vaults. The MASs then updates the affected mirrors.

A single SyncServer can be both an MAS and an RS.

See [Setting Up a Mirror Server](#) for details on how to set up a mirror server.

Mirror Attributes

- All actions that write to a mirror directory update the local metadata in the mirror directory. When looking at a workspace with objects in the mirror state, a combination of the local metadata for both the workspace and the mirror directory is used to determine the correct version of the objects. This allows the DesignSync command, such as the `ls` and `url` commands, to show the correct state of the object.
- Mirrors support all defined configurations.
- When a check-in occurs from a client, it creates a new version in the vault, returns control back to the client, and the client writes the object into the mirror and updates the local metadata in the mirror directory.
- A mirror write through will occur for all fetch states. Regardless of the fetch state, if a mirror write through is done, then the metadata is updated to reflect what was written to the mirror directory.
- No other commands, with or without the `-mirror` option, write through to the mirror. Commands like `populate -mirror` and `cancel -mirror` do not write to the mirror directory. However, the `co -mirror` command writes through to the mirror directory if the correct up-to-date version is not already in the mirror. Most commands only create links from a workspace to the files in the mirror directory.

As a DesignSync administrator, you can:

- Set up a mirror directory and navigate through this mirror knowing that everything is being kept up-to-date.
- Set up your environment (from that LAN where the check-ins occur) to write through to your mirror when checking a new version into the server. You do not have to wait for the mirror update process to update the mirror.
- Check the status of a mirror.

- Requeue any failed mirror transactions.

Restrictions

- Because mirroring is implemented with UNIX links, mirrors are not supported on Windows platforms.
- The mirror directory and the users accessing it must be on the same LAN.
- Mirrors for modules cannot be linked to from a workspace. A module cache should be used instead.

Related Topics

General Mirror Topics:

Mirrors Versus Caches

DesignSync Data Manager User's Guide: Using a Mirror

Mirror Administration Topics:

Administering Mirrors

First Steps for Troubleshooting the Mirror System

Architecture of the Mirror System

How Mirrors Work

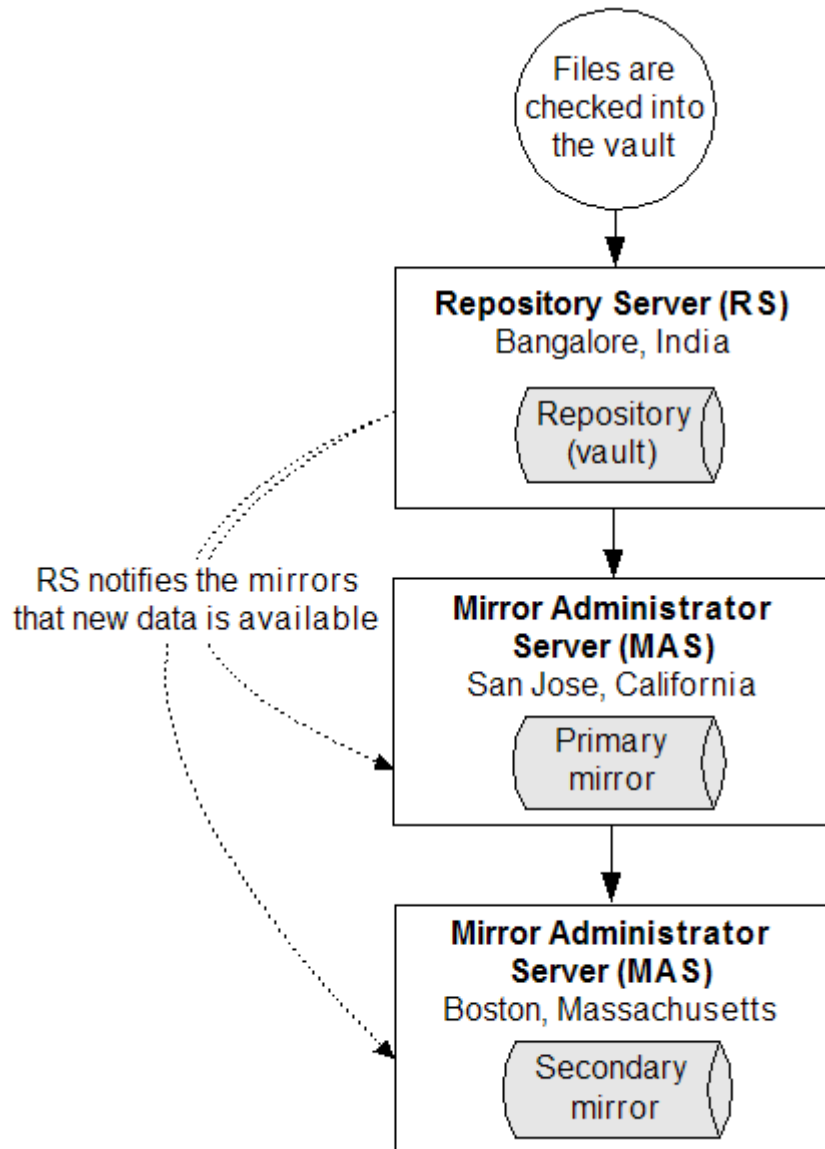
The Repository Server (RS) records all the mirrors that fetch its project data. Whenever activity occurs on the RS, the change is recorded in a transaction log file.

The system continuously checks the RS transaction log for updates. When new entries appear in the log, the mirror sites are notified. The MAS at each mirror site receives the change set and updates the mirrors that are affected by these new entries. The transactions are received in batches which are broken up into batches and processed individually. If for any reason the transaction processing fails, the mirror system writes out the transactions in the queue, including the ones that failed and the transactions queued up behind the failure. When mirror transaction processing resumes, it begins by processing the new transactions that have come in. The saved transactions can be manually or automatically requeued for processing. For more information on setting up transaction requeuing, see *Understanding Mirror Transaction Queuing and Transaction Failures*.

Secondary mirrors are updated after the primary mirror site has been updated. If a new version of an object is not yet available on the primary mirror, the secondary mirror

continuously attempts to fetch the required object from the primary mirror until the object is available.

A system that uses both primary and secondary mirrors might be like the following:



In this example, new design files are checked into the RS in Bangalore. The RS has a record of all the mirrors that reference its repository and sends notifications to the MASs in San Jose and Boston. When these two mirror sites receive the notifications, the primary mirror gets the new data from the RS and the secondary mirror gets the new data from the primary mirror.

Whenever the MAS is notified of a change, either a checkout or populate operation is performed to update the mirror. Each mirror is populated once a day, regardless of

whether any changes have occurred. In addition, mirrors are populated under other circumstances:

When a mirror is created.

- For DesignSync vault or legacy module mirrors, when the number of files to be updated exceeds 100. When fewer than 100 files are checked into the repository, the files are updated in the mirror by a checkout operation. If the mirror needs to check out more than 100 files, a populate operation is performed. You can modify the populate and check-out options for a server or a port. See Populate options for all mirrors on the MAS (MUPNewVersForPopulate) and Co options for all mirrors on the MAS (CoOptions) for details.
- For non-legacy module mirrors, a populate is always performed.

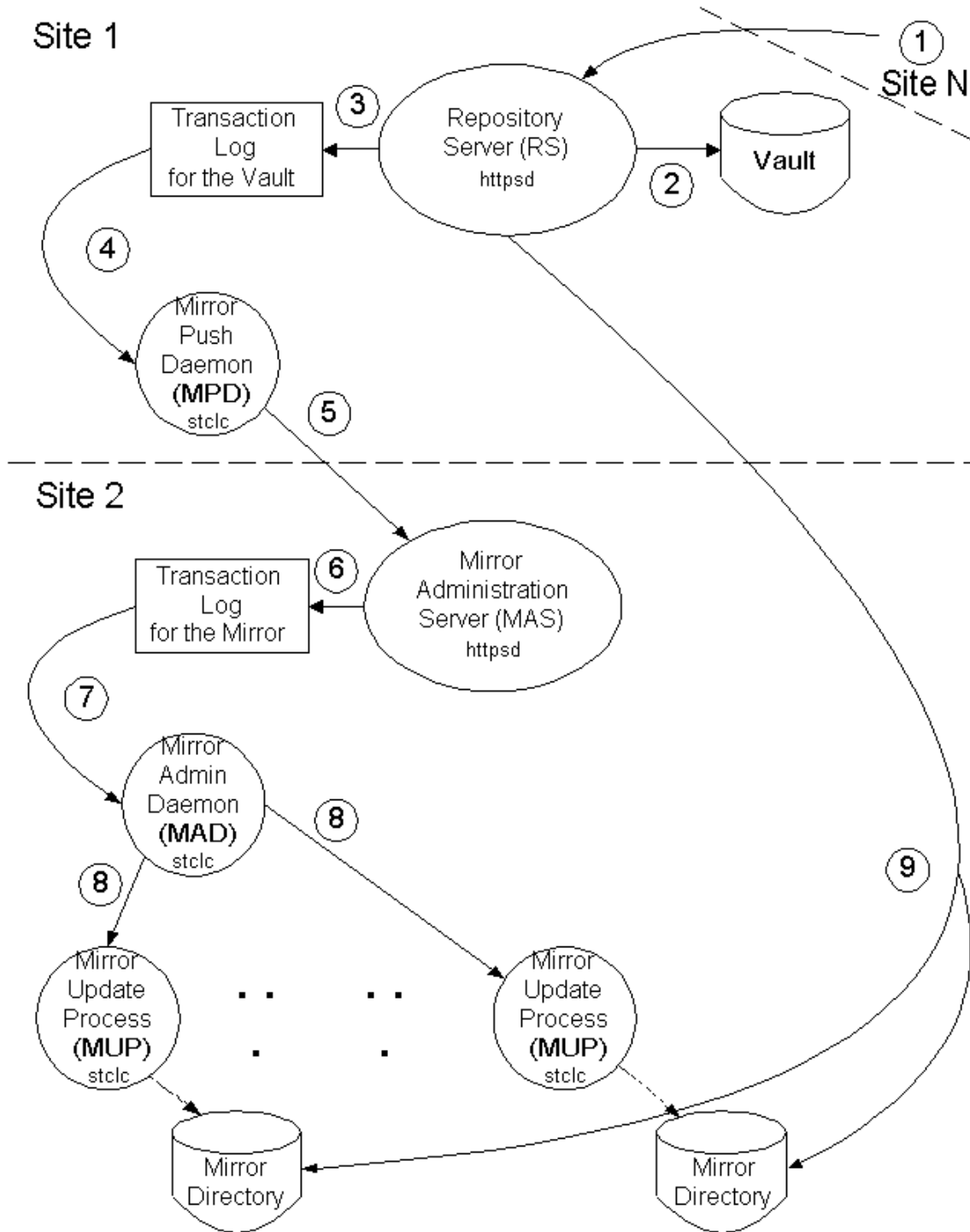
When a mirror is reset.

- Whenever an MAS or RS is restarted. Mirrors automatically resynchronize to the current state of the repository following a system interruption or power failure at an MAS or RS. You can use the Mirror Status panel to check the status of your mirrors and repository servers from either the MAS or RS.

Other operations can trigger populates (either incremental or full), such as modifying a selector.

Note: If you implemented the SUID solution for mirror directories during installation, all of your Mirror Administration Servers must be owned by syncmgr. If syncmgr is not the owner, mirror operations done through the SUID client tools will fail.

The flow diagram below shows how mirror directories are automatically updated, reacting to a change in the vault being mirrored. Descriptions of the steps follow.



1. A DesignSync client operation, from any site, is issued that will modify the vault data being mirrored. For example, a `ci` command is run, of a locally modified managed object.
2. A new file version is created in the Repository Server's vault.

3. The Repository Server updates its transaction log with the new version information, recording the change in vault content as vault log entries.
4. The Mirror Push Daemon on the Repository Server reads the transaction log via the Tcl interface, for the "change set".
5. The Mirror Push Daemon pushes the change set via `rstcl` to each Mirror Administration Server. Only the changes relevant to the mirrors being managed by a particular Mirror Administration Server are pushed to that Mirror Administration Server.
6. The Mirror Administration Server writes the changes to its transaction log as mirror log entries, and then returns control back to the Mirror Push Daemon. From the successful return of the Mirror Administration Server, the Mirror Push Daemon knows that the Mirror Administration Server has received the changes.
7. The Mirror Administration Daemon reads the change set from the mirror's transaction log via the Tcl interface.
8. The Mirror Administration Daemon spawns a Mirror Update Process for each affected mirror that needs to be updated.
9. The mirror directories are updated.

For modules, the Mirror Update Process uses `populate` to update the mirror.

For DesignSync objects and legacy modules, the update is done with `co` if the originating DesignSync client command was a `ci`. If the originating DesignSync client command was some other operation (such as `tag`, `mvfolder`, `retire`, etc.) then each Mirror Update Process performs a `populate`, to update the mirror. `populate` is also run by each Mirror Update process if the originating DesignSync client command was a `ci` that produced new versions of many files. For more information, see the topic Update command (MUPNewVersForPopulate).

Using Scripted Mirrors

Scripted mirrors reduce the administrative costs of manually creating mirrors as needed.

Using a TCL script, you can define the situations to create a static mirror rather than having to manually create the mirror. This reduces the burden on the mirror administrator and enhances communication across geographically dispersed development teams by providing immediate access to desired object versions.

Each geographic site that wishes to receive data from the repository automatically must run a special SyncServer called a 'Mirror Administration Server' (MAS). Mirror directories, associated with specific vaults, are registered with the MAS. The MAS automatically receives notifications from the repository whenever a defined relevant change such as a checkin or tag occurs and the script generated mirror is automatically updated.

The TCL script that contains the scripted mirrors definition must be accessible to the MAS. The MAS looks for the script in one of the two directories, as discussed in [Creating Scripted Mirrors](#).

The script contains a specified set of input parameters, for example: vault URL, tags, and a root mirror directory. The script locates a specific mirror directory for the project containing the specified vault. The script creates the mirror directory, if needed, and returns information back to the MAS. The MAS uses this information to locate the mirror directory and update the mirror by initiating a Mirror Update Process (MUP) to update the mirror, and add it to the list of active mirrors.

The MAS processes scripted mirrors as well as any other regular registered mirrors. More than one scripted mirror can be registered.

The performance of the MAS is directly related to the total number of normal and generated mirrors. Most of the generated mirrors are for static tagged versions. The generated mirror definitions are eventually automatically removed from the active list maintained by the MAS (as specified in the registry settings discussed in [Scrub Generated Mirrors \(ScrubGeneratedMirrors\)](#)). The mirror directories themselves remain intact. If any activity develops on such a mirror, the script sends an update to the MAS so the generated mirror gets placed back on the active list.

The status of a scripted mirror is visible in the Mirror Status panel of the DesignSync web UI when the mirror generation or update is in process, or if there is a failure during mirror generation or update. Successful scripted mirror transactions are visible for a short time, but are usually cleared quickly to allow users to focus on normal mirrors and scripted mirror failures.

Creating Scripted Mirrors

The TCL script that contains the scripted mirrors definition must be accessible to the MAS. The MAS looks for the script in one of the following two directories:

- `syncinc/share/tcl` (`$SYNC_DIR/share/tcl`)
- `custom/site/share/tcl` (`$SYNC_CUSTOM_DIR/site/share/tcl`)

DesignSync includes an out-of-the-box script to mirror all the projects from a specified repository. You can use this script as is, or customize it. This script is called `generate_mirror.tcl` and is located in `$SYNC_DIR/share/tcl`.

Note: Within the script that creates scripted mirrors, DesignSync recommends that you disable transactions for non-cacheable objects. For more information on disabling

transactions for non-cacheable objects, see [Send Uncachable Transactions to the Server \(SendUncacheableTransaction\)](#).

Related Topics

[Adding a Mirror](#)

[Using Scripted Mirrors](#)

Using Mirrors with Modules

When you create a mirror for a module, a mirror directory is created for the module selector. If the module has hierarchical references to sub-modules and the module is being mirrored recursively, a submirror is created for each referenced sub-module.

Note: If you have two hrefs to a sub-module in a hierarchy, only one submirror is created for the sub-module.

When a hierarchical reference is removed from a module, the sub-module is removed during the update of the upper level module. When a sub-module is removed from the mirror, the submirror is deleted.

When a new hierarchical reference is added and module is being mirrored recursively, the hierarchical reference is populated in the mirror and a new submirror is created for the submodule.

You cannot use a mirror directory for a module that is already being used by a legacy module or a DesignSync vault. You cannot use a mirror directory for a DesignSync vault or a legacy module that is already being used for a module.

You can automatically create generated mirrors of module versions by defining a scripting mirror. For information on defining scripted mirrors, see [Using Scripted Mirrors](#).

See the `rmhref` command description on how to remove a hierarchical reference.

Using Mirrors with Legacy Modules

When you create a mirror for a legacy module, a mirror directory is created for the module configuration. If the module configuration has hierarchical references to other configurations, a submirror directory is created for each referenced configuration residing within the parent configuration's directory cone (hierarchy of subdirectories) as well as each configuration residing outside that directory cone.

When a hierarchical reference is removed from a module configuration, the effect on the submirrors depends on the relationship between the parent (referencing) configuration and the referenced configuration:

- When a hierarchical reference is removed and the referenced configuration resides within the directory cone of the parent configuration, DesignSync deletes the submirror directory and disables the submirror.
- When an hierarchical reference is removed and the referenced configuration resides outside the parent configuration's directory cone, DesignSync retains the submirror directory and does not disable the submirror.

See the `rmhref` command description on how to remove a hierarchical reference.

Related Topics

[Adding a Mirror](#)

[Viewing or Editing a Mirror](#)

[Creating Scripted Mirrors](#)

Finding Mirrored Data

Mirror administrators need to identify available primary mirrors, when defining a secondary mirror. End users need to identify where data is being mirrored at their site, so they can `setmirror` to that mirror directory. DesignSync's `mirror wheremirrored` command can be used for both of these purposes. The `mirror wheremirrored` command can also be used to report the status of mirrors, and is faster than using the **Show RS Status** panel, or using the `mirror status` command with the `-servertype RS` option. (When options to the `mirror wheremirrored` command are used to decrease the number of mirrors whose status is reported.) Only active (enabled) mirrors are reported by `mirror wheremirrored`.

The information shown by the `mirror wheremirrored` command is for RS's that are running DesignSync V6R2009 or higher. RS's should be upgraded to a version of DesignSync that supports the `mirror wheremirrored` command first, so that the RS's are able to receive "wheremirrored" information. If a MAS is upgraded from a version that doesn't support "wheremirrored" to one that does, the "wheremirrored" upgrade step must be performed after each associated RS is upgraded.

See the `mirror wheremirrored` command documentation for details on the upgrade process.

Related Topics

[ENOVIA Synchronicity Command Reference: mirror status](#)

ENOVIA Synchronicity Command Reference: mirror wheremirrored

ENOVIA Synchronicity Command Reference: setmirror

Displaying Mirror Status

Secondary mirror fetch (SMAllowFetchFromRepository)

Viewing or Editing a Mirror

The Mirror Definitions panel lets you manage your DesignSync mirrors. From this panel, you can select a mirror to edit, view, enable or disable, delete, or reset. You also can configure the table on the Mirror Definitions panel to display only the information you need.

To access the Mirror Definitions panel, select **View/Edit Mirrors** from the **Mirrors** section of the **DesignSync** menu in the DesignSync Web UI.

You also can work with mirrors using the command-line interface. See the `mirror` command descriptions in the ENOVIA Synchronicity Command Reference for details.

The Mirror Definitions panel displays a table that lists all of the mirrors on your MAS. You can sort the information in a column by clicking on its column heading. Clicking a second time reverses the sort order.

In the **Selection** column, choose the mirror(s) to manage by selecting one or more of the checkboxes next to the mirror name. At the bottom of the column, you can click **Select All** to select all of the mirrors or **Clear All** to remove all of the selections.

After you have selected one or more mirrors, use the buttons at the bottom of the table to manage the mirror(s):

- Click **View** to go to the View Mirror panel, where you can see details of the mirror definition.
- Click **Edit** to go to the Edit Mirror panel, where you can amend the mirror definition.
- Click **Enable** to reactivate a disabled mirror.

When a mirror is enabled, it is effectively re-registered with the Repository Server (RS). The RS for each mirror is contacted and the mirror's definition on the RS is replaced. If the mirror cannot re-register, the mirror is disabled. Enabling an already enabled mirror could disable the mirror if the RS is unreachable.

- Click **Disable** to deactivate a mirror.

When a mirror is disabled, it is marked disabled on the MAS and an attempt is made to remove the mirror definition from the mirror's RS. Failing to remove the definition from the RS does not cause an error. The RS mirror definition is replaced when the mirror is enabled or removed when the mirror is deleted.

- Click **Reset Mirror** to perform a full populate of the mirror and leave the mirror directory in the same state as if the mirror had been removed and then repopulated.

Resetting a mirror also resets its submirrors. If the mirror and any of its submirrors are disabled, they are re-enabled by the reset operation.

- Click **Delete** to remove the selected mirror(s).

Deleting a mirror removes the mirror definition from both the MAS and the RS but does not stop any updates that are in progress. The mirror directory is not removed. If the MAS can be contacted but the mirror cannot be removed because the RS cannot be contacted, the mirror is disabled.

- Click **Options** to go to the Mirror Definitions Report Options panel, where you can configure what information appears in the table on the Mirror Definitions panel.

Viewing a Mirror

The View Mirror panel lets you see the details of a mirror definition. Click **Edit** to change the mirror definition.

Editing a Mirror

The Edit Mirror panel lets you change a mirror definition. The options are the same as those on the Add Mirror panel.

Click **Submit** when you have made your changes. Click **Reset** to remove your changes from the panel.

Editing mirrors that reflect modules

When the mirror reflects a module, you cannot edit the following options:

- Vault Selector
- Vault to Reflect

Editing scripted mirror definitions

Scripted mirror property changes are propagated to existing generated mirrors, with the exception of the following:

- Mirror Directory
- Vault Selector
- Vault to Reflect

Therefore, if any of these three scripted mirror parameters are subsequently changed, the existing generated mirrors will continue with the old parameters. Any new mirror generated by the script will inherit the changed parameters.

NOTE: A scripted mirror's *type* setting cannot be edited.

Mirror Definitions Report Options

The Mirror Definitions Report Options panel lets you configure the columns that appear in the table on the Mirror Definitions panel. You can configure:

- **Report Columns** - The default columns are listed in the text field. You can delete entries from this field to remove the column from the table on the Mirror Definitions panel. To add columns, select the information you want to display from the **<Field Name>** pulldown menu.
- **Sort Criteria** - The default sort order for rows of the table are listed in the text field. You can delete entries from this field to prevent DesignSync from sorting according to these criteria. To sort according to additional criteria:
 1. Select the information you want to sort on from the **<Field Name>** pulldown menu.
 2. Select either **ascending** or **descending** from the pulldown menu to list your sort criteria in ascending or descending alphabetical order. Ascending sorts are indicated by a **+** preceding the field name; descending sorts are indicated by a **-** preceding the field name.
 3. Click **Add to Sort Criteria** to insert the additional criteria into the text field.

If you select multiple values, the primary sort is by the first value specified, the secondary sort by the second value specified, and so on.

- **Save** - Select this option to save your selections. If you save your selections, the Mirror Definitions panel displays these choices each time you access the panel.

When you have made your selections, click **Done**. The Mirror Definitions panel appears and displays the table of mirrors in your custom format.

Related Topics

Mirror Overview

Setting Up a Mirror Server

Adding a Mirror

Displaying Mirror Status

Displaying Mirror Status

The Mirror Status panel displays the status of the mirrors administered by the mirror server you selected:

- If you are viewing status for a Mirror Administration Server (MAS), you see the status of the mirrors on the MAS server.
- If you are viewing status for a Repository Server (RS), you see the status of the mirrors that reflect vaults on the RS.

To access the Mirror Status panel:

- If a server has been defined as an RS and you have access rights, **Show RS Status** is displayed in the **Mirrors** section of the **DesignSync** menu of the DesignSync Web UI. Click **Show RS Status** to go to the RS view of the Mirror Status panel.

The RS view of the Mirror Status panel displays the status of all mirrors that reflect vaults on the RS. The status of the mirrors is determined by querying the MASs for the mirrors.

- If a server has been defined as an MAS and you have access rights, **Show MAS Status** is displayed in the **Mirrors** section of the **DesignSync** menu. Click **Show MAS Status** to go to the MAS view of the Mirror Status panel.

The MAS view of the Mirror Status panel displays the status of all mirrors on the MAS.

Response times when displaying mirror status from the RS may be slow because the RS contacts each MAS, for each mirror reflecting vaults on the RS. Instead, use the `mirror wheremirrored` command to obtain status. This will be faster, when options to the `mirror wheremirrored` command are used to decrease the number of mirrors whose status is reported. Only active (enabled) mirrors are reported by the `mirror wheremirrored` command.

You also can work with mirrors using the command-line interface. See the `mirror` command descriptions in the ENOVIA Synchronicity Command Reference for details.

The Mirror Status panel contains a table listing all the mirrors administered by your MAS. You can sort the mirrors by clicking on a column head. Clicking a second time reverses the sort order. All times are reported in the browser's time zone.

The banner in the Mirror Status panel displays the logical name of the server and the date and time when the status was obtained. You can configure what table columns to display, but the default table columns contain the following information:

- **Status** - An indicator of the health of the mirror. Possible status values are:
 - good (green) - The mirror is working normally.
 - warning (yellow) - The mirror update process has issued a warning. Either the update has failed once or the mirror update process (MUP) is taking longer than the time specified in the Warning Threshold option in MAS Settings panel.
 - failure (red) - The mirror has failed or the MAS has cannot communicate with the RS.
 - disabled (silver) - The mirror is disabled.
 - unknown/incomplete (blue) - The status of the mirror is unknown or the mirror(s) status file is not complete or in the correct format.
- **Mirror Name** - The name assigned to an MAS mirror. Click a hyperlinked mirror name to go to its View Mirror panel.
- **Category** - The optional name used to identify the mirror.
- **Last Up To Date** - The last time the mirror was up-to-date. If updates are in progress, this value is the starting time of the update.
- **In Progress** - Whether any updates are running.
- **Last Heartbeat** - The last time the mirror received a signal (heartbeat) from the RS. If a failure occurred, you can click on the hyperlinked time to view the log file with details of the failure.
- **First Failure** - The time of the first update process failure. (This time does not reflect heartbeat failures.) Click on a hyperlinked time to view the log file with details of the failure.

The time displayed in this column may not match the timestamp in the log file, which reflects the most recent failure.

Click **Options** to go to the Mirror Status Report Options panel, where you can configure the display of the table on the Mirror Status panel.

Click **View All Mirrors Log** to display the mirror log file in your browser window:

- If you are viewing status for an RS, you see the log for the repository server (`mpdlog.txt`). Errors when the MAS cannot communicate with the RS are displayed in this file.
- If you are viewing status for an MAS, you see the log for the mirror server (`madlog.txt`). Problems updating mirrors are displayed in this file.

Both of these log files reside in

\$SYNC_CUSTOM_DIR/servers/<host>/<port>/share/content/mirrors/logs.

If you have previously viewed the log file, you may need to refresh your browser to see the latest information.

Mirror Status Report Options

The Mirror Status Report Options panel lets you configure the columns that appear in the table on the Mirror Status panel. You can configure:

- **Report Columns** - The existing report columns are listed in the text field. You can delete entries from this field to remove the column from the table on the Mirror Definitions panel. To add columns, select the values that you want to display from the **<Field Name>** pulldown menu.
- **Sort Criteria** - The default sort order for rows of the table are listed in the text field. You can delete entries from this field to prevent DesignSync from sorting according to these criteria. To sort according to additional criteria:
 1. Select the information you want to sort on from the **<Field Name>** pulldown menu.
 2. Select either **ascending** or **descending** from the pulldown menu to list your sort criteria in ascending or descending alphabetical order. Ascending sorts are indicated by a **+** preceding the field name; descending sorts are indicated by a **-** preceding the field name.
 3. Click **Add to Sort Criteria** to insert the additional criteria into the text field.

If you select multiple values, the primary sort is by the first value specified, the secondary sort by the second value specified, and so on.

- **Status Format** - Select this option to display the mirrors' status using text instead of colored symbols.
- **Auto-Update** - Select this option to update the table at the specified interval. Use the pulldown menu to select the interval.
- **Save** - Select this option to save your selections. If you save your selections, the Mirror Status panel displays these choices each time you access the panel.

When you have made your selections, click **Done**. The Mirror Status panel appears and displays the table of mirrors in your custom format.

Related Topics

Mirror Overview

Setting Up a Mirror Server

Adding a Mirror

Viewing or Editing a Mirror

ENOVIA Synchronicity Command Reference: mirror wheremirrored

Administering Mirrors

Mirror administration can be done automatically through the data replication system, or manually, if desired. This topic discusses managing mirrors directly. For information on using the data replicated system to manage mirrors, see [Setting up Data Replication](#).

You can set up the following types of DesignSync mirrors:

- Standard mirror - Fetches design objects directly from the repository server.
- Primary mirror - Fetches design objects directly from the repository server and serves them to secondary mirrors. A primary mirror must be on a UNIX host that supports hard links. This is not supported for non-legacy modules. This is not supported for non-legacy modules.
- Secondary mirror - Fetches design objects from a primary mirror instead of directly from the repository server.

Secondary mirrors help reduce network traffic at the repository server. A secondary mirror communicates with the Repository Server to determine what objects need to be updated in the mirror. The secondary mirror fetches the updated contents from the primary mirror instead of from the Repository Server.

- Scripted mirror which will automatically generate a mirror when the conditions for creating a scripted mirror are met.

Mirrors are managed by a SyncServer. You can use the web interface to DesignSync to create and administer two types of DesignSync mirror servers:

- Mirror Administration Server (MAS) - The SyncServer at a mirror site that manages the mirrors. When objects change in a repository associated with an MAS, the MAS is notified and then updates its affected mirrors. You can create mirrors only on servers with an MAS.
- Repository Server (RS) - The SyncServer that manages a repository (vault) that is mirrored at one or more mirror sites.

You use the web interface to set up Repository Servers and Mirror Administration Servers and to create and edit mirrors. See [Mirroring Overview](#) for details on how to set up your mirror servers and your mirrors.

You also can use the `mirror` commands to work with mirrors. See the `mirror` command descriptions in the ENOVIA Synchronicity Command Reference for details.

Related Topics

General Mirror Topics:

Mirroring Overview

Mirrors Versus Caches

Mirror Log Files

Fetching Files from the Mirror or Cache

DesignSync Data Manager User's Guide: Using a Mirror

Understanding Mirror Transaction Queuing and Transaction Failures

The Mirror Administration Server (MAS) updates mirrors based on the information from the transaction it receives from the Repository Server (RS). When the volume of transactions is very high, the queue will get large. DesignSync places a limit on the volume of transactions that the mirror is capable of handling to optimize performance and throughput for the mirrors

If an update to a mirror fails, then the MAS creates a file containing the transaction that failed and all subsequent transactions in the queue, and, when it resumes, processes the subsequent transactions. The failed transaction can be retried automatically or manually, after the condition that caused the failure has been corrected.

The MAS status panel displays a list of:

- all current normal mirrors
- in-process scripted mirrors
- scripted mirrors that are in the transaction queue, but have not started processing

Using the MAS status panel, you can easily view any mirror transaction failures and requeue transactions after fixing the cause of the failure. There are three ways to resubmit the transactions that were not processed by the mirror after a mirror failure:

- Automatic Requeuing where the transactions move to the end of the queue and are processed in order, up to a maximum of 20 times, depending on the AutoRequeue settings.

- Manually Requeuing where the administrator chooses the time to send the transaction log containing the failed and unprocessed transactions through a DesignSync mirror requeue command.
- Manual Mirror Rerun simulates a mirror populate to recover and update the mirror.

Automatic Requeuing

When automatic requeuing is enabled, any transactions that failed, or, if the MAS has crashed, any transactions that were in the queue at the time of the crash, are automatically requeued for processing AFTER any subsequently created transactions are processed. This ensures that the system will not get hung up processing a single transaction that continues to fail. Automatic requeuing is disabled by default. For information on enabling automatic requeuing, see [Enable Automatic Requeuing \(AutoRequeue\)](#).

Manual Requeuing Using the Requeue Command

The Mirror Status Panel logs failed transactions to a transaction log file named `<mirrorName>.trans` located in the `$SYNC_CUSTOM_DIR/share/mirrors/run` directory. To rerun these transactions, use the `mirror requeue` command.

```
stcl> mirror requeue -name <Mirror> <Serverurl>
```

Tip: Using "*" as the mirror name will requeue all .trans files.

For more information on the `mirror requeue` command, see [The Synchronicity Command Reference: mirror requeue](#).

Running the Queued Mirror Transactions

If a mirror fails to update, DesignSync creates a script that can be run to set up the environment to repopulate the mirror. The run scripts are named `<MirrorName>.run` and are located in `$SYNC_CUSTOM_DIR/share/mirrors/run` folder.

When you notice failures in the Mirror Status panel, you can execute the .run shell script to run the automatically generated TCL code that will set the environment and repopulate the mirror.

```
> $SYNC_CUSTOM_DIR/share/mirrors/run/<MirrorName>.run
```

Scripting Email Notifications for Failures

To optimize mirror performance, DesignSync does not trigger an email notification for a mirror failure. However as an administrator, you may want to be notified periodically when there are transaction failures. The administrator can set up a cron job to

periodically fire off a tcl script to check for failures in the mirror status and, if failures are detected, send email to the administrator account. The tcl script will need to check the status of the mirrors, and check for failure status, and notify the administrator.

DesignSync provides some sample code for modification in
`$SYNC_INct/share/tcl/smirror/smirror_API.tcl`.

A very basic status checking script could include the following TCL code:

```
package require smirror::API
```

```
smirror::API::checkForFailureAndNotify
```

This will check for fail status on the mirrors, and notify the users specified in the 'NotifyList' using the 'General Settings' panel of the mirror system.

The cron job created should check periodically, perhaps once a day for mirror failures, for example, you can add your script to to daily cron directory (`/etc/cron.daily`). Your cron script should contain the following line, edited to include the appropriate values:

```
dssc rstcl -server sync://<host>:<port> -script <ScriptName>.tcl
```

Related Topics

Mirroring Overview

Verifying that the Mirror System is Running (HeartBeatInterval)

Save all mup log files (SaveLogFiles)

Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry)

Enable Automatic Requeueing (AutoRequeue)

Synchronicity Command Reference: mirror requeue

Setting Up a Mirror Server

The General Mirror Settings panel lets you enable your server as a Mirror Administrator Server (MAS) or a Repository Server (RS). An MAS server automatically manages the updating of the mirrors that populate from an RS vault.

To access the General Mirror Settings panel, click **General Settings** in the **Mirrors** section of the **DesignSync** menu in the DesignSync Web UI.

You also can work with mirrors using the command-line interface. See the `mirror` command descriptions in the ENOVIA Synchronicity Command Reference for details.

You can enable a server to be both an RS and an MAS.

Enabling a Mirror Administration Server

Check **Enable MAS** to configure the server as a Mirror Administration Server (MAS). The screen refreshes and displays the MAS options. After you designate a MAS and set your options, you must stop and restart the server to display the additional mirror options in the DesignSync menu.

If your mirror is used to manage Cadence objects, the user who starts the MAS must have the Cadence executable directories defined in his or her `PATH` environment variable. See DesignSync Data Manager User's Guide: How DesignSync Recognizes Cadence Data for details.

You can configure the following options:

- **Tracing** - Select this option to enable dynamic server/mirror tracing. Enabling tracing generates entries to the `madlog.txt` file located in:

```
$SYNC_CUSTOM_DIR/servers/<host>/<port>/share/content/mirrors/logs/madlog.txt
```

The `madlog.txt` file is rotated to avoid excessive disk space use. However, when you turn on tracing for debugging purposes, logs are not rotated.

mup (mirror update process) logs also contain debug output. The location of these logs is configurable, but by default they are in:

```
$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/muplogs/dss_<date>_<time>_<pid>.log (successful mirror update logs)
<mirror_directory>/_SYNC/saved.dss.mirror.log (failed mirror update logs)
```

Only one log file per mirror is maintained. When the mirror is regenerated, the old log file is replaced with the updated version. The Mirror Status panel "First Failure column" links to the files in this directory. For more information on locating MUP logs, and configuring the location of the MUP logs, see Save all mup log files (SaveLogFiles).

- **Default User** - Specify the user name and password for the default user used for communications back and forth between the MAS and the RS. The default user name and password also are used for communication back and forth between a primary mirror and a secondary mirror. The default user must be added as a user profile on the RS. Similarly, the default user for the MAS of a secondary mirror must be added as a user profile on the server for the primary mirror. If these user profiles are not added, any mirrors you create will be disabled.

When you create a mirror (see *Adding a Mirror*), you can select the user name and password of the user used for communication back and forth among the mirror servers. If you select the default user name and password, the default user is used. You can omit the user name if you have already defined a default user.

You must specify a default user when you use hierarchical references or DesignSync REFERENCES. Such references are registered as a submirrors, which require a default user.

- **Require Default User** - Select this option to avoid entering a user name and password for communications back and forth between the MAS and the RS when you create a mirror.
- **Require SUID** - Select this option to require that SUID to be activated before mirrors can be created or mirror directories can be moved on the MAS server. SUID is enabled by default. The SUID option sets the UNIX permissions on new mirror directories to 755. If this option is not selected, permissions on new mirror directories are set to 777 and are open to any user.

If you create a new mirror for a mirror that is already populated, possibly from an earlier installation, the permissions must be set appropriately on the contents of the mirror directory. You also must set permissions appropriately when converting a non-SUID mirror to an SUID mirror.

When you use SUID, the owner of the MAS server must also be the owner of the client tools (syncmgr).

- **Server Name** - Optionally enter a user-friendly name for the MAS server. If you do not supply a name, all references to the MAS in status reports default to the server's host name and port number. The system does not check whether the server name is unique across all MAS servers.
- **Notification List** - Enter the email addresses of users to notify when the mirror generates email notifications. These users receive notifications when the mirror is removed. You can use email addresses or user names separated by commas. The email addresses you enter are appended to the notification list when you create a mirror (see *Adding a Mirror*).

Note: You will not receive an email if the mirror fails or the mirror doesn't respond to a heartbeat check. You can optionally set up a script to run periodically and alert you to any problems. For more information, see [Understanding Mirror Transaction Queuing and Transaction Failures](#).

- Click **Modify** to display the CC List Helper, where you can select additional users to add to the list or remove users from the list.
- **Warning Threshold** - This setting allows the user to declare a mirror to have a "Warning" status if the mirror has not been up-to-date in a set period of time.

Note: If there is a large amount of data being populated, the mirror update process can take longer than the Warning Threshold value. In this case, if you have selected a low threshold value, the mirror returns a warning status when the threshold value is reached, even though the update process is proceeding normally. The mirror status becomes normal once the update is complete.

Warning Threshold for Low Disk Space

Warning e-mail is sent to the MAS administrator, if a disk partition used by the MAS falls below 32Mb of free disk space. Similarly, if the disk partition for a mirror directory falls below 32 Mb of free disk space, warning e-mail is sent to users on that mirror's notification list. As with other e-mail generated by the mirror system, warning e-mail is sent once every 24 hours, if the low disk space condition remains.

Unlike e-mail messages sent from the mirror system when errors are cleared, there isn't a follow-on message sent when available disk space is above the warning threshold.

If a mirror update fails due to running out of disk space, the standard e-mail for a mirror update failure is sent. That e-mail includes the failure reason.

The minimum amount of free disk space used for the warning threshold is configurable. You can specify the amount of free disk space as an absolute value, or as a relative percentage of the total disk space.

You can set different warning thresholds for the directories used by the MAS (such as for MUP logs), than for its mirror directories.

For configuration details, see [Absolute value for mirror directory disk space check \(FreeSpaceAbsWarn\)](#).

Enabling a Repository Server

Check **Enable RS** to designate the server as a Repository Server. The screen refreshes and displays the **Tracing** option.

Select **Tracing** to enable dynamic server/mirror tracing. Enabling tracing generates entries to the `mpdlog.txt` file located in:

```
SYNC_CUSTOM_DIR/servers/host/port/share/content/mirrors/logs/mpdlog.txt
```

The logs are rotated to avoid excessive disk space use.

Related Topics

Adding a Mirror

Viewing or Editing a Mirror

Displaying Mirror Status

Adding a Mirror

The Add Mirror panel lets you create a new mirror for a configuration of vaults in a DesignSync repository. After you create a mirror, users can use the `setmirror` command to associate the mirror directory with their workspace.

To access the Add Mirror panel, click **Add Mirror** in the **Mirrors** section of the **DesignSync** menu of the DesignSync web UI. This menu item is available only after you have set up a Mirror Administration Server (MAS) on the server hosting the mirror and then restarted the server. See [Setting Up a Mirror Server](#) for details.

You also can create a mirror from the command-line interface using the `mirror create` command. See [ENOVIA Synchronicity Command Reference: Mirror Create Command](#) for details.

Adding a mirror also creates submirrors for referenced data. See [Creating Mirrors for Referenced Data](#), below, for details.

A mirror containing custom collection (CTP) objects must have a definition for each custom collection.

You can specify the following options for your new mirror. Required fields are indicated by a red asterisk next to the field name.

- **Mirror Name** - Enter your own name for the mirror. The name must be unique among the mirrors already defined on the Mirror Administration Server (MAS).
- **Mirror Directory** - Enter the path of the mirror directory. Except for non-legacy modules, the path cannot be shared by other mirrors defined on the

MAS.

Note: For modules, a mirror directory can be the base directory for multiple modules. Since the base directory for multiple modules can be the same, multiple non-legacy modules can have the same mirror directory.

For non-legacy modules, the mirror directory path and the relative path of the module must be unique.

For non-legacy modules, you cannot use a mirror directory that is being used by a legacy module or a DesignSync vault. You cannot use a mirror directory for a DesignSync vault or a legacy module that is already being used for non-legacy a module.

For modules, if you have two hrefs to a sub-module in a hierarchy, only one submirror is created for the sub-module.

When defining a mirror, DesignSync always sets Unix permissions on the mirror directory, regardless of whether the mirror directory already exists. If you intend to change the Unix permissions, you should create the mirror as disabled. That will create only the top level mirror directory. You can then set the desired permissions on the top level mirror directory. Lastly, enable the mirror. That will populate the mirror directory, with sub-directories inheriting the permissions of the top level mirror directory.

If you are editing a mirror and you specify an existing mirror directory, the permissions on all sub-directories and the owner of all files and sub-directories must be the same as the top-level mirror directory.

- **Vault to Reflect** - Enter the Sync URL of the Module or DesignSync vault or configuration for a legacy module that you want to mirror. Specify the URL in the following form:

```
sync[s]://<host>[:<port>]/<path_to_vault>
```

where `sync[s]://` is the DesignSync protocol or secure protocol, `<host>` is the host name of the server, `<port>` is the port number of the server, and `<path_to_vault>` is the path to the server's vault. For example:

```
sync://zen.our_company.com:3758/Projects/ASIC/layout
```

- **Vault Selector** - Enter a selector or configuration that you want to mirror. You can use wildcard values within the selector list when creating generated mirrors from scripts. The default is Trunk:Latest.

Notes:

- Selectors, as with most DesignSync specifiers, are case sensitive.
- When creating scripted mirrors, you should not use dynamic selectors like Latest. Variable selectors should be generated with a specific wildcard character. For example, if having a naming scheme where all released modules are tagged with the prefix Rel and the date of release, you can set the selector as Rel*. This will create a new mirror each time a module is tagged in your naming scheme.
- **Category** - Optionally add a category name to help you sort and select mirrors using the Mirror Definitions panel and the Mirror Status panel.
- **Description** - Optional description to explain what data is being mirrored to allow for easier identification.
- **Notification List** - Enter the email addresses of users you want to notify when the mirror generates email notifications. You can use email addresses or user names separated by commas.

Users added to the **Notification List** on the General Mirror Settings panel are appended to the list on this panel.

- **Module Recursion** - Projects that use DesignSync references always recursively mirror the contents of their reference vaults. Select this option to mirror a Module's content and, recursively, the contents of all its submodules. This option is enabled by default.

Deselect this option to mirror only the module's contents. Any submodules of the module will not be mirrored. This option has no effect on non-module vaults.

Href Mode: This sub-option provides a set of three radio buttons (dynamic, static, and normal) to select the href mode.

- **Fetch State** - Select the fetch state of objects in the mirror.
 - **Get** - specifies updating the mirror with local copies. (Default)
 - **Share** - specifies updating the mirror with links to the file cache. The links to the file cache are defined with the Cache Type setting and can be either soft links or hard links.
- **Cache Directory** - Specify the path to the mirror specific file cache. If you are adding a mirror with the share state and no Cache Directory is specified, the default cache or project caches will be used, as determined by the MUP's registry files, which are made up of the server's registry files and the MirrorRegistry.reg file.
- **Cache Type** - Select the type of links created if the Fetch State is defined as **Share**.

- **Hard Links** - creates hard links to the file cache. When hard is specified, a hard link will only be created if it can be, meaning only if the file cache and the mirror are located on the same file system partition. If they are not, soft links are created instead automatically. (Default)

Note: Even if hard links are disabled, if the mirror is defined to use hard links, the mirror system will attempt to create hard links to the versions in the file cache.
- **Soft Links** - creates symbolic links to the file cache.
- **Mcache Mode** - Specifies how to handle module caches in the mirror.
 - **Link** - specifies creating mcache links to static module versions in recursively mirrored modules.
 - **Server** - specifies always fetching module versions from the server. (Default)
- **Enable Mirror** - Select whether the mirror is enabled after you create it. The default is to enable the mirror.
- **Mirror Type** - Select the type of mirror you want to create. Your choices are:
 - **Normal** - The default DesignSync mirror, which fetches design objects directly from the RS. This option is not supported for non-legacy modules.
 - **AutoGen Mirror Script** - The name of the TCL file that contains the scripted mirror definition. For more information on creating the mirror script, see [Creating Scripted Mirrors](#).
 - **Primary** - Fetches design objects directly from the RS and serves them to secondary mirrors. This option is not supported for non-legacy modules.

A primary mirror must be on a UNIX host that supports hard links.

When you create a primary mirror or change an existing mirror to a primary mirror, a full populate operation is automatically performed for the mirror.

- **Secondary** - Fetches objects from a primary mirror instead of directly from the RS. If you define a secondary mirror, you must provide the location of its primary mirror in the **Primary Mirror Server** field. Specify the URL as follows:

```
sync://<host>[:<port>]
```

You can also use `syncs://` for SSL ports, but you must use the SSL port defined when you installed the server.

Select **Use the default Username and Password** to use the settings for the default user defined in the General Mirror Settings

panel. This user is used to authenticate communication between the secondary mirror server and the primary mirror server. This option is available only when you have already specified default user in the General Mirror Settings panel.

Select **Specify Username and Password** to specify a user that will authenticate communication between the secondary mirror server and the primary mirror server.

The user name and password you enter in these fields must be added as a user profile on the repository server. Similarly, the user profile for the MAS of a secondary mirror must be added as a user profile on the server for the primary mirror. If these user profiles are not added, any mirrors you create will be disabled. These options do not appear when you have selected **Require Default User** in the General Mirror Settings panel.

- **Client to Server Communication** - Optionally control authentication for communication from the mirror on the LAN to the Repository Server. You can choose to:
 - Use the default username and password (set in the General Mirror Settings panel).
 - Specify a username and password.

This option does not appear when you selected **Require Default User** in the General Mirror Settings panel.

- **Server to Client Communication** - Optionally control authentication for communication from the Repository Server (RS) to the mirror. The user must have a user profile on the RS. You can choose to:
 - Use the default username and password (set in the General Mirror Settings panel)
 - Specify a username and password.
 - This option does not appear when you selected **Require Default User** in the General Mirror Settings panel.
- **Fetch State** - The fetch state to use when creating a scripted mirror. This can be overridden by specifying a fetch state value when creating the mirror.

The default value is "share." The only other allowed value is "get."

The registry setting is:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\
m1\FetchState="share"
```

- **Cache Directory** - The cache directory to use when creating a scripted mirror. This can be overridden by setting the cache directory value when creating the mirror.

There is no default value.

The registry setting is:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\
m1\CacheDir=" "
```

- **Cache Link Type** - The cache link type to use when creating a scripted mirror. This can be overridden by specifying a cache link type when creating the mirror.

The default value is "soft." The only other allowed value is "hard."

The registry setting is:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\
m1\CacheLinkType="soft"
```

- **Mcache Mode** - The Mcache mode to use when creating a scripted mirror. This can be overridden by specifying a mcache mode value when creating the mirror.

The default value is "server." The only other allowed value is "link."

The registry setting is:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\
m1\McacheMode="server"
```

Creating Mirrors for Referenced Data

When you create a mirror, submirrors for referenced data (hierarchical references to modules or legacy module configurations or DesignSync references) are created automatically.

If an href to a legacy module or DesignSync vault ends up using the same submirror directory because multiple hrefs de-reference to the same directory, then the vault URL and selector of the submirror directory must match. If the URL and selector do not match, the submirror is not created. You can work around this issue in one of two ways:

1. Change the relative path of the href or DesignSync reference that it does not resolve to the same submirror directory.
2. Define a different mirror directory for the parent mirror so that the href's or DesignSync reference's relative path does not overlap with an existing submirror directory when the path is resolved.

Related Topics

Setting Up a Mirror Server

Viewing or Editing a Mirror

Displaying Mirror Status

Setting Permissions for the Mirror

All mirror directories must grant full access to the users of the mirror, unless SUID is being used.

Even though mirrored files are read-only, a user must have write access to the mirror because the user's process sometimes updates the mirror directory when checking in a new version. This client write-through always happens when a mirror is set on the workspace using the `setmirror` command.. DesignSync checks that the user has write access to the mirror.

Note: It is possible and highly recommended to enforce the read-only intent of mirror directories (SUID) and not require that all users have write access. See the "SUID Configuration" section

Note: DesignSync creates mirror directories with wide-open permissions:

```
777 -- read/write/execute privileges for owner/group/others
```

To set different permissions for a mirror directory, see the Mirror Directory field description in the Adding a Mirror topic.

Related Topics

Mirroring Overview

DesignSync Data Manager User's Guide: Using a Mirror

Administering Mirrors

Adding a Mirror

Mirrors Versus LAN Caches

Resetting The Mirror Daemons

The **Reset MAS Daemon** (mad) and **Reset RS Daemon** (mpd) hyperlinks in the **DesignSync** menu in the DesignSync web UI, let you reset your mirror daemons for the server. Resetting the mirror daemons terminates the existing daemon process and starts a new one. These hyperlinks appear only when you have AdministrateServer access permissions. (See the ENOVIA Synchronicity Access Control Guide for details on access rules.)

Resetting the RS daemon clears heartbeat failures. The most common reason for heartbeat failures is an IP address change, typically on a system hosting a Mirror Administration Server (MAS). The mpd (mirror push daemon) on the Repository Server (RS) cached the original IP address, so can no longer communicate with its MAS's. Restarting the RS will clear this problem. However, you will not have to restart the entire server, only its mpd.

Note: In most cases, when a problem is detected with a mirror daemon, DesignSync automatically resets the daemon.

To reset the Mirror Daemon:

1. Click **Reset MAS Daemon** or **Reset RS Daemon** in the **Mirrors** section of the DesignSync menu.
2. Click **OK** on the confirmation pop-up window.

When the daemon has been reset, you see an Operation Successful panel that confirms the operation.

Note: To reset mirror services for all servers at your site or all servers at all sites, use the `resetmirrordemons` command. For command syntax and usage details, enter the following command in a DesignSync client:

```
stcl> resetmirrordemons -help
```

Related Topics

Architecture of the Mirror System

Legacy Mirrors

Legacy Mirrors

As of Version 4.2, we no longer support local mirrors nor the Remote Mirror Assurance package. Please see [Administering Mirrors](#) for the currently supported method.

Related Topics

[Upgrading Legacy Mirrors](#)

General Mirror Topics:

[Mirroring Overview](#)

[Mirrors Versus Caches](#)

[Using a Mirror](#)

[Setting Up a Mirror Server](#)

Upgrading Legacy Mirrors

The 4.1 release of Developer Suite introduced a new method for managing mirrors. For mirrors set up in earlier versions of Developer Suite, you can create a script to upgrade the legacy remote mirrors to the new functionality. If you have already upgraded your mirrors to current management scheme, you do not need to perform the upgrade again for 5.x.

DesignSync includes an upgrade utility that identifies your legacy remote mirrors and creates a script for upgrading the mirrors. You can specify which mirrors to upgrade. Note that this script does not convert local mirrors (vault mirrors).

Note: As of the 4.2 release of Developer Suite, the Remote Mirror Assurance package and local mirrors are no longer supported.

Before you start, make sure that:

- You have created the MAS server and set up its default user and parameters. See [Setting Up a Mirror Server](#) for details.
- The MAS server's default user also exists as a user on the RS server.
- Your MAS and RS are enabled. See [Setting Up a Mirror Server](#) and [Viewing or Editing a Mirror](#) for details.
- You have removed all mirror update cron entries for mirrors that you are going to upgrade.
- You have disabled the generation of Revision Control notes if they were enabled solely for the purpose of the mirror assurance check daemon trigger.

To upgrade your mirrors, you take the following steps:

1. Identify the host name defined for each of your mirror hosts.
2. Run the upgrade utility to create an upgrade script for your mirrors.
3. Run the upgrade script.
4. Remove RMA trigger.

The upgrade mirror steps are described in the following sections.

Identifying Your Mirror Host Names

To successfully upgrade your mirrors, you must supply the host name that was defined for each of your mirror hosts. To determine the names of your mirror hosts, enter the following command:

```
RemoteMirrorList -s <server_URL>
```

Where `<server_URL>` is the URL of the server for the vault.

The output from this command includes `HostId`, the host name of the machine that houses your mirror(s).

Here is a sample output from a `RemoteMirrorList` command:

```
Vault: sync://servalan.our_company.com:3021/Projects/Sportster
HostId: zen
MirrorPath: /home/syncmgr/Mirrors/Sportster
Interval: 60
NotifyList: vila@our_company.com
LastPopulated: Tue Aug 07 20:00:00 EDT 2005
LastModified: Mon Aug 06 15:18:01 EDT 2005
LastStatus: Unknown
```

See `RemoteMirrorList` for details on using this command.

Running the Upgrade Utility

To run the upgrade utility, enter the following at the `stcl` prompt:

```
stcl> package require smirror::upgrade
```

This command sets up the upgrade utility.

Then enter the following command:

```
stcl> smirror::upgrade::CreateScript -masurl <MASurl> -rsurl
<RSurl> -hostfilter <filter> -outfile <output-script-file> -
prefix <name>
```

You can specify the following options:

- `-masurl <MASurl>` - The Sync URL of the MAS server where you want to register your mirrors. For example:

```
-masurl sync://zen.our_company.com:3758
```

If you have more than one MAS, you can use the `-hostfilter` option to select the mirrors that should be registered with this MAS.

- `-rsurl <RSurl>` - The Sync URL of the server containing the vault that is referenced by the mirrors. For example:

```
-rsurl sync://servalan.our_company.com:3021
```

- `-hostfilter <filter>` - A filter that lets you specify what hosts you want to upgrade. Specify one or more host names and use the asterisk wildcard to select all mirrors on the host(s). The host name must be exactly the same as the name output by the `RemoteMirrorList` command.

For example, if you have legacy mirrors on the hosts `zen`, `blake`, and `avon` but you only want to upgrade the mirrors on `zen`, you would enter

```
-hostfilter zen*
```

- `-outfile <output-script-file>` - The name of the script to use to upgrade your mirrors. For example, to name your script `upgrade_mirrors`, you would specify:

```
-outfile upgrade_mirrors
```

- `-prefix <name>` - The name to use for the mirrors. When upgrading your mirrors, the script assigns a number to each mirror and prepends this numerical designation with the name you specify for the `-prefix` option. For example, to prefix the names of all the mirrors on your `avon` server with "avon," you would specify:

```
-hostfilter avon* -prefix avon
```

The mirrors on your `avon` server will be named `avon_0`, `avon_1`, `avon_2`, and so on.

If your MAS server will be managing mirrors on multiple servers, you need to run the upgrade utility to produce a separate script for each server.

After you have generated your upgrade script, you can edit the parameters in the script. For example, you may want to change the mirror names or add the `-RSuser <user>` option to pass the user name to the RS. The script contains commands like the following example:

```
#!/bin/sh
# the next line restarts using stcl \
exec stclc -- "$0" "$@"
package require smirror::clientCLI
mirror create sync://zen.our_company.com:3758
-name New_Push_Mirror_0
-mirror_dir /home/syncmgr/Mirrors/Sportster
-vault sync://servalan.our_company.com:3021/Projects/Sportster
-selector relA:
-notify vila@our_company.com
```

See the `mirror create` command description in the ENOVIA Synchronicity Command Reference for more information on using this command.

Running the Upgrade Script

After you have used the upgrade utility to create your upgrade script, run your script from the `stclc` command line. The utility converts the specified mirrors to the new mirror system.

If the script encounters an error, the mirror might be created but not enabled or the mirror might not have been created. The following are two recovery alternatives:

1. If there are multiple mirrors being upgraded and some were created and others were not, you could edit the script to remove or comment out any 'mirror create' commands for already created mirrors, fix the problem causing the script error and then rerun the script.
2. If the mirror was created but failed to be enabled, you should fix the problem that caused the enable to fail and then manually enable the mirror. If there were multiple mirrors being upgraded, for any mirrors that were already created, including this one, the 'mirror create' commands for these mirrors could be removed or commented out from the upgrade script.

Note: If mirrors were already successfully created, the 'mirror create' commands for these mirrors can be left in the script. This will result in added time to contact the MAS server and the mirror creation of those mirrors will fail anyway.

Remove RMA trigger

Remove the RMA checkin Notes trigger. If this checkin trigger remains, you will get an error during checkin operation:

Failure creating RevisionControl note.
 Notes for 'ci' will be disabled for the duration of the client process.
 Failure was: Failed:som: Error 5: Operation failed.

Once your mirrors are upgraded, you can use the DesignSync Web UI to manage the mirrors.

General Mirror Topics:

Mirroring Overview

Mirrors Versus Caches

Setting Up a Mirror Server

Scripts and Files used when Upgrading Legacy Mirrors

Setup of a legacy mirror upgrade involves the use of the following scripts:

RemoteMirrorList

Lists each registered remote mirror directory and provides status information on it.

Syntax:

```
% RemoteMirrorList -s <server_URL> [-h <hostid>]
```

This option...	Specifies...
-s	The URL of the server for this vault
-h (Optional)	A unique identifier for the machine on which the RemoteMirrorUpdate script runs Note: Some situations may require the use of the -h (hostid) option. See Using the -h Option for more information.

The RemoteMirrorList script provides the following status information:

This indicator ...	Shows...
Vault	The path to the vault that the mirror reflects including the selector
HostId	The unique identifier for the machine on which the RemoteMirrorUpdate script runs
MirrorPath	The path to the mirror directory
Interval	The interval, in seconds, of the time between checks of a mirror to evaluate how up-to-date it is

NotifyList	Addresses (specified when the mirror was registered) to which the script sends email when the mirror and the vault are not synchronized
LastPopulated	When the remote mirror was last populated by the RemoteMirrorUpdate script Note: The initial value of LastPopulated is the time when the RemoteMirrorRegister script was run.
LastModified	When the vault was last modified
LastStatus	The return from the populate command used by the RemoteMirrorUpdate script when it was run Note: Until the remote mirror update is run, the mirror's LastStatus will be listed as <code>Unknown</code> .
LastNotified	The time when e-mail was last sent (if the mirror is not synchronized with the vault) Note: The LastNotified value is displayed only after notification that the mirror is out of date has been sent. Therefore, this information may not be displayed on your system.

For example:

```
% RemoteMirrorList -s sync://mirrorserver.mycompany.com:2888
-----
Vault: sync://mirrorserver.mycompany.com:2888/Projects/ASIC
HostId: 808f2baa
MirrorPath: /users/admin/Projects/mirror/ASIC
Interval: 60
NotifyList: mirrormanager@mycompany.com sysadmin1@mycompany.com
LastPopulated: Tue Jun 06 15:18:01 EDT 2000
LastModified: Mon Jun 05 20:00:00 EDT 2000
LastStatus: Unknown
-----
```

Note: The `RemoteMirrorList` script sources the `SYNC_CLIENT_SCRIPT` file, which has a default value of `<SYNC_DIR>/syncinc.custom`. Remote Mirror Scripts use this file to define their custom variables.

(Remote Mirror scripts invoke an `stcl` session, which sources `<SYNC_DIR>/syncinc` upon start-up. In turn, `<SYNC_DIR>/syncinc` sources the `SYNC_CLIENT_SCRIPT` file.)

RemoteMirrorUnRegister

Removes a remote mirror directory from the list of registered mirrors, that is, the entry for this mirror is removed from the server's `RemoteMirrorList.txt` file.

Syntax:

```
% RemoteMirrorUnRegister -m <path_to_mirror> -f <force> -s
<server URL> [-h <hostid>]
```

This option...	Specifies...
-m	The file system path to the mirror directory to be updated
-h (Optional)	<p>A unique identifier for the machine on which the <code>RemoteMirrorUpdate</code> script runs. You would use this option if there are several remote mirrors being updated from the same server. For more information on this option, see Using the -h Option to Specify a Unique Identifier for a Remote Mirror.</p> <p>The default is the <code>hostid</code> of the machine where you run the <code>RemoteMirrorRegister</code> script. (In a typical remote mirror setup, you run both the <code>RemoteMirrorRegister</code> and <code>RemoteMirrorUpdate</code> scripts on the same machine.)</p> <p>Note: If you specify a value for this option with <code>RemoteMirrorRegister</code>, you must specify the same <code>-h</code> option value with this <code>RemoteMirrorUnRegister</code> script.</p>
-s	The URL of the server where the mirror is registered.
-f	<p>The force option.</p> <p>If the mirror directory specified is not associated with a server-vault, the script stops.</p> <p>Use the <code>-f <force></code> option to override this behavior.</p>

Note: The `RemoteMirrorUnRegister` script sources the `SYNC_CLIENT_SCRIPT` file, which has a default value of `<SYNC_DIR>/syncinc.custom`. Remote Mirror Scripts use this file to define their custom variables.

(Remote Mirror scripts invoke an stclt session, which sources `<SYNC_DIR>/ .syncinc` upon start-up. In turn, `<SYNC_DIR>/ .syncinc` sources the `SYNC_CLIENT_SCRIPT` file.)

RemoteMirrorList.txt

Contains information on each registered remote mirror.

- The path to the remote mirror
- The unique identifier for the machine on which the RemoteMirrorUpdate script runs
- The vault and selector associated with the mirror. If no selector is shown, it defaults to branch1:Latest.
- The check interval
- The notification list
- The last notification time
- The last modified time
- The mirror's last populated time
- The mirror's last populate status

Location: On the machine that is the server for the vault, in:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/share/RemoteMirrorList.txt
```

Associated files: The RemoteMirrorList script uses this file.

Related Topics

General Mirror Topics:

Mirroring Overview

Mirrors Versus Caches

Mirror Administration Topics:

Administering Mirrors

Upgrading Legacy Mirrors

Mirroring Overview

Setting Up a Mirror Server

Module Administration

Overview of Module Administration Tasks

Before you can use modules with the web interface to DesignSync, you need to perform some initial setup to ensure that they operate successfully:

- Upgrade any legacy DesignSync REFERENCES to hierarchical references. This allows you to take advantage of hierarchical query and subscription capabilities. For more information, see *The DesignSync Data Manager User's Guide: Upgrading DesignSync Vaults*.
- Set up email notification of module RevisionControl notes. For more information, see *Setting Up Email Notification of Module RevisionControl Notes*.
- Map note types, if the submodule's SyncServers use note types different from those of the upper-level module's SyncServer. For more information, see *Mapping Note Types*.
- Store logins, if your users do not have the same login on all SyncServers they access in their design work.

To determine if you need to store logins, set up the module hierarchy and perform a query. If the query succeeds, you do not need to store logins. For more information, see the *HCM User's Guide: Storing Logins*.

- Implement a policy regarding access to module commands governing module operations. For more information, see *Access Controls for Modules*.
- Customize the display of module commands and command options in the DesignSync GUI. This optional task allows you to present a common look across your user-base. For more information, see *Commands and Command Options*.
- Set up a module cache. This optional task improves the speed of module data transfers between the client and the server. For more information, see *Setting Up a Module Cache*.
- Determine if the DesignSync settings that affect the behavior of modules require changes. In most cases, Module operations follow DesignSync settings, for example:
 - Minimum checkin comment length. The checkin operation for modules follows this setting. For information about the minimum checkin comment length setting, see *The General Tab-Administrator View*.

In other cases, the settings are no longer applicable, for example:

- The registry setting for keeping version identifiers (RmVaultKeepVid) is no longer applicable.

Note: For legacy modules only, the **rmmod -vaultdata** operation follows this setting.

Important: Legacy modules do not follow the following DesignSync settings:

1.
 - Default fetch state. Modules do not support the DesignSync default fetch state of Always point to the latest version (Mirror). If a default fetch state of Mirror has been set in DesignSync, you must change that setting at the site or project level. For information about the default fetch state setting, see The General Tab-Administrator View in SyncAdmin Help.
 - Settings that specify dereferencing of symbolic links to files or directories.

Note: To operate on symbolic links to files or directories, the **populate** operation requires links to be both under DesignSync management and modified.

- Create module views on the server. For more information, see Overview of Module Views.
- Create external modules to provide linking between DesignSync and other code management systems. For more information, see Overview of External Modules.

Related Topics

Setting Up Email Notification of Module RevisionControl Notes

Moving Modules

DesignSync provides the ability to move modules to a different virtual location within the Modules location on the server. The module move process moves all the module members and hierarchical references and preserves the module history.

Note: The process does not update notes, access controls, subscriptions or mirrors. It does update the hierarchical references.

IMPORTANT: Before you move a module, verify that you have a current backup of the server.

Module Move Process

The module move (`mvmmod`) command combines the functionality of the module import/export and `modulereconnect` process into a single command to speed up and simplify the process when moving to a new location on the same server.

Note: To move to a different server, you must use the Module Export and Import functionality.

In order to run the `mvmmod` command, you must have access rights to the command. For information on setting access controls, see the *ENOVIA Synchronicity Access Control Guide*.

Note: If any of the referenced/updated modules are located on server versions prior to V6R2014, which do not contain the export/import functionality, you must install a server-side utility to allow the client running the `modulereconnect` feature to update the hierarchical references. For more information on the using export/import with older servers, see *Changing Hierarchical References on Older Servers*.

When you run the `mvmmod` command, the DesignSync server performs the following operations:

1. by default, freezes module in the old location so no changes can be made while the module is being moved.
2. bundles the module into a compressed format containing the history, hierarchical reference information, and module contents.
3. imports the module to the new location and recreates the hierarchical references.
4. updates the hierarchical references that point to the old module to the new module location.

Note: Both modules remain in a frozen state after the move.

To move a module:

1. Backup the original server containing the module. For more information on performing a backup, see *Backing Up Your Server*.
2. Use the `mvmmod` command to specify the old location of the module and the new location.
`mvmmod [-[no]freeze] <oldURL> <newURL>`
3. Make any structural changes desired and verify that the module is ready to be released for general use.
4. Verify that the access controls on the module are correct for usage in the new location.
5. Unfreeze the new module location.
`unfreezmod <newURL>`
6. Create any new mirrors, if necessary.
7. Update your team with the new module location information, if necessary.

Related Topics

Backing Up Your Server

Module Export and Import

ENOVIA Synchronicity Command Reference: unfreezmod

ENOVIA Synchronicity Command Reference: mvmod

ENOVIA Synchronicity Command Reference: importmod

ENOVIA Synchronicity Command Reference: exportmod

Module Export and Import

DesignSync provides the ability to move modules to a different server or a different location on the same server while preserving all the module members, hierarchical references, and module history.

Understanding Module Export

Understanding Module Import

Understanding Modules Reconnect

Procedure for Exporting/Importing a Module

Note: The export process does not export notes, access controls, subscriptions or mirror definitions.

Understanding Module Export

When you export a module, you create a compressed (tar) file. The tar file that's created is stored on the server in the following unique location in the \$SYNC_DIR data directory:

```
$SYNC_DIR/custom/servers/<host>/<port>/Export.sync/<category-  
path>/<module-name>.tar
```

Note: By providing a single, unique location for the archive file, DesignSync avoids the possibility of overwriting the archive with a different module of the same name. It also ensures that only one tarred version of the the module can exist on the server at any given time.

Part of the moving process (export and import together) focuses on updating the hierarchical references to and from the module. The hierarchical references to the module, (visible with the DesignSync whereused command) are not changed or removed when the module is exported. When the module is imported, the hierarchical references are recreated and this new module is added to the whereused information of the referenced submodules. DesignSync does not remove, on import, the references to the old module since you are not required to delete the module.

Important: By default, the command freezes the module before beginning the exportmod and does not remove the freeze when the operation completes.

Note: The export process does not export any purged module versions.

In order to run the module export command, you must have access rights to the command. For information on setting access controls, see the ENOVIA Synchronicity Access Control Guide.

The exportmod operation does the following:

1. by default, freezes the module so no changes can be made while the module is being exported.
2. bundles the module into a compressed format containing the history, hierarchical reference information, and module contents.
3. Optionally removes the freeze on the module.

Understanding Module Import

When you import a module, you import the compressed file that was created by the module export (`exportmod`) command. If you are moving the module to a different server you need to locate the existing compressed module and copy it to the correct location on the new server. The compressed tar file should be placed in the following location on the new server:

```
$SYNC_DIR/custom/servers/<host>/<port>/Import.sync/<category-  
path>/<module-name>.tar
```

Notes:

- This means you will need to know the new module name, if applicable, and the category location, if applicable, before importing the module. If you are changing the module name, rename the tar file to the new module name.
- The specified module location must be empty in order to import the module. If there is already a module in that location, you must remove it before performing the import.

In order to run the module import command, you must have access rights to the command. For information on setting access controls, see the *ENOVIA Synchronicity Access Control Guide*.

The importmod operation does the following:

1. creates a new module with the same name as the tar file (without the .tar extension) in the location on the server that matches the Import.sync subdirectory structure. If the categories do not already exist, DesignSync "creates" them.
2. unpacks the module members and history of the module.
3. stores the hierarchical reference information for the reconnectmod command.
4. optionally removes the compressed module (.tar) file.

Understanding Modules Reconnect

After you imported the module to the new location, you may want to reconnect the hierarchical references that referred to the old module to the new module and update the back reference information on the referenced modules of the moved module to point to the new module.

In order to run the module reconnect command, you must have access rights to the command on all the servers with modules being updated. For information on setting access controls, see the *ENOVIA Synchronicity Access Control Guide*.

The reconnect module operation does the following:

1. Modifies the module versions of the modules with hierarchical references to the old module to point to the new module. This allows you to remove the old module without fear of breaking the hierarchical references.
2. Updates the whereused information for all the module versions of the new module and removes the whereused information from the old module, if the old module still exists.

Note: If any of the referenced/updated modules are located on server versions prior to V6R2014, which do not contain the export/import functionality, you must install a server-side utility to allow the client running the modulereconnect feature to update the hierarchical references. For more information on the using export/import with older servers, see Changing Hierarchical References on Older Servers.

Procedure for Exporting/Importing a Module

This procedure walks through the full process of moving a module to a different server. It includes exporting the old module, importing the new module, reconnecting the

hierarchical references to the new module and removing the old module. The design flow that best suits your work environment may be slightly different.

To export/import a module:

1. Backup both the original server containing the module and the destination server for the new module. For more information on performing a backup, see [Backing Up Your Server](#).
2. Use the `exportmod` command to create the compressed, transportable module.
`exportmod [-[no]force] [-[no]freeze] <ServerURL>`
3. Copy or transfer the file from the export location (`$(SYNC_DIR)/custom/servers/<host>/<port>/Export.sync/<category-path>/<module-name>.tar`) to the import location (`$(SYNC_DIR)/custom/servers/<host>/<port>/Import.sync/<category-path>/<module-name>.tar`). The name and category path determine the name and location of the module on the server. If the module name is changing, rename the zip file to the new module name. For example, if both machines share NFS drives in a common DesignSync server location, the copy might look something like this:

```
cp
/tools/Dassault/DesignSyncServers/primaryServer/custom/servers/primaryServer/2647/Export.sync/ChipDesign/NZ20Chip.zip
/tools/Dassault/DesignSyncServers/newServer/custom/servers/newServer/2647/Import.sync/ChipDesign/NZ20Chip.zip
```
- Optionally, you can use `ftp` to transfer the file.
4. Use the `importmod` command to unpack and create the new module.
`importmod [-[no]freeze] [-[no]keep] <ServerURL>`
5. Use the `reconnectmod` command to update the hierarchical references. The `reconnectmod` command allows you update all references or specify specific modules to update.
`reconnectmod [(-parents <TCLlist>)|(-from <oldServerURL>)] <ServerURL>`
6. Make any additional structural changes desired and verify that the module is ready to be released for general use.
7. Verify that the access controls on the module are correct for usage in the new location.
8. Use the `unfreezmod` command to unfreeze the new module location.
`unfreezmod <newURL>`
9. Create any new mirrors, if necessary.
10. Update your team with the new module location information, if necessary.
11. Use the `rmmod` command to remove the old module.
`rmmod -[no]notes <serverURL>`

Related Topics

Backing Up Your Server

Moving Modules

ENOVIA Synchronicity Command Reference: unfreezmod

ENOVIA Synchronicity Command Reference: mvmod

ENOVIA Synchronicity Command Reference: importmod

ENOVIA Synchronicity Command Reference: exportmod

Changing Hierarchical References on Older Servers

When you move a module, part of the process involves updating the hierarchical references. If the moved module is referred to by modules on servers that do not contain the export/import feature (releases prior to V6R2014), you will need to install a utility on those servers to support updating the hierarchical references.

The server-side tcl script, `hcmChangeHrefs.tcl` is contained in distributions which support export/import functionality in this location:

```
$SYNC_DIR/share/tcl/hcmChangeHrefs.tcl
```

You must copy the script to the same location on the pre-V6R2014 server in order to make it available to the `reconnectmod` command (and the `mvmod` command, which uses `reconnectmod`).

After copying the script to the correct location, create a new access control granting the script Checkin rights to, minimally, any modules being reconnected. For information on setting access controls, see the *ENOVIA Synchronicity Access Control Guide*.

Module Views

Overview of Module Views

A design environment frequently includes modules with thousands of member files and many sub-modules. In most cases, a user only needs access to a subset of this data. If the data is stored on the DesignSync server in module format and managed as part of a project, a DesignSync administrator or project leader can define sets of filtering rules to create a particular view of the module that only includes the target files. This **module view** is stored on the DesignSync server for convenient repeated use. A DesignSync

user can then apply one or more of the module views when populating a workspace on the client.

By using module views, a DesignSync administrator or project leader can be sure the user has the correct files. Unlike using filters, where each user is expected to customize his own environment, using a module view, the DesignSync administrator can define the environment once and allow users to populate their workspaces uniformly.

Note: When applying a Module Views, the workspace stores the Module View name. When the Module View is used, the server traverses the module hierarchy, beginning with the module being processed, searching for the specified View name and uses the first matching Module View.

Use the DesignSync view command set to create and maintain Module views.

Related Topics

[Creating Module View Definitions](#)

[Viewing the Module View](#)

ENOVIA Synchronicity DesignSync Data Manager User's Guide: Understanding Module Views

ENOVIA Synchronicity Command Reference: populate command

ENOVIA Synchronicity Command Reference: setview command

ENOVIA Synchronicity Command Reference: view list command

ENOVIA Synchronicity Command Reference: view get command

ENOVIA Synchronicity Command Reference: view put command

ENOVIA Synchronicity Command Reference: view remove command

ENOVIA Synchronicity Command Reference: showstatus command

Creating Module View Definitions

A module view is created by loading a file containing the module view definition onto the server. The module view consists of a series of filter and hrefilter definitions supplied in a TCL list with the rest of the view properties.

The format of the TCL list:

```
{Name ViewName [Description {Description contained within braces}]
  [Filter filterdef[,filterdef2[,...]]]
  [HrefFilter hreffilterdef[hreffilterdef2[,...]]}
```

Name: The name of the view. The name must be unique within the module. The name may contain letters, numbers, underscores (_), periods (.), and hyphens (-). All other characters, including whitespace, are prohibited. This field is required.

Description: – A free-form text description of the view. If the description contains more than one word, you must enclose the description within braces, as shown above, or in double quotes. The text description can help clarify the purpose or target audience for the view. This field is optional.

Filter: DesignSync filter definition. There are three kinds of filters for Module Views.

- Include filter
- Exclude filter
- Extended include filter

The include and exclude filters are the standard filters used by DesignSync. The filter includes or excludes objects and is usually used for a group of objects, for example including or excluding all objects in a specifically named folder, or all objects with a certain file extension. For more information on constructing a DesignSync filter, see *ENOVIA Synchronicity DesignSync Data Manager User's Guide: Filter Field*.

The extended include filter, includes specific module objects, eliminating the processing overhead involved in filtering for matching lists of files. The extended include filter is applicable only for Module Views. The extended included format is:

```
++|<member_path>| [<member_path>| [...|] [, ]]
```

where:

++ indicates that this filter is an extended include filter.

<member_path> is an extended glob expression that includes the object path and name. An extended glob expression is a standard glob-style expression, but extended to allow the use of the ". . ." syntax to mean "match any number of directory levels".

| separates and terminates the extended include filter. If the filter contains both include filters and extended include filters, use a comma to separate the two filter types.

Note: The extended include filter is mutually exclusive with the exclude filter. You can use include filters with the extended include filter. You must specify at least one filter or hreffilter.

Important: The extended include filter **MUST** be prefixed with the ++| notation. Omitting the second + or the | results in the filter being treated as a standard include filter.

HrefFilter: DesignSync hreffilter definition. For more information on constructing a DesignSync filter, see *ENOVIA Synchronicity DesignSync Data Manager User's Guide: Hreffilter Field*. You must specify at least one filter or hreffilter.

Example of Module View Definition File

This example includes five named module view definitions. When you load module views onto the server, you can load all the definitions in the file or name specific module views to load.

Note: Any line with a # character in column 0 is considered a comment and ignored.

```
# 300mm.txt - Filters for the 300mm Chip manufacturing line

{Name RTL Description {Basic RTL view} Filter +*.v,+*.rtl
HrefFilter doc,layout-tools}
# This is a wrapped comment followed by a wrapped\
  view definition, formatted for easy reading.

{Name DOC Description {Basic Documentation view, formatted for
easy reading}

  Filter +*.doc,+*.txt,+*.html,+*.htm,+*.xml,+*.gif,+*.jpg

  HrefFilter doc,images}

#RTL view with filters only, no hreffilters.

{Name RTLNOHREF Description {RTL view only filters} Filter
+*.v,+*.rtl}

#RTL view with hreffilters only, no filters.

{Name RTLNOF Description {RTL view, with only hreffilters}
HrefFilter doc,layout-tools}

#RTL view with no description
```

```
{Name RTLNODESCRIPTION Filter +*.v,+*.rtl HrefFilter doc,layout-
tools}
```

#RTL view with extended include filter and Href Filters.

```
{Name RTLSPECIFICFILES Filter ++|.../RTL/NZ12-
1.VHDL|.../RTL/NZ12-2.VHDL|.../RTL/NZ12-3.VHDL| HrefFilter
doc,layout-tools}
```

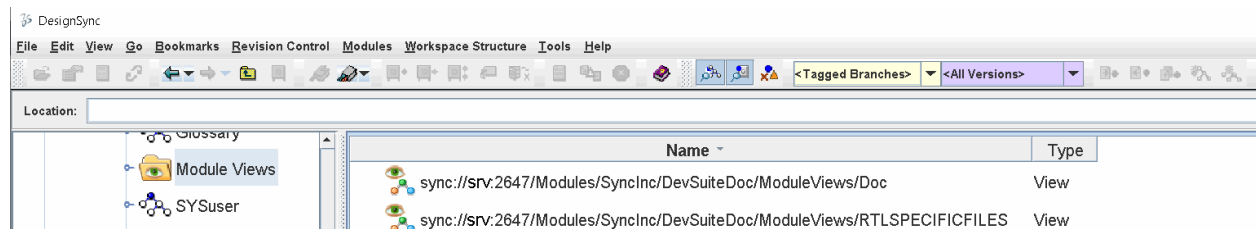
#RTL view with extended include, standard filter and Href Filters.

```
{Name RTLSPECIFICFILES Filter ++|.../RTL/NZ12-
1.VHDL|.../RTL/NZ12-2.VHDL|.../RTL/NZ12-3.VHDL|,+.../*
HrefFilter doc,layout-tools}
```

Displaying the Contents of a Module View

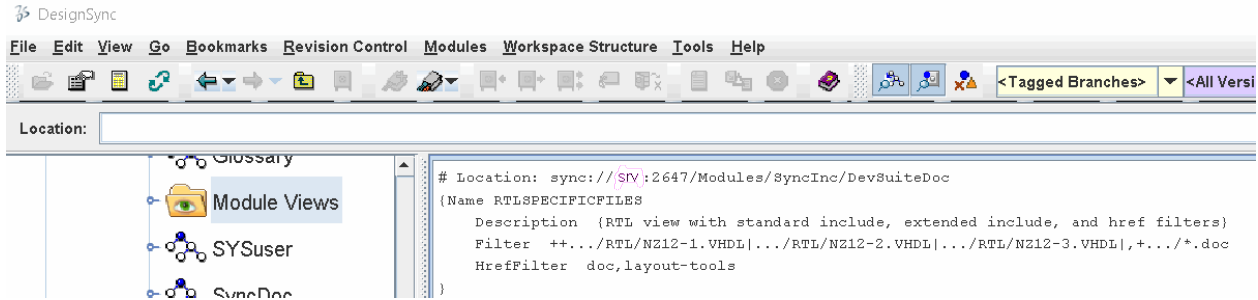
After the module view has been loaded onto the server, you can view the definition in the DesignSync GUI client, by navigating to the location specified as the source of the Module View when the view was put onto the server. All Module Views in that location are stored in the Module Views sub-folder. Module Views are not versioned objects (although you can check in a text file containing module view definitions for history tracking).

When you list the Modules Views, the name is the full path to the Module View, including the View name and the object type is View. This is the same information returned by the `view list` command.



Click on the view name to display the view definition. Note that any comments from the original file are not retained, but each view includes a comment showing the view location. This is the same information returned by the `view get` command.

DesignSync Data Manager Administrator's Guide



Note: Any comments that were in the file are not displayed as part of the view definition, but a comment has been added to the view definition that shows the location on the server. This allows you to easily distinguish between identically named views in different locations.

Related Topics

ENOVIA Synchronicity Command Reference: view list command

ENOVIA Synchronicity Command Reference: view get command

External Modules

Overview of External Modules

External modules are references to projects stored within a different change management system. Using an external module allows you to maintain and use a common DesignSync workspace containing both DesignSync managed data, and externally managed data.

Using DesignSync, you create a hierarchical reference in your DesignSync module to the external module containing the data you want to include in your hierarchy. You can then use DesignSync to maintain an up-to-date copy of the external module data.

Note: In its current form, external modules support the following DesignSync commands for the external CM system:

- populate
- rmmod
- tag
- ls
- showstatus
- showhrefs
- swap replace/restore
- entobj synchronize

Related Topics

DesignSync User's Guide: External Modules

DesignSync User's Guide: Creating a Hierarchical Reference

DesignSync User's Guide: Populating Your Work Area

Defining the External Module

Defining the External Module Interface

Each external change management system requires an interface. The interface is created as set of tcl files. DesignSync uses the files to facilitate communication between the external change management system and the DesignSync client.

The external module interface files are stored in the `$SYNC_CUSTOM_DIR/site/share/client/tcl` directory. Once the files are in the directory, they are available to all DesignSync clients.

The external module interface file

The interface file must contain the following:

- A call to load the ExternalModule package. This package provides the connectivity between the DesignSync clients and external CM system.
- A defined namespace which is used as the external module interface name. This name is used in the external module URL to associate the external module with the interface.

Tip: To easily identify the interface within the `tcl` directory, you should include the namespace name in the Tcl file name.

- A set of procedures that define the change management actions available in the external CM system.

External module package

The external module package call loads the tcl script that interface between DesignSync and the CM system. It is automatically loaded when a DesignSync clients starts. The first line of your custom external module interface should be a call to this package. The call should look like this:

```
package require ExternalModule
```

Important: Do not modify the ExternalModule.tcl file.

External interface name

The namespace name uniquely identifies the procedures that operates with the external code management system. The documentation refers to the namespace name as the interface name for clarity.

You should use a name that is easily identifiable. External type names follow the same naming restrictions as modules. For a list of prohibited characters, see *DesignSync Data Manager User's Guide: URL Syntax*.

The example below uses the interface name "cvs."

```
namespace eval ::ExternalModule::cvs {}
```

External module tcl procedure

The procedure defines the action being taken and provides a wrapper to send information stored in the external module hierarchical reference to the external code management system. The procedure call to receive and evaluate the data is constructed in the following form:

```
proc ::ExternalModule::<interface-name>::<ds-command> <cmd-parameters> <external-module-data> <DesignSync-options> [<external-module-options>]
```

- *interface-name* The name of the interface.
- *ds-command* The name of the DesignSync command.
- *cmd-parameters* An ordered list of command arguments:
- *external-module-data* The *external-data* component of the external module URL. This component includes the identifying information for the data on the external code management system, for example: the configuration name, version number, etc.
- *DesignSync-options* An ordered list of DesignSync command options to apply to the operation. For any unspecified options, DesignSync uses the defined default value.
- *external-module-options* An ordered list of the external change management system options to apply to the operation.

Within the procedure, these values and how they are passed to the external CM system can be further defined.

The interface code may provide output to the user. If the DesignSync command being used supports the `report` option, you can provide the output for a specific report level. For example for brief mode, you might provide only a failure message, for normal mode, a success or failure message, and for `-verbose` mode, a set of connection interfaces

and a list of all the objects affected. In this way, you can provide the user with the appropriate level of information.

Return values and errors

The return values are specific to the command being performed.

If the interface throws a Tcl error, DesignSync considers this a failure to run the command on the external module. DesignSync has a set of error handling behavior for each command. For information on how DesignSync handles incomplete operations, see the documentation for the specific command being performed.

Related Topics

Using DesignSync Commands with External Modules

Using DesignSync Commands with External Modules

Each DesignSync command has specific requirements for command parameters and return values. External module options specified through the `-xtras` option are always passed through the DesignSync system without processing by, but are processed by the external code management system or the Tcl processing script.

DesignSync supports the following commands for external modules:

- Populating An External Module (`populate`)
- Removing an External Module (`rmmod`)
- Tagging an External Module (`Tag`)
- Show object list (`ls`)
- Show Object Status (`showstatus`)
- Show Hierarchical References (`showhrefs`)
- Replacing a Swapped External Module (`Swap Replace`)
- Restoring a Swapped External Module (`Swap Restore`)
- Enterprise Objects Synchronize (`entobj synchronize`)

Populating An External Module (`populate`)

The `populate` command has specific command parameters requirement, and allows you to specify external module options through the `-xtras` options on the command or the `Extras` option on the `populate` dialog.

Command parameters

The command parameter for the `populate` command is an ordered list containing the following information:

- An absolute path (representing the relative path of the hierarchical reference) to the base directory into which the external module must be fetched. This information is required by the system.
- An optional list of object paths to fetch when the user does not want to update the entire external module. The object paths are relative to the base directory. If a list is not included, the entire module is updated.

External module options

The external module options can be provided to the interface through the command line option `-xtras` or the Extras option on the populate dialog.

The Extras option contains a list of command line options to pass to the external code management system. Any options specified with the `-xtras` option are sent verbatim, with no processing by the populate command, to the Tcl script that defines the external code management system.

Return Value

The Tcl handler must return an ordered list with two values. The first value is the number of objects successfully processed and the second value is the number of objects that failed to be processed.

Removing an External Module (rmmod)

The `rmmod` command removes the external module and its contents from your workspace. It does not remove data from the remote module. The `rmmod` command has specific command parameter requirements, and allows you to specify external module options through the `-xtras` options on the command or the Extras option on the `rmmod` dialog.

Command parameters

The command parameter is a single argument containing the base directory of the external module.

External module options

The external module options can be provided to the interface through the command line option `-xtras`.

The Extras option contains a list of command line options to pass to the external code management system. Any options specified with the `-xtras` option are sent verbatim, with no processing by the populate command, to the Tcl script that defines the external code management system.

Return Value

The Tcl handler must return an ordered list with two values. The first value is the number of objects successfully processed and the second value is the number of objects that failed to be processed.

Tagging an External Module (Tag)

The tag command allows tagging of the members within an external module. You can specify external module options through the -xtras options on the command.

Command parameters

The command parameter for the tag command is an ordered list of the following arguments:

1. Tag name to apply to the external module.
2. Server URL, including optional selector. If no selector is specified, DesignSync uses the default.
3. List of object paths to operate on within the external module (optional)..

Note: Each module is tagged non-recursively.

External module options

The external module options can be provided to the interface through the command line option -xtras. Any options specified with the -xtras option are sent verbatim, with no processing by the populate command, to the Tcl script that defines the external code management system. Either the Tcl script or the remote CM system can process these options.

Return Value

The Tcl handler must return an ordered list with two values. The first value is the number of objects successfully processed and the second value is the number of objects that failed to be processed.

Show object list (ls)

The ls command to show the list of the members within an external module. You can specify external module options through the -xtras options on the command.

Command parameters

The command parameter for the ls command is an ordered list of arguments. The first argument is the base directory of the external module, and optional second argument is a list of object paths to operate on within the external module.

External module options

The external module options can be provided to the interface through the command line option `-xtras`. Any options specified with the `-xtras` option are sent verbatim, with no processing by the `ls` command, to the Tcl script that defines the external code management system. Either the Tcl script or the remote CM system can process these options.

Return Value

You have the option to choose how the return value is formatted. If you have specified `-format list`, however, the Tcl script must format the output to match the list format expected for `ls`.

Show Object Status (`showstatus`)

The `showstatus` command shows the status of the members within an external module. It has specific command parameters requirement, and allows you to specify external module options through the `-xtras` options on the command.

Command parameters

The command parameter for the `showstatus` command is a single argument containing the base directory of the external module.

External module options

The external module options can be provided to the interface through the command line option `-xtras`. Any options specified with the `-xtras` option are sent verbatim, with no processing by the `showstatus` command, to the Tcl script that defines the external code management system.

Return Value

The `showstatus` command returns the status of the external module members. You have the option to choose how the list is formatted; using the `-format` option to specify the format of list or text. The `showstatus` list format is a TCL list. If you support this format, you must use the TCL list format as described in *ENOVIA Synchronicity Command Reference*: `showstatus`.

Show Hierarchical References (`showhrefs`)

The `showhrefs` command shows hierarchical references. It has specific command parameters requirement, and allows you to specify external module options through the `-xtras` options on the command.

Hierarchical reference conflicts are reported in the final list output when all of the following conditions are true;

- *conflict* property is set to yes during command processing
- the *-conflicts* option is provided

These properties are not, however; returned in the final list output or included in the conflict summary.

Over-ridden hierarchical references can also be reported in the final output when the *-overridden* option is used with the *-report verbose*.

Command parameters

The command parameter for the *showhrefs* command is a single argument containing either:

- the workspace base directory of the external module
- the sync URL of the external module.

External module options

The external module options can be provided to the interface through the command line option *-xtras*. Any options specified with the *-xtras* option are sent verbatim, with no processing by the *showhrefs* command, to the Tcl script that defines the external code management system.

Return Value

The *showhrefs* command returns the list of hierarchical references. You have the option to choose how the list is formatted; using the *-format* option to specify the format of list or text. The *showhrefs* list format is a TCL list. If you support this format, you must use the TCL list format as described in *ENOVIA Synchronicity Command Reference: showhrefs*.

The following fields must be returned for each module in the external module hierarchy:

- client base directory (*basedir <basedirectory>*)
- relative path (*relpath relpath*)
- sync URL (*url url*)
- type set to External (*type External*)

Any fields not returned will be interpreted as being empty ("").

If a hierarchical reference is in conflict, the returned state of the hierarchical reference is:

- *skipped-conflict* -- not populated because of a module conflict
- *added-conflict* - the module that was populated when modules were discovered to be in conflict.

If a hierarchical reference is overridden, the returned state of the hierarchical reference is:

- skipped-override - when the reference was skipped because it was overridden.
- added-override - when the reference was fetched as the overriding reference.

Replacing a Swapped External Module (Swap Replace)

While the swap replace command can be run on an external module instance to replace one version of the module with another, the command actually runs `rmmod` and `populate` to support the replace. Because of this, the swap replace command itself does not need a Tcl handler.

Command parameters

Not applicable. For more information, see [Populating an External Module \(populate\)](#) or [Removing Data from the Remote Data Source \(rmmod\)](#).

External module options

The external module options can be provided to the interface through the command line option `-xtras`. Because it may be necessary for the `populate` or `rmmod` commands to know they were called from `swap replace`, when `swap replace` is run on an external module, a "swapreplace" option (`-swapreplace`) is automatically added to the `-xtras` options.

The `Extras` option contains a list of command line options to pass to the external code management system. Any options specified with the `-xtras` option are sent verbatim, with no processing by the `swap replace` command, to the Tcl script that defines the external code management system.

Return Value

The Tcl handler should return an ordered list with two values. The first value is the number of objects successfully processed and the second value is the number of objects that failed to be processed.

Restoring a Swapped External Module (Swap Restore)

While the swap restore command can be run on an external module instance to replace the workspace module version of a module with the default workspace version of a module, the command actually runs `rmmod` and `populate` to support the restore. Because of this, the swap restore command itself does not need a Tcl handler.

Command parameters

Not applicable. For more information, see [Populating an External Module \(populate\)](#) or [Removing Data from the Remote Data Source \(rmmod\)](#).

External module options

The external module options can be provided to the interface through the command line option `-xtras`. Because it may be necessary for the `populate` or `rmmod` commands to know they were called from `swap restore`, when `swap restore` is run on an external module a `swaprestore` option (`-swaprestore`) is automatically added to the `-xtras` options.

The `Extras` option contains a list of command line options to pass to the external code management system. Any options specified with the `-xtras` option are sent verbatim, with no processing by the `swap restore` command, to the Tcl script that defines the external code management system.

Return Value

Determined by the commands run, either `populate` or `rmmod`.

Enterprise Objects Synchronize (entobj synchronize)

The `Enterprise Objects Synchronize` command (`entobj synchronize`) is not explicitly run on external module instance. It is implicitly called when the `entobj synchronize` command is run with a specified depth of all on a module hierarchy that includes an external module. In this case, `entobj synchronize` launches the `showhrefs` hook when processing the external module. Because of this, the `entobj synchronize` command itself does not need a Tcl handler.

Command Parameters

Not applicable. For more information, see [Show Hierarchical References](#).

External Module Options

The external module options can be provided to the interface through the command line option `-xtras`.

Any options specified with the `-xtras` option are sent verbatim, with no processing by the `entobj` command, to the Tcl script that defines the external code management system.

Sample External Module Tcl Script

The following is a sample Tcl script showing a `populate` call to the CVS code management system. It is intended as an example only and may not represent the most effective or efficient way to structure your script.

DesignSync Data Manager Administrator's Guide

This example is included in installation in

*`$SYNC_DIR/share/examples/ExampleExternalModuleInterfaces/`. Within this directory are subfolders containing examples for *performce*, *subversion* and *cvs*, each containing the corresponding example handlers. All Tcl files containing the external module Tcl handlers must be located in `$SYNC_CUSTOM_DIR/site/share/client/tcl/` in order to be used.*

```
#####
# Copyright (c) 2010 Dassault Systemes. All rights reserved.
# All External Module tcl scripts must be located in:
#$SYNC_CUSTOM/share/site/share/client/tcl/
#####

# Required call to ExternalModule.tcl

package require ExternalModule

#Defining the package name as cvs. This value becomes the
# external-module-type value.

namespace eval ::ExternalModule::cvs {}

#####
# workspace Directory to fetch the module into.
# external-module-data The external module data
# supplied as part of the external module hierarchical
# reference url. The "sync:///ExternalModules/cvs"
# string is stripped off.
#
# DesignSync-options A list of command line options used
# to call the DesignSync command that initiated the CM system
# interface call. This includes defaults.
#
# external-module-options Optionally, a set of additional
# options specific to the CM system operation can be
# supplied. This is anything that appears in the command
# after '-xtras'.
#####

proc ::ExternalModule::cvs::populate {workspace external-module-
data DesignSync-options external-module-options} {

    global env

#Create a running log in the user's home directory.

    set pass 0
```

```
set fail 0

set logfile ~/debug_log

if [file exists $logfile] {
    set chan [open $logfile a]
} else {
    set chan [open $logfile w]
}

puts $chan "workspace $workspace\n"
puts $chan "data $external-module-data\n"
puts $chan "DesignSync-options $ds_options\n"
puts $chan "external-module-options $cvs_options\n"
close $chan

set components [split $data :]
set host [lindex $components 0]
set username [lindex $components 1]
set module [lindex $components 2]

set env(CVSRROOT)
:ext:$username@$host:/home/$username/cvsroot

set env(CVS_RSH) ssh

set old_pwd [pwd]

cd [file dirname $workspace]

hrPuts stdout "pwd is [pwd]\n"

exec cvs checkout -d [file tail $workspace] $module

set retval [list $pass $fail]
```

DesignSync Data Manager Administrator's Guide

```
    cd $old_pwd  
    return $retval  
}
```


Upload Archive

Upload Archive

The Upload Archive interface allows you to upload or update a tar or gzipped tar archive to DesignSync in an efficient manner so that, instead of replacing the archive with the next version, DesignSync updates only the elements within the archive file that have changed from the previous version.

By performing a change (delta) calculation and only checking in the changed object set, DesignSync provides both improved speed during checkin and checkout and reduces the amount of disk space required for storing the IP.

To access the **Upload Archive** panel, select **Upload Archive** in the **Upload** section of the **DesignSync** menu in the DesignSync Web user interface. Click **Submit** when you have made your changes.

Note: In order for the Upload section to appear in the DesignSync Web interface, you must have TransferFile access. For more information on TransferFile access, see *Access Control Guide: Access Controls for Upload*.

Click on the image for more information about each field.

Upload Archive

*** Denotes a required field**

File*	<input type="button" value="Browse..."/> finalIP.tar
Collections exist	<input type="checkbox"/>
Vault Path*	<input type="text" value="sync://ABCo.com:2647"/> <input type="button" value="Browse..."/>
Temporary Directory	<input type="text"/>
Create new vault	<input checked="" type="checkbox"/>
Branch	<input type="text"/> <input type="button" value="Browse..."/>
Tag	<input type="text"/> <input type="button" value="Browse..."/>
Comment	<input type="text" value="New IP"/>

Field Descriptions

These sections explain the fields on the panel.

File

Select the Browse button to select a tar or gzipped tar archive to upload or update on the server. The archive must be visible to the web browser. The file extension for the tar file must be either tar or .tgz in order for DesignSync to recognize the file.

Collections exist

Check this box when the compressed package includes collections objects. When processing collection objects, DesignSync upload does not use reference mode. This can be slower, but handles the objects correctly. If there are no collections, leave the box unchecked which tells DesignSync to use reference mode and speeds up the operation.

Vault Path

Enter or Browse to the location of the module or vault that contains the uploaded or updated archive

Note: You must be using the panel on the same server to which you are uploading the archive.

Temporary Directory

If you do not want to use the default temporary directory, you can specify a server location in the form of an absolute path, to expand the archive before running the delta comparison and checking in the changes. This setting overrides any default values set.

If no value is set, DesignSync users the following order to determine where to unpack the archive:

1. If the registry key for Temporary Directory for Upload is specified, that value is used as the tmp directory.
2. If the SYNC_TMP_DIR environment variable is set on the server machine, that value is used as the tmp directory.
3. If the TMPDIR environment variable is set on the server machine, that value is used as the tmp directory.
4. If no other values are set, DesignSync uses the /tmp directory on the server machine.

Create new vault

Select this checkbox if this is an initial upload.

Branch

Enter or Browse to select the Branch. If no Branch is selected, the default, Trunk, is used. If you use the Browse button, you must be using the panel on the same server to which you are uploading the archive.

You cannot specify a branch for the initial upload. The initial upload is always done on the Trunk branch.

Tag

Applies the specified tag to the data being uploaded. This tag can be used to get the data later for example, when populating the archive into a workspace. If you use the Browse button, you must be using the panel on the same server to which you are uploading the archive.

If the tag already exists it moves to the new version.

Note: An automatically generated tag, in the form Archive.<#> is also applied to the data being imported, where the initial value of # is 1, and then the number is incremented as archive is updated.

Comment

Stores a comment with the uploaded IP. If a comment is required for checkin operations, then a comment is required for this operation as well, and must conform to the minimum comment length, if a minimum comment length is required.

Performance Optimizations

Performance Optimization Overview

DesignSync is a client/server architecture whose performance depends heavily on your system and network configuration. Disk, network, and CPU speed, the amount of virtual memory (RAM plus swap space) on your server machine, and other parameters all directly impact DesignSync performance.

The most critical configuration factors affecting performance are where the SyncServer runs and where its associated vaults are stored. Always try to put your vaults on disks that are local (not NFS-mounted) to the server. DesignSync is disk I/O intensive, so a fast disk that is local to the machine running the SyncServer greatly improves overall throughput and performance.

By default, DesignSync uses a multithreading optimization to speed up check-in, check-out, and populate operations. The following are additional optimizations that may improve your DesignSync performance:

UNIX Only

- Changing the DesignSync temporary directory (specify a tmpfs-mounted directory to speed up file transfers between client and server)
- Disable DesignSync recognition of Cadence Design Systems and Synopsys collections (when your project does not use Cadence or Synopsys data). Doing so improves DesignSync performance by eliminating unnecessary processing. You can enable this performance option through SyncAdmin's Third Party Integration Options. **Note:** Recognition of Cadence or Synopsys objects may already be disabled, depending on how DesignSync was installed and configured.
- Fetching files from the mirror or cache (instead of fetching from the vault during 'co -lock', 'co -get', 'populate -lock', and 'populate -get' operations).
- Enabling link-in for large files.

UNIX and Windows

- Turning off compression (when transferring large files over a high speed network)
- Turning off keyword expansion
- Turning on direct fetch.

Client-Side Optimization

Fetching Files from the Mirror or Cache

Note: This optimization is available on UNIX platforms only.

DesignSync can retrieve an object for editing ('`co -lock`', '`populate -lock`') or as read-only ('`co -get`', '`populate -get`') from the vault or from a remote mirror or cache. By default, DesignSync retrieves objects from the remote mirror or cache whenever possible. Typically, performance can improve if DesignSync retrieves the object from the mirror or cache directory whenever possible. The mirror and cache are always on the user's LAN, whereas the vault requires a network hop (except for client vaults). In cases where you have very fast access to the server, but the mirror or cache is on an nfs-mounted disk, always fetching from the server may be faster.

Because the optimization of fetching from the mirror or cache is configuration dependent and can even vary from project to project for a given user, the fetch behavior is configurable at the site, user, and command-line levels.

Users can override the site-wide setting from the **Tools=>Options=>General=>Command Defaults** dialog from the DesignSync graphical interface. Users can also specify the `-from` command-line option to the `co` command and the `populate` command to control the fetch location on a per-command basis. The command-line option overrides both the user and site registry settings. When this optimization is enabled (the default), DesignSync looks for the requested version in the mirror, then the cache, then the vault.

Limitations:

- This optimization applies to the '`co -lock`', '`co -get`', '`populate -lock`', and '`populate -get`' commands.
- This optimization is not applied if the user specifies the '`-key`' option to the `co` or `populate` commands. Revision-control key behavior can only be controlled by retrieving from the vault.
- DesignSync retrieves a version from the mirror only if the workspace selector matches the mirror selector and the '`-from local`' option, which is the default, is used.

Fetching Files Directly to Your Work Area

You can optimize DesignSync performance by fetching objects directly into your work area.

By default, DesignSync writes a fetched file to the `.SYNC` metadata directory under a temporary name. When the entire file is successfully transferred from the vault, DesignSync renames the file and saves it in the work area. This approach ensures that no corrupted information is written into the work area. In the event of a network or disk problem during the transfer, the fetch is rolled back and no data is written to the work area.

If you opt to fetch files directly, DesignSync transfers the files into your work area without copying and renaming them. Thus populate, check-out, and other fetch operations perform more quickly. However, if your system is unstable, you risk data corruption in your work area when you fetch files directly.

Your administrator can enable direct fetch for all users on your LAN using the SyncAdmin tool. See Performance Options for more information.

To enable direct fetch for your local work area:

1. From the DesignSync menu, select the **Tools=>Options** to open SyncAdmin.
2. From the tree menu on the left, choose **General => Performance**.
3. On the Performance dialog box, select **Enable direct fetch**.

Related Topics

[ENOVIA Synchronicity Administrator Tool Overview](#)

Multithreading Optimization

DesignSync takes advantage of parallel process threads to speed up check-in, check-out, and populate operations. DesignSync uses multiple client threads that communicate with multiple server children, thus reducing the I/O client/server bottleneck. While one client thread is waiting for a server operation to complete, another client thread can be performing client-side operations. Likewise, while one server child is waiting for a client thread to respond, another server child can be operating on the server vault.

In DesignSync's multithreading paradigm, multiple threads operate on multiple objects in directories simultaneously. To ensure the integrity of revision control operations, a single thread carries out all stages of a revision control operation on an object.

As a system administrator, you can control aspects of the multithreading optimization. You can change the number of client threads DesignSync uses to carry out revision control operations or you can turn off multithreading entirely. You might find that in diagnosing network problems, for example, you might want to turn off multithreading to isolate the problem. Then, when you have resolved the problem, you can resume the multithreading optimization. You use the Performance tab of the **SyncAdmin** tool to fine-tune the multithreading optimization.

Limitation on Temporary Filenames Used by Triggers

If you use temporary filenames to communicate information between preObject and postObject triggers, you need to make sure your filenames are unique. With the multithreading optimization, multiple preObject triggers might be run before any

postObject triggers are run. For example, a preObject trigger which writes to a temporary file, <TMP_DIR>/tempfile, might not work, because the data will be written multiple times before any postObject trigger gets a chance to read it.

Turning Off Compression

By default, DesignSync compresses data before it is transferred between client and server. Compression generally improves the speed of data transfers by reducing the size of the files being transferred. In some circumstances, disabling compression may improve performance. For example, if your data consists of large binary files and your network is fast, the time required to compress the data may exceed the time saved by transferring smaller files. Note that compression also increases the security of your data transfers.

You, the LAN administrator, can turn off compression for all users on your LAN using the SyncAdmin tool. Compression can also be disabled on a per-user basis from the **Tools=>Options=>General** dialog box from the DesignSync graphical interface.

Related Topics

[ENOVIA Synchronicity Administrator Tool Overview](#)

Using Delta Transfer Hooks to Tune Client/Server Communications

DesignSync features the option to transfer only the changes between subsequent file versions to provide more efficient file transfers over high latency/poor bandwidth networks thereby saving engineering time and network resources

This feature, called delta transfer, is a transfer compression mechanism that relies on computing and transferring the difference between the file that needs to be sent and another file with a similar content (base file). On the receiving end, the original file is restored by applying the difference to a copy of the base file. In cases where differences are small, this mechanism can result in a drastic reduction of the transfer size. For more information on how to use delta transfer, see [Enable file delta transfer between client and server](#).

You can enable or disable using delta transfer as the default transfer mechanism for client/server interaction with SyncAdmin using the Communications tab. You can also use client triggers. DesignSync provides two client triggers, `TransferHook`, to control the transfer mechanism on a per-server basis, and `TransferDeltaHook`, to control the transfer mechanism on a per-object basis.

Using the TransferHook client trigger

DesignSync Data Manager Administrator's Guide

The `TransferHook` client trigger controls communications between the specific client calling the hook and the server(s) defined within the trigger. This allows you to restrict delta transfer to servers that benefit from the delta transfer.

The `TransferHook` trigger also allows you to set the buffer size to increase bandwidth use according to the available network.

DesignSync allows the following output values for `deltaTransfer` and `compressedTransfer`:

- `ALLOW` - allow the operation to use the specified transfer mechanism.
- `DENY` - do not allow the operation to use the specified transfer mechanism.
- `UNKNOWN` - use the default transfer mechanism.

Example of a `TransferHook` client trigger

This example shows a `transferHook` trigger. It uses an event with two input properties: `serverName` and `serverIP`. It may have one or more output properties: `deltaTransfer`, `compressedTransfer`, and `socketBufferSize`.

```
trigger create wanHook -require type transferHook -tcl_script {
    if {[string first $serverIP "10.1."] != 0} {
        # Server is on the WAN
        set deltaTransfer ALLOW
        set compressedTransfer DENY
        # Socket buffer size in KB.
        set socketBufferSize 512
    }
}
```

Using the `TransferDeltaHook` client trigger

The `TransferDeltaHook` client trigger examines all the objects being operated on by the command that called the trigger, so can be expensive in terms of time to run, however it can save processing time on the server by excluding objects from attempting to use the delta transfer feature. Using `TransferDeltaHook` you can exclude objects that meet any of the following criteria:

- compressed files
- files below a certain size (which therefore to not reap the benefits of delta transfer)

The `TransferDeltaHook` is only called when the operation is being performed on a server that has delta transfer enabled.

Note: The `TransferHook` must set an `ENABLE` value on `DeltaTransfer` in order to use the `TransferDeltaHook` trigger on the desired objects.

Example of a `TransferDeltaHook` client trigger

This example shows a transfer Delta Hook trigger and uses an event with one input property `objPath`, and one output property, `deltaTransfer` with values **ALLOW**, **DENY**, **UNKNOWN**. To enable delta transfer for a file the hook must return **ALLOW**. This hook is invoked per file and disables delta transfer for any files compressed with gnu zip or tar.

```
trigger create deltaHook -require type transferDeltaHook -tcl_script {
if {[string equal [file extension $objPath] ".gz"] |
    [string equal [file extension $biotopes] ".tar"]} {
    # Compressed file
    set deltaTransfer DENY
} else {
    set deltaTransfer ALLOW
}
}
```

Related Topics

Communications

Data Compression

Delta Transfer Compression

Turning Off Keyword Expansion

You can optimize DesignSync's performance by turning off keyword expansion if you do not use keywords in your files. By default, DesignSync expands all keywords that are used in your files.

Important: You should use this optimization only if you do not use keys in your files.

When you turn off keyword expansion, the "key expansion" and "retain timestamp" options are ignored when you check in and check out files under revision control.

When you do a fetch operation after turning off keyword expansion, the following commands will skip the fetching stage of the operation:

- `co -lock` (only works with `-from local`)
- `co -get` (only works with `-from local`)
- `populate -lock` (only works with `-from local`)
- `populate -get` (only works with `-from local`)
- `ci -keep`
- `ci -lock`

The latest version of files that already exist in your work area from a previous fetch operation remains unchanged, and any keywords in these files are still expanded. The keywords are not expanded in any new or updated files that you fetch.

To turn off keyword expansion, follow these steps:

1. Start **SyncAdmin**.
2. Navigate to the **Performance Tab**.
3. Turn off **Enable Keyword Expansion**.
4. Click **Apply**.
5. Restart DesignSync.

Related Topics

DesignSync Data Management User's Guide: Using Revision Control Keywords

Linking Large Files

Linking Large Files

If your DesignSync installation meets certain criteria, you can speed up the checkin of large files by creating hardlinks to the files in your DesignSync vault. This performance optimization, called Link-In, is possible only when:

- The server and its clients are running DesignSync 4.1 or later.
- Users' workspaces are on the same file system as the DesignSync vault.
- You use Copy Vault as the check-in method. (See About Vault Types for details.)

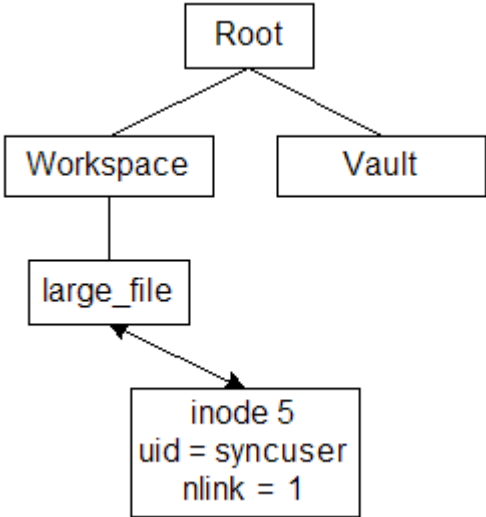
Link-In is a client setting and works for all servers and vaults accessed by the client.

This feature is available only on UNIX systems.

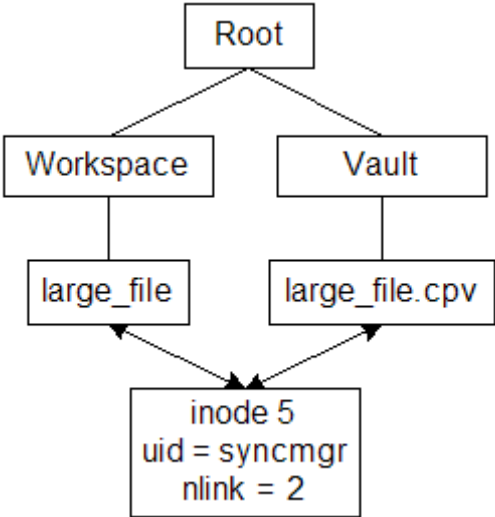
How Link-In Works

When a user checks in a large file and Link-In is enabled, DesignSync creates a hardlink in the vault instead of physically copying the file. At the same time, ownership of the large file is transferred from the user to the syncmgr account. (See Overview of DesignSync Administration for details on this account.)

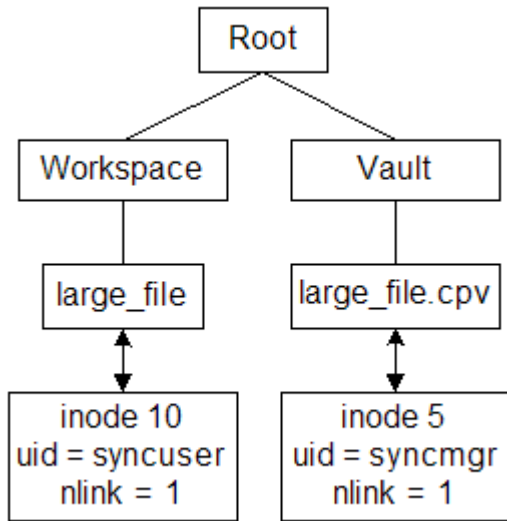
Link-In utilizes UNIX inodes, which store the information about the files in the system. If a user's workspace includes a large unmanaged file, the inode for this file includes information on the owner and the number of links to the file:



When the user checks in this file and Link-In is enabled, DesignSync creates a hardlink in the vault to the file's inode and changes the ownership from the user to the syncmgr. Both the user's workspace and the vault are linked to the same inode:



If the user attempts to edit the file, new inode information is created for the edited file and the hardlink from the vault still points to the original inode:



See Setting Up Link-In for details on how to set up Link-In on your system.

Setting Up Link-In

Link-In is a performance optimization that speeds up the checkin of large files by creating hardlinks in the vault. (See Linking Large Files for details on how this feature works.)

To use Link-In on your UNIX system, your DesignSync installation must meet these requirements:

- The server and its clients must be running DesignSync 4.1 or later.
- Your DesignSync vault and your users' workspaces must be on the same file system.
- You must use Copy Vault as the check-in method. See Setting Vault Types for details.

In addition, your system administrator must edit, compile, and install the C program `sync_chown` to change ownership from the user to the `syncmgr` account when large files are checked in. See Installing `sync_chown` for details.

You cannot specify the `-keep` option for the `ci` command when using Link-In.

You can use Link-In for Cadence collections. When checking the file sizes and linking Cadence collection objects:

- The size of a collection is the sum of the sizes of the member files.
- Only member files with the Copy Vault vault type are linked in.
- Link-In is enabled only if no modified collection member has the Zipped Copy Vault vault type.

Configuring Link-In

To configure Link-In, use the SyncAdmin **General** => **Performance** tab page. (See Performance Options for details.) The settings on this page let you specify the following aspects of Link-In behavior:

- Whether DesignSync performs disk space checks before checking in large files.

The **Check disk space** option to SyncAdmin determines whether DesignSync checks the vault for available space before checking in large files. You can enable this option without enabling Link-In. For Link-In, this check also makes sure that the workspace is on the same file system as the vault.

- The minimum size for large files.

The default minimum size for a large file is 256 megabytes. DesignSync does not perform disk space checks on files smaller than this size. You can use the field on the Performance tab page to change the minimum size of a large file.

The disk space check requires DesignSync to perform a round-trip query of the server vault. If you set the minimum file size too small, the time required to query the server may offset any performance gains.

- Whether Link-In is enabled.

The **Enable Link-In** option to SyncAdmin lets you turn on the Link-In functionality. If you select this option, you also must enable the **Check disk space** option.

- Whether ownership of Milkyway files is changed to syncmgr.

IMPORTANT: While the Synopsys MilkyWay integration options are still visible, the integration has been deprecated.

Installing `sync_chown`

To use Link-In on your UNIX system, your system administrator must edit, compile, and run the C program `sync_chown`. This program changes ownership from the user to the `syncmgr` account when large files are checked in.

The `sync_chown` program operates only on files that:

- Are on the same file system as the `sync_chown` program.
- Are below the root directory for the `sync_chown` program.
- Are regular files and are not symbolic links.
- Are owned by the user running the `sync_chown` program.
- Do not have set-uid, set-gid, or sticky bits.

These restrictions ensure that you can run `sync_chown` without compromising system security.

To install `sync_chown`:

1. Create a directory to use as your `SYNC_CHOWN_ROOT`. This directory:
 - Must be a directory on the same file system as the user's workspace and the server vault. This restriction is required to create hard links.
 - Must be a parent directory of the user's workspace and the server vault.

You do not need to place the entire `server_vault` on this file system. You can create a symbolic link from the `server_vault/Projects/` directory to a directory under your `SYNC_CHOWN_ROOT` directory.

- Must reside on a file system that is mounted with the `suid` option so that `setuid` root programs are allowed.
2. Copy the example program file into your `SYNC_CHOWN_ROOT` directory from:

```
$SYNC_DIR/share/examples/sync_chown/sync_chown.c
```

3. Edit the customization section of the example `sync_chown.c` program and change the following value:

```
#define SYNC_CHOWN_ROOT "@sync_chown_root@"
```

In place of `"@sync_chown_root@"`, substitute the directory you want to use as `SYNC_CHOWN_ROOT`. For example:

```
#define SYNC_CHOWN_ROOT "/home/apps/data"
```

Installing `sync_chown` in the `SYNC_CHOWN_ROOT` directory ensures that security restrictions are enforced.

4. Compile `sync_chown.c` using a C compiler:

```
cc -o sync_chown sync_chown.c
```

5. Change the ownership of the `sync_chown.c` file to root and make it setuid. As root, execute these commands:

```
chown root sync_chown
```

```
chmod u+s sync_chown
```

Repeat this procedure to install `sync_chown` for other server vault file systems.

Server-Side Optimizations

Changing the DesignSync Temporary Directory

Note: For the Windows environment, use the `TEMP` variable that is set up during the Windows install. To change the `TEMP` directory install location, please refer to Windows help.

For best performance, locate the DesignSync temporary directory on a locally mounted temporary file system (commonly called tempfs), not a disk-based file system. A locally mounted directory is mapped to virtual memory (RAM and swap space), which means access is much faster than the disk I/O associated with a disk-based file system.

Performance optimization takes place when a file is transferred between a DesignSync client and a SyncServer (typically during checkin, checkout, and populate operations). During these operations, the SyncServer stores the file in a temporary directory until the operation can be completed safely. For example, when you checkout a version of a file, the server:

1. Extracts the version from the vault file
2. Places the version file in the temporary directory
3. Transfers the version file to the client
4. Removes the version file from the temporary directory

Note: Some operations do not use the temporary directory. These operations instead fetch objects by streaming from the vault. These operations are:

- Populate and checkout operations that fetch the Latest version of an object. (However, populate and checkout operations that fetch versions earlier than Latest still use the temporary directory.)
- Populate and checkout operations that request a lock on the object.
- Checkin operations.

The `/tmp` directory is the typical temporary directory. You can identify local file systems by running the `mount` command.

There are issues to be aware of when using a locally mounted directory:

- The file system must be mounted locally.
- The maximum file size that the file system can handle is determined by your system's virtual memory. If a file cannot fit into RAM, it is swapped out. Therefore, you must ensure that your virtual memory can accommodate your largest revision-controlled file.
- The maximum number of files that a locally-mounted file system can handle is calculated based on the physical memory (RAM) of the machine and not the size of the swap space. You cannot increase this limit by adding swap space.

Note: If DesignSync cannot write files to the temporary directory for any reason (the directory is full, or protections are incorrect), DesignSync returns an error message indicating that the file transfer can't take place:

- For checkins, the error message is: Cannot send a file to server. Check that the file exists and, has the correct UNIX file permissions.
- For checkouts/populates, the error message is: Cannot receive a file from the server. Check the disk space on the server, especially `TMPDIR`.

How to Change the Temporary Directory Location

During installation, you were prompted for a temporary directory (which also set up `TMPDIR`). If after installing DesignSync, you need to change the DesignSync temporary directory for a given SyncServer, rerun `sync_setup`, choose "Configure a server", and provide a new temporary directory when `sync_setup` prompts for the temporary directory. You can make this change without further affecting the SyncServer's configuration.

Note: The temporary directory `<TMPDIR>` that you specify must be mounted locally.

Creating Custom Utilities

Creating Custom Utilities

If you have common or repetitive tasks consisting of a sequence of DesignSync and/or shell commands, you can bundle them into a utility. That utility allows you to use a single command to run the defined operations and to expose that to the user-base so other users can take advantage of it.

In order to allow others to find and use the utility, you can document and publish it. The publishing process creates a list of the utilities in an HTML page for easy browsing.

DesignSync provided Utilities

DesignSync provides a set of utilities that contain simple functionality. All of the DesignSync provided utilities follow the guidelines described in Anatomy of a Custom Utility for content and design. The utilities are composed of three basic files located in the following locations:

- The shell script - `-$SYNC_DIR/share/bin`
- The TCL script - `$SYNC_DIR/share/client/tcl`
- The generated documentation - `$SYNC_DIR/share/content/util/doc`

The documentation is automatically generated as part of the client installation.

Note: All the examples in this topic are based on the DesignSync provided utilities.

Creating and Defining a Custom Utility

Anatomy of a Custom Utility

Creating the Shell Script Wrapper

Creating the TCL Code for the Utility

Creating the Documentation

Anatomy of a Custom Utility

A custom utility generally consists of the following files:

- A shell script wrapper file, (Bourne shell (sh) is recommended, but not required), which allows the utility to be run directly from a UNIX shell. Bourne shell is recommended because it is available on all supported UNIX

platforms. If your site has a different preferred scripting language, however, you may use it. The file must be located in the site or enterprise binaries location (\$SYNC_CUSTOM_DIR/site/share/bin or \$SYNC_ENT_CUSTOM/share/bin)

Notes:

- All DesignSync supplied utilities are in Bourne shell. The DesignSync provided utilities are located in \$SYNC_DIR/share/bin)
- In order to run the utility without typing the full path name, the user's path variable must contain the path to the custom site or enterprise binaries location. For information on setting the enterprise path, see Using Environment Variables.
- A TCL file containing the utility code. The file must be located in the site or enterprise TCL location.
(\$SYNC_CUSTOM_DIR/site/share/client/tcl or \$SYNC_ENT_CUSTOM/share/client/tcl).
- An HTML documentation file which documents the utility for the users. The TCL file contains the source for the generated HTML. The HTML is created by running the mkutildoc utility. The file must be located in the site or enterprise documentation location
(\$SYNC_CUSTOM_DIR/site/share/content/doc or \$SYNC_ENT_CUSTOM/share/content/doc)

Creating the Shell Script Wrapper

In most cases, the shell script will be a very simple wrapper that invokes a stlcl client and calls the TCL program containing the actual code for the utility. You can use the shell script to accept arguments to pass to the TCL code.

The shell script must be stored with the custom binaries in the location for custom site or enterprise binaries. (\$SYNC_CUSTOM_DIR/site/share/bin or \$SYNC_ENT_CUSTOM/share/bin)

Tip: To provide easy utilities access, include the custom binaries areas in the users' path definitions.

This simple example shows how the DesignSync provided utility, extractmodule, takes a single argument, a server URL and passes it to the underlying TCL utility.

```
#!/bin/sh
#
#####
#####
# (c) Dassault Systemes, 1993 - 2013. All rights reserved.
#####
```

```
#####  
#  
# Extract the contents of a module version (no metadata)  
#  
# \  
  
exec stclc -batch -quiet -- "$0" ${1+"$@"}  
package require shareutil::extractmodule  
eval extractmodule $::argv  
exit
```

Creating the TCL Code for the Utility

The TCL file contains the code for your utility and the documentation for your utility, in the form described in [Creating the Documentation](#). The file must be located in the site or enterprise TCL location. (`$SYNC_CUSTOM_DIR/site/share/client/tcl` or `$SYNC_ENT_CUSTOM/share/client/tcl`). The utility should include the 'shareutil' package in order to take advantage of the ability to be exposed to the users. This code fragment includes the TCL code that is used to incorporate command line version of the documentation into the utility so that if you were type "command -usage" or "command -help," you would see the appropriate documentation on the command line.

```
#  
#####  
#####  
# (c) Dassault Systemes, 1993 - 2013. All rights reserved.  
#####  
#####  
#  
## all utilities require the 'shareutil' package.  
package require shareutil  
# indicate the package that is being provided here  
package provide shareutil::extractmodule 1.0  
# declare the namespace and the required procs  
  
namespace eval ::shareutil::extractmodule {  
  
# DOC VARIABLE DEFINITIONS  
...  
}  
  
# var to keep track of all the subfolders we find  
variable dirlist ""  
# utility proc to find all the .SYNC folders in a given
```

```

hierarchy
    proc findDirs {d} {
        variable dirlist
        foreach sd [glob -nocomplain $d/*. * $d/* ] {
            if { [file isdirectory $sd] } {
                if { [file tail $sd] != "." && [file tail
$sd] != ".." && [file tail $sd] != ".SYNC" } {
                    lappend dirlist $sd
                    # and recurse
                    findDirs $sd
                }
            }
        }
    }
}
# check arg
proc go {args} {
    variable SHORT
    variable USAGE
    variable EXAMPLE
    variable DESCRIPTION
    variable dirlist
    if { [llength $args] != 1 } {
        puts $USAGE
        return
    }

    # process help/usage/description
    set DOC [list "USAGE" $USAGE "SHORT" $SHORT
"EXAMPLE" $EXAMPLE "DESCRIPTION" $DESCRIPTION]
    if { [::shareutil::ProcessHelp $DOC $args] } {
        return
    }
}

...

```

Creating the Documentation

The documentation is defined within the TCL file and then processed by the `mkdocutil` that creates the HTML page and adds the utilities processed to the index. During the namespace and process section of the TCL process. The documentation is divided into key sections.

USAGE

Command syntax and usage showing what options and arguments the command accepts and what options are required. Conventionally, DesignSync shows variable input in `<angle brackets>` and optional options or arguments with `[square brackets]`.

SHORT

Short, preferably one line, description of what the utility does. This description appears in the auto-generated index file next to the utility name.

DESCRIPTION

Full description, including anything that the user might need to know in order to use this utility. This section can be used to explain, in detail, the arguments and options for the command.

EXAMPLE

Examples of how you use the utility. This can include sample input and output.

This simple example shows the fragment of the TCL code containing the documentation definition.

```
# define the help vars
# usage: simply state the site/enterprise as the optional args
    variable USAGE {
        mkutildoc [-site | -enterprise] [-force] [-help] [-usage]
[-description] [-example]
    }
# the short description
    variable SHORT {
        update index files and generate documentation for the
utilities.
    }

# a useful example
    variable EXAMPLE {
        This example shows updating the utilities stored in the
custom site section only.
        This command updates both the documentation available for
the command and the index page containing a link to the
documentation.
        stcl> mkutildoc -site
        or
        % $SYNC_DIR/share/bin/mkutildoc -site
        This example updates all the utilities, including the ones
provided with the distribution, as well as the site and
enterprise utilities. This command updates both the
documentation available for the command and the index page
containing a link to the documentation.
        stcl> $SYNC_DIR/share/bin/mkutildoc
        or
        % $SYNC_DIR/share/bin/mkutildoc
    }

# the full description
```

```
variable DESCRIPTION {
```

```
mkutildoc updates the index files and generates the html
documentation for the utilities.
```

```
    This updates the following files:
```

- o pkgIndex
- o tclIndex
- o html documentation

```
    The utility can be used to update the utility section
for all areas containing utilities, including the Dassault
Systems provided utilities, or restricted to the site or
enterprise areas.
```

```
    The command takes the following arguments:
```

```
-site | -enterprise : The -site option is used to restrict
the command operation to the site-specific custom utilities
area, $SYNC_CUSTOM_DIR/site/share/client/tcl.
```

```
The -enterprise option is used to restrict the command
operation to the enterprise-wide custom utilities area,
$SYNC_ENT_CUSTOM/share/client/tcl
```

```
These two options are mutually exclusive. If both options
are specified, they are silently ignored and the command
updates all utilities.
```

```
-force : Update the documentation even if there are no
changed detected.
```

```
-help : Shows the complete help for the command.
```

```
-usage : Shows only the command syntax (USAGE section)
```

```
-description : Shows only the long description (DESCRIPTION
section)
```

```
-example : Shows only the examples (EXAMPLE section)
```

```
}
```

Related Topics

Publishing Custom Utilities

Publishing Custom Utilities

After you have created the TCL file and the shell script wrapped to run the utility, you should publish the utility so that users can find it. A link to the DesignSync provided utilities is available through the main doc index page. There are separate index pages for the DesignSync-provided utilities, the site-wide utilities, and the enterprise-wide utilities. When you run the procedure to update the utilities, it updates these index pages and creates an HTML page with the documentation extracted from the TCL file.

Publishing the Custom Utilities

One of the DesignSync provided utilities, `mkutildoc`, generates the following:

- For each utility, one HTML file containing formatted documentation from the source in the TCL utility file.
- For each location (DesignSync-provided, site, and enterprise) one HTML list, called `index.html`, containing an alphabetized list of the names and short descriptions of all the utilities in that location.

The HTML doc and the index pages are created in the following locations:

- DesignSync-provided utilities documentation:
`$SYNC_DIR/share/content/util/doc/index.html`
Note: This documentation is linked directly in the Documentation product index page.
- Site-wide utilities documentation:
`$SYNC_CUSTOM_DIR/site/share/content/util/doc/index.html`
- Enterprise-wide utilities documentation:
`$SYNC_CUSTOM_DIR/enterprise/share/content/util/doc/index.html`

The files are generated into the doc area for the respective locations:

- DesignSync provided utilities: `$SYNC_DIR/share/content/util/doc`
- Site-wide utilities: `$SYNC_CUSTOM_DIR/site/share/content/util/doc`
- Enterprise-wide utilities:
`$SYNC_CUSTOM_DIR/enterprise/share/content/util/doc/`

The `mkutildoc` uses a set of templates to transform the documentation variables in the TCL script into an HTML formatted document. These templates are located in `$SYNC_DIR/share/content/util/doc`.

Tip: To provide easy utilities access to the users, include the binaries directory where the utilities are stored (`$SYNC_CUSTOM/site/share/bin` or `$SYNC_ENT_CUSTOM/share/bin`) in the users' path definition.

Format of the Generated Index Files

The index is an alphabetically sorted list of all the utilities at that location (DesignSync-provided, site-wide, or enterprise-wide). For each utility, the index lists the name of the utility and the short description. If no short description is provided in the TCL script, this section is blank for that utility.

Tip: To make it easier for users to find the utilities, you can create a custom version of the template, `utilindex.template`, which contains links to the utilities locations you are using. The custom template should be in the utilities doc directory for that location: For example, you could add something like this to the site-wide index template.

```
<h3>Other Utilities</h3>

<ul>

<li><a href="/syncnt/content/util/doc/index.html">Enterprise-
Specific Utilities</li>

<li><a href="/syncinc/util/doc/index.html">DesignSync Provided
Utilities</li>

</ul>
```

Note: The Apache server that supports the DesignSync web interface provides aliases to refer to user-available locations within the DesignSync installation. These aliases can be used to refer to the utilities locations:

- DesignSync-provided utilities location:
`/syncinc/util/doc/index.html`
- Site-specific utilities location:
`/syncsite/content/util/doc/index.html`
- Enterprise-specific utilities location:
`/syncnt/content/util/doc/index.html`

You can also use a relative path to link to the other index files.

Important: If you modify the index template, you must remove the generated index files before you rerun the `mkutildoc` utility.

Format of the Generated HTML Files

DesignSync Data Manager Administrator's Guide

The script that creates the HTML file processes each section heading as a level 4 (<h4>) heading. The text of each section is displayed as a code block within the section.

At the top of the file is a link back to the DesignSync-provided index. Within the template,

Note: The documentation itself is created by placing the information inside variables defined in the TCL script for the utility. For more information on creating custom TCL scripts, see [Creating the Tcl code for the utility](#).

Important: If you modify the doc template, you must remove the generated HTML files before you rerun the `mkutildoc` utility.

Related Topics

[Creating Custom Utilities](#)

[DesignSync Utilities Documentation \(for DesignSync-provided utilities\)](#)

Registry Files

Overview of Registry Files

DesignSync configuration information is stored in registry files. The DesignSync tools automatically save configuration settings in the appropriate registry file:

- DesignSync Web UI's Administer Server panel writes to the server's port registry file.
- The SyncAdmin (DesignSync Administrator) tool can write to the site-wide registry file, a project registry file, or any specified registry file.
- When invoked via the DesignSync GUI's **Tools=>Options** menu, SyncAdmin writes to the user's registry file.

To programmatically read, write or delete registry values, use the sregistry command set.

Less commonly used registry settings do not have a SyncAdmin interface. Those can be found in the Registry Settings topics in this book. The Registry Settings topics cover all available registry settings, noting whether the registry key has a SyncAdmin interface.

Client Registry Files

The registry files that contain configuration information for DesignSync clients are listed below in the order of precedence.

- `<SYNC_USER_CFGDIR>/UserRegistry.reg`

The `UserRegistry.reg` file contains a user's personal preference settings. These settings can be changed using the **Tools=>Options** menu in the DesignSync graphical user interface. `SYNC_USER_CFGDIR` defaults to `<HOME>/synchronicity` on Unix, and `%AppData%\Synchronicity` on Windows.

- `<SYNC_PROJECT_CFGDIR>/ProjectRegistry.reg`
- The `ProjectRegistry.reg` file lets a project leader set preferences for a project. To do so, the project leader creates a project directory and a `ProjectRegistry.reg` file within that directory. The project leader sets the `SYNC_PROJECT_CFGDIR` environment variable to the project directory and then customizes the `ProjectRegistry.reg` file using SyncAdmin. Project team members must set the `SYNC_PROJECT_CFGDIR` environment variable to the project directory so that their DesignSync clients can read the

project registry. For more information, see the Project Specific Configurations topic.

- `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`
1. The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. Installing the software generates a default `SiteRegistry.reg` file. This file contains the default settings that are inherited by every user. By using the SyncAdmin tool, a DesignSync administrator can override some of these default settings that will then be inherited by DesignSync clients and SyncServers. Users can override some of these defaults by choosing their own personal preferences in the DesignSync GUI.

Note: `SYNC_SITE_CNFG_DIR` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `SYNC_SITE_CNFG_DIR` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`
1. The `SynRegistry.reg` file contains information that is required by DesignSync to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Server Registry Files

The registry files that contain configuration information for SyncServers are listed below in the order of precedence.

- `<SYNC_CUSTOM_DIR>/servers/<host>/<port>/PortRegistry.reg`
(UNIX only)
- The `PortRegistry.reg` file contains preference settings that are defined for a particular host `<host>` and port `<port>` number for a specific SyncServer. Since multiple servers can be configured in the same `SYNC_DIR` installation, each server has its own `PortRegistry.reg` registry file. A server's `PortRegistry.reg` file is generated when the server is installed. `SYNC_CUSTOM_DIR` defaults to `<SYNC_DIR>/custom`.

If your server preferences are not port-specific, you can instead include your preferences in the site registry file, `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`, which is read by all servers in an installation.

Note: The `SYNC_SITE_CNFG_DIR` environment variable defaults to `<SYNC_CUSTOM_DIR>/site/config`.

- `<SYNC_SITE_CNFG_DIR>/SiteRegistry.reg`
1. The `SiteRegistry.reg` registry is accessible to both DesignSync clients and servers. Use the site registry to set up default preferences for all servers in an installation. Then use the port registry to set preferences for a particular SyncServer. Installing the software generates a default `SiteRegistry.reg` file.

Note: `SYNC_SITE_CNFG_DIR` is equivalent to `<SYNC_CUSTOM_DIR>/site/config`; if `SYNC_SITE_CNFG_DIR` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `SiteRegistry.reg` registry.

- `<SYNC_ENT_CUSTOM>/config/EntRegistry.reg`

The `EntRegistry.reg` registry is accessible to both DesignSync clients and servers. This file is used for enterprise-wide settings.

Note: `SYNC_ENT_CUSTOM` is equivalent to `<SYNC_CUSTOM_DIR>/enterprise`; if `SYNC_ENT_CUSTOM` is not set, but `SYNC_CUSTOM_DIR` is set, DesignSync will still access the `EntRegistry.reg` registry.

- `<SYNC_DIR>/share/SyncRegistry.reg`
1. The `SynRegistry.reg` file contains information that is required by DesignSync to find and load shared object libraries. This file should never need to be modified.

Note: In order for changes made to the server registry files to take effect, you must restart your SyncServer. An alternative to restarting the SyncServer is to use the registry reset command in a server-side script to re-read the registry files.

Related Topics

Using Environment Variables

Alphabetical List of Registry Keys

Alphabetical List of Registry Keys

For ease of reference, this topic contains a list of all the published DesignSync registry keys available to the DesignSync users and administrators. The list is contained in a table which contains:

- the simple key name (not the full registry key path), as a link to the topic where the key is documented,
- whether the key can be added to the client or server registry file
- the category where the documentation is located in the Registry Files book
- the name of the option in SyncAdmin, if applicable, as a link to the topic where the setting is documented.

Tip: Because this list is strictly alphabetical, it is particularly useful if you know all or part of the key name you want more information about. If you are looking for general customization information, start with the page devoted to that topic. For example, if you want to know how to customize backup functionality, start by viewing the topics available in the Data Storage and Maintenance Registry Settings book.

Key	Server/Client		Category	Sync Opt
	Server	Client		
ACAdminEnabled	Yes	No	SyncServers	Ena Con Adm
ACDenyLogRotateSize	Yes	Yes	Client/Server Communication	Acc den size
ACLogDeny	Yes	Yes	Client/Server Communication	Acc den
Active (General)	Yes	No	Mirror Administration Server Registry Settings	
Active (Mirror Directories)	Yes	No	Mirror Administration Server Registry Settings	
AddAccessControls	Yes	No	Modules Registry Settings	
AdminAllowUserCommand		Yes		

DesignSync Data Manager Administrator's Guide

AdminAllowUserFlags				
AdminAllowUserStates				
AdminAllowUserTags				
AdvancedAddMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedCancelFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedCiFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedCoFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedCompareFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedContentsFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedLockBranchFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedMcacheFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedMkBranchFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedMkHrefFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedMkModFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedMvMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedRetireFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedRmMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con

Registry Files

AdvancedSetFilterFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedStatusFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedTagFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedUnlockFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedVaultInfoFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
AdvancedVhistoryFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
Allow Recursion\Cadence NonView Folder				
AllowableDriveTypes	No	Yes	DesignSync Client Optimizations Registry Settings	
AllowAutoRootCreation	No	Yes	Modules Registry Settings	Allo crea wor root
AllowCiObjectStates				
AllowCoObjectStates				
AllowFilteredLinks	Yes	No	Modules Registry Settings	
AllowIntermediateLinks	No	Yes	Modules Registry Settings	
AllowNonSecureCachePopulate	Yes	No	Data Storage And Maintenance Registry Settings	
AllowRecursion\Cadence View Folder				
AllowVaultRelocation	No	Yes	DesignSync Client Commands Registry Settings	
AlwaysShowTags	No	Yes	Modules Registry Settings	
Ancestor (DiffTool Flags)	No	Yes	DesignSync diff Display Registry Settings	
AutoBlend				
AutoCreateLog File and DefaultLogFile	No	Yes	DesignSync Client Startup Registry Settings	Initi
AutoDeleteEmptyMirrors	Yes	No	Mirror Administration Server Registry Settings	
AutoMerge	Yes	No	Modules Registry Settings	
AutoRefreshTimer	No	Yes	DesignSync GUI Registry Settings	

DesignSync Data Manager Administrator's Guide

AutoRequeue	Yes	No	Mirror Administration Server Registry Settings
BackupEarliestTime	Yes	No	Data Storage and Maintenance Registry Settings
BackupFrequency	Yes	No	Data Storage and Maintenance Registry Settings
BackupFullBackupDay	Yes	No	Data Storage and Maintenance Registry Settings
BackupRetentionTime	Yes	No	Data Storage and Maintenance Registry Settings
BackupNumberOfIncrementals	Yes	No	Data Storage and Maintenance Registry Settings
BackupLogRetentionTime	Yes	No	Data Storage and Maintenance Registry Settings
Binary (DiffTool Flags)	No	Yes	DesignSync diff Display Registry Settings
Browser	No	Yes	DesignSync GUI Registry Settings
Branch	No	Yes	DesignSync GUI Registry Settings/Columns
Cache	No	Yes	DesignSync GUI Registry Settings/Columns
CacheDisableRefCount	No	Yes	DesignSync Client Optimizations Registry Settings
CacheUseHostPost	No	Yes	DesignSync Client Environment Registry Settings
Cadence NonView Folder	No	Yes	Vendor Objects Registry Settings
Cadence View Folder	No	Yes	Vendor Objects Registry Settings
Cadence\AutoCheckInComment	No	Yes	Vendor Objects Registry Settings
Cadence\MaxDepth	No	Yes	Vendor Objects Registry Settings
CadenceView	Yes	No	Vendor Objects Registry Settings
CDOA	No	Yes	Vendor Objects Registry Settings
CDOA\Exclude	No	Yes	Vendor Objects Registry Settings
CDOA\ExcludeProc	No	Yes	Vendor Objects Registry Settings
CheckChangeAffectsMirror	Yes	No	Repository and Mirror Administrator Server Registry Settings
CheckQuota	Yes	No	
Client\LookAndFeel		Yes	
Client\name	No	Yes	
Client\name\CommandLine	No	Yes	
Client\name\Enabled	No	Yes	
Client\name\Priority	No	Yes	
Client\Theme		Yes	
Client\TimeFormat		Yes	
Client\Tools		Yes	
ClientMetadataLockNumTries	No	Yes	Workspace Metadata Registry Settings

Registry Files

ClientMetadataLockTimeout	No	Yes	Workspace Metadata Registry Settings	
ClientUseMirrorCheck	No	Yes	DesignSync Client's Mirror Functionality Registry Settings	
ClientVaultGetOnly	No	Yes	DesignSync Client Commands Registry Settings	
CoFetchRetired	No	Yes	DesignSync Client Commands Registry Settings	
Collections\Units	No	Yes	Vendor Objects Registry Settings	
Columns	No	Yes	DesignSync GUI Registry Settings	
CommitImmediate	No	Yes	DesignSync Client Commands Registry Settings	Con cha imm imm noir
ConfirmPopForce	No	Yes	DesignSync GUI Registry Settings	
CoOptions (All Mirrors)	Yes	No	Mirror Administration Server Registry Settings	
CoOptions (Specific Mirror)	Yes	No	Mirror Administration Server Registry Settings	
Specified Mirrors"				
CountRevCtlAttachments	Yes	No	ProjectSync Registry Settings	
CustomColumns				
CustomToolbar		Yes		
DataSheetInFrame	No	Yes	DesignSync GUI Registry Settings	
DBCheckpointTime	Yes	No	Data Storage and Maintenance Registry Settings	
DefaultAutoRootPath	No	Yes	Modules Registry Settings	Allo crea wor root
DefaultFetchType	No	Yes	DesignSync Client Commands Registry Settings	Def Sta
DefaultLogDir	No	Yes	DesignSync Client Startup Registry Settings	Put dire
DefaultLogFile	No	Yes	DesignSync Client Startup Registry Settings	Initi
DelegateAC	No	Yes	Enterprise Design Settings	
Diff2Format	No	Yes	DesignSync diff Display Registry Settings	
Diff Tool\Command	No	Yes		
DiffTool\Flags\IgnoreWhite	No	Yes	DesignSync diff Display Registry Settings	
DiffView\BlackBg	No	Yes		
DirSymbolicLinkMode				
DSVS\AutoCheckInComment				

DesignSync Data Manager Administrator's Guide

EnableAliasesKeyword	Yes	No	SyncServers	
EnableClientMirrorUpdate	No	Yes	DesignSync Client's Mirror Functionality Registry Settings	
EnableCommentEditor	Yes	Yes	DesignSync Client Commands Registry Settings	Use com
EnableDirectFetch	No	Yes	DesignSync Client Optimizations Registry Settings	
EnableKeywordExpansion	No	Yes	DesignSync Client Optimizations Registry Settings	
EnableLinkIn	No	Yes	DesignSync Client Optimizations Registry Settings	
EnablePreCheckin	No	Yes		
EnableRealMirrorPaths	Yes	Yes	DesignSync Client Commands Registry Settings	
EnforceRestrictedCharacters	Yes	Yes	DesignSync Client Commands Registry Settings	Res Cha Res Cha
EnsureRoot	Yes	Yes	DesignSync Client Environment Registry Settings	
EnterpriseUserMapType	Yes	No	Enterprise Design Settings	
Expire	No	Yes	DesignSync GUI Registry Settings	
ExtendCadenceLMColumns	No	Yes	Vendor Objects Registry Settings	
FetchFromLocal	No	Yes	DesignSync Client Commands Registry Settings	Fet or c pos
FetchModuleNoHrefsAsModule	No	Yes	Modules Registry Settings	
FetchReadOnly	No	Yes	DesignSync Client Commands Registry Settings	Che only lock
FileEditor	No	Yes	DesignSync GUI Registry Settings	File
FileRetainTime	No	Yes	DesignSync Client Commands Registry Settings	reta mo time
FilterCrLf (Windows only)	No	Yes	DesignSync Client Commands Registry Settings	Ena Pro
FindAbsentFiles	Yes	No	Troubleshooting the Mirror System Registry Settings	
FindModuleByName	Yes	No		
FisrvDoesOp	No	Yes	Vendor Objects Registry Settings	
FontSize	No	Yes	DesignSync GUI Registry Settings	
ForbidAddMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Cor

Registry Files

ForbidCancelFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidCiFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidCmnds	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidCoFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidCompareFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidContentsFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidLockBranchFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidMcacheFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidMkBranchFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidMkHrefFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidMkModFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidMvMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidRetireFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidRmMemberFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidSetFilterFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con

DesignSync Data Manager Administrator's Guide

ForbidStatusFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidTagFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidUnlockFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidVaultInfoFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
ForbidVhistoryFlags	Yes	Yes	DesignSync Client Commands Registry Settings	GU Cus Con
FreeSpaceAbsErr	Yes	No	Data Storage and Maintenance Registry Settings	
FreeSpaceRelErr	Yes	No	Data Storage and Maintenance Registry Settings	
FreeSpaceAbsWarn (General)	Yes	No	Mirror Administration Server Registry Settings	
FreeSpaceAbsWarn (Mirror Directories)	Yes	No	Mirror Administration Server Registry Settings	
FreeSpaceRelWarn (General)	Yes	No	Mirror Administration Server Registry Settings	
FreeSpaceRelWarn (Mirror Directories)	Yes	No	Mirror Administration Server Registry Settings	
GenerateNote	Yes	No	ProjectSync Registry Settings	
HeartBeatInterval	Yes	No	Troubleshooting the Mirror System Registry Settings	
HelpMode	Yes	No	DesignSync Client Commands Registry Settings	Con Ref Mod
History\Expire		Yes		
History\RecentMax		Yes		
HrefModeChangeWithTopStaticSelector	Yes	Yes	Modules Registry Settings	
IfLock	No	Yes		Che pro Mod (-ifl
Ignore Case (DiffTool Flags)	No	Yes	DesignSync diff Display Registry Settings	
IgnoreInvalidPaths	No	Yes	Modules Registry Settings	
IgnoreKeys (DiffTool Flags)	No	Yes	DesignSync diff Display Registry Settings	
IgnoreWhite (DiffTool Flags)	No	Yes	DesignSync diff Display Registry Settings	

Registry Files

Initial folder (InitialDir, InitialDirSpecify)	No	Yes	DesignSync GUI Registry Settings	
InitScript	No	Yes	DesignSync Client Startup Registry Settings	Run scri
Keystrokes		Yes		
Keystrokes\Add Bookmark		Yes		
LogAtStartup	No	Yes	DesignSync Client Startup Registry Settings	Log auto beg
LogDetailedOutput	No	Yes	DesignSync Client Startup Registry Settings	Rec com com
LogFileMaxSize	Yes	No	Repository Server Registry Settings	
LookAndFeel	No	Yes	DesignSync GUI Registry Settings	
MADAddMcacheModeServer	Yes	No	Mirror Administration Server Registry Settings	
MADNotifyInterval	Yes	No	Mirrors	
MADUpdateFailureNotifyTime	Yes	No	Mirror Administration Server Registry Settings	
MADUpdateFailureRetries	Yes	No		
MaintenanceRetryAttempts	No	Yes	Client Server Communication	Ret in M Mod
MaintenanceRetryInterval	No	Yes	Client Server Communication	Nur Atte
MaintenanceTimeout	Yes	No	Data Storage and Maintenance Registry Settings	
ManagedLinkInMirrorMode	No	Yes	DesignSync Client's Mirror Functionality Registry Settings	
ManageViewManifest	No	Yes	Vendor Objects Registry Settings	
MapMemberOps	No	Yes		
MaxVersionsDisplayed	Yes	No	Modules Registry Settings	
MaxWorkerThreads	No	Yes	DesignSync Client Optimizations Registry Settings	
Mcache\IgnoreInvalidPaths				
Mcache/Mode	No	Yes	Modules Registry Settings	
Mcache/Paths	No	Yes	Modules Registry Settings	
McacheAllowIntermediateLinks				
MilkywayLinkInExemption	No	Yes	Deprecated.	
MinCiCommentLength	No	Yes		
Mirrors\LogFileMaxSize	Yes	No	Mirror Administration Server Registry Settings	
ModuleFailureRetryAttempts	No	Yes	DesignSync Client Commands Registry Settings	Ret Che

DesignSync Data Manager Administrator's Guide

ModuleFailureRetryInterval	No	Yes	DesignSync Client Commands Registry Settings	Ret Che
ModuleToolbar		Yes		
MPDPushFailureNotifyTime	Yes	No	Repository Server Registry Settings	
MPDPushFailureRetries	Yes	No	Repository Server Registry Settings	
MPDNotifyInterval	Yes	No	Repository Server Registry Settings	
MUPNewVersForPopulate	Yes	No	PopulateOptions	
MUPsMaxPerMAD	Yes	No	Mirror Administration Server Registry Settings	
NFSTimeout	Yes	No	Data Storage and Maintenance Registry Settings	
NoAssignmentUserList	Yes	No	Enterprise Design Settings	
NoChecksumMod	No	Yes	DesignSync Client Commands Registry Settings	
NotifyExcludeByFilter	No	Yes	DesignSync Client Commands Registry Settings	Ret in M Mod
NumberHoursToPopulateAllMirrors	Yes	No	Mirror Administration Server Registry Settings	
ObsoleteTimer	No	Yes	DesignSync GUI Registry Settings	
OrphanTimeout	No	Yes	Workspace Metadata Registry Settings	
PBFCAAllowUserToOwnFile	Yes	No	DesignSync Client Environment Registry Settings	
PBFCEnabled	Yes	No	DesignSync Client Environment Registry Settings	
PerformUpgrade	Yes	No	Mirror Administration Server Registry Settings	
PGMaintenanceTime	Yes	No	Data Storage and Maintenance Registry Settings	
PopulateDoVersionExtendedInfo	No	Yes	DesignSync Client Modules Registry Settings	
PopulateIgnoreLockedObjects	No	Yes		
PopulateNoEmptyDirs	No	Yes	DesignSync Client Commands Registry Settings	Pop dire
PopulateOptions (All Mirrors)	Yes	No	Mirror Administration Server Registry Settings	
PopulateOptions (Specific Mirror)	Yes	No	Mirror Administration Server Registry Settings	
PopulateUseIncremental	No	Yes	DesignSync Client Commands Registry Settings	
Precheckin	No	Yes	DesignSync Client Optimizations Registry Settings	
ProcessSizeLimit	Yes	No	Data Storage and Maintenance Registry Settings	

Registry Files

ProjectCache	No	Yes	DesignSync Client Environment Registry Settings	Pro
ProjectCacheTclScript	No	Yes	DesignSync Client Environment Registry Settings	Pro
ProjectDescription	No	Yes	DesignSync Client Environment Registry Settings	Pro
ProjectVault	No	Yes	DesignSync Client Environment Registry Settings	Pro
ProxyNamePort	Yes	No		
RecentMax	No	Yes	DesignSync GUI Registry Settings	
RefreshOnTreeSelect	No	Yes	DesignSync GUI Registry Settings	
Replicate Scrub Age	Yes	No	Mirror Administration Server Registry Settings	
Replicate Scrub Time	Yes	No	Mirror Administration Server Registry Settings	
RequireCIComment and MinCiCommentLength	No	Yes	DesignSync Client Commands Registry Settings	Min com
RestrictedCharacters	Yes	Yes	DesignSync Client Commands Registry Settings	Res Cha Res Cha
RmVaultKeepVid	No	Yes	DesignSync Client Commands Registry Settings	Keep info dele
RunInitScript and InitScript	No	Yes	DesignSync Client Startup Registry Settings	Run scri
SaveLocal	No	Yes	DesignSync Client Commands Registry Settings	Loc sav
SaveLogFiles	Yes	No	Troubleshooting the Mirror System Registry Settings	
SavePreMergedFile	No	Yes	DesignSync Client Environment Registry Settings	
Script	Yes	No	Troubleshooting the Mirror System Registry Settings	
ScrubGeneratedMirrors	Yes	No	Troubleshooting the Mirror System Registry Settings	
separateTechFile	No	Yes	Deprecated.	
ShowHrefsNeedCheckinStatus	No	Yes	Modules Registry Settings	
ShowSharedDevelopmentAreaOption	No	Yes	Development Areas Registry Settings	
Simulink\Add\File	No	Yes	Vendor Objects Registry Settings	
Simulink\Checkin\ModuleInstance	No	Yes	Vendor Objects Registry Settings	
Simulink\Checkin\Selected	No	Yes	Vendor Objects Registry Settings	
Simulink\Lock\Selected	No	Yes	Vendor Objects Registry Settings	

DesignSync Data Manager Administrator's Guide

Simulink\Mkmod\Workspace	No	Yes	Vendor Objects Registry Settings	
Simulink\OutputCmds	No	Yes	Vendor Objects Registry Settings	
Simulink\Populate\InitialWs	No	Yes	Vendor Objects Registry Settings	
Simulink\Populate\ModuleInstance	No	Yes	Vendor Objects Registry Settings	
Simulink\Populate\Revert	No	Yes	Vendor Objects Registry Settings	
Simulink\Cancel\Selected	No	Yes	Vendor Objects Registry Settings	
Simulink\Populate\Selected	No	Yes	Vendor Objects Registry Settings	
Simulink\Populate\Version	No	Yes	Vendor Objects Registry Settings	
Simulink\Remove\Selected	No	Yes	Vendor Objects Registry Settings	
Simulink\Remove>ShowRmComment	No	Yes	Vendor Objects Registry Settings	
Simulink\Show>Status	No	Yes	Vendor Objects Registry Settings	
Simulink\Tag\InitialBranch	No	Yes	Vendor Objects Registry Settings	
Simulink\Tag\ModuleVersion	No	Yes	Vendor Objects Registry Settings	
SiteFilter and Filter	No	Yes	DesignSync Client Commands Registry Settings	Exc site (Sit The exc site (Us
SMAAllowFetchFromRepository	Yes	No	Mirror Administration Server Registry Settings	
Startup	No	Yes		
SymbolicLinkMode and DirSymbolicLinkMode	Yes	Yes	DesignSync Client Environment Registry Settings	
SyncGUI	No	Yes	DesignSync GUI Registry Settings	
Synopsys	Yes	No	Deprecated.	
SynopsysMilkywayExec	Yes	No	Deprecated.	
TagList	No	Yes	DesignSync GUI Registry Settings	
tcl_filename\Enabled				
tcl_filename\FileName				
tcl_filename\Priority				
tcl_filename\Type				
tcl-command\Enabled				
tcl-command\Priority				
tcl-command\TclScript				
tcl-command\Type				
Theme		Yes		
TimeFormat	No	Yes	DesignSync GUI Registry Settings	
TokenLifeTime	Yes	No	ProjectSync Registry Settings	

Tools\...\Command		Yes		
Tools\...\Description		Yes		
Tools\...\Icon		Yes		
Tools\...\Output		Yes		
Tools\...\SelObjAction		Yes		
Tools\...\Type		Yes		
TransactionLogRetainRecords	Yes	No	Data Storage and Maintenance Registry Settings	
TransferCompressed	Yes	Yes	Client Server Communication	Ena com betw and
TransferDelta	Yes	Yes	Client Server Communication	Ena tran clie
TransferTimeout	No	Yes	Client Server Communication	Lim Tim
TreeViewLargelcons	No	Yes	DesignSync GUI Registry Settings	
trigger_name\Exclude				
trigger_name\Exclude\property_value				
trigger_name\Require				
trigger_name\Require\property_value				
TrustedAgents	Yes	Yes	Client/Server Communication	
TurnOffAutomaticRetry	Yes	No	Troubleshooting the Mirror System Registry Settings	
UniqueKeys	Yes	No	Data Storage and Maintenance Registry Settings	
UpdateScheme	Yes	No	Enterprise Design Settings	
UploadTmpDir	Yes	No	Data Storage and Maintenance Registry Settings	Ten Ser
UseLegacyModules	Yes	Yes	Modules Registry Settings	
UseLockForExport	No	Yes	Deprecated.	
UseSyncExclude	No	Yes	DesignSync Client Commands Registry Settings	
VaultBrowser/.../Color	No	Yes	VaultBrowser	
ValidateReferenceWorkspace	No	Yes	DesignSync Client Optimizations Registry Settings	
WebObjectCache	No	Yes	DesignSync Client Environment Registry Settings	

Client/Server Communication

Audit Trail Log Maximum File Size (ACDenyLogRotateSize)

When the Access Control system denies access to a command and the Audit Trail Log is enabled, you can specify the maximum size of the log. After the maximum file size is reached, the file is automatically cleared when the next access control request is denied. By default the maximum log size is 1 MB. This option can also be set with SyncAdmin | Site Options | Access control deny log rotation size.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\AccessControl\
ACDenyLogRotateSize=dword:1
```

Allowed Values

Key value	Description
ACDenyLogRotateSize=dword:1	Sets the maximum log file to 1 MB. (Default)
ACDenyLogRotateSize=dword:<MB>	Specifies the maximum log file in MB.

Related Topics

[HTTP Proxies](#)

[Limit Transfer Timeout](#)

[Maintenance Retry Attempts](#)

[Maintenance Retry Interval](#)

[Data Compression](#)

[Audit Trail Log \(ACLogDeny\)](#)

Audit Trail Log (ACLogDeny)

When the Access Control system denies access to a command, you can log information about the action that was denied. If logging is enabled, each time a command is denied, a line appended to the log with the following information:

- The denied action (Populate, Checkout, Checkin, Tag, etc.)
- The user performing the action
- Parameters specified with the action (URL, Selector, etc.)

Note: You can optionally capture the reason that the access was denied.

The log file is named `deny.log` and is located in the logs directory within the custom server directory

`($SYNC_CUSTOM_DIR/servers/<serverName>/<port>/logs/deny.log)`

The maximum file size of the log is configurable through Audit Trail Log Maximum File Size (`ACDenyLogRotateSize`).

This option can also be set with SyncAdmin | Site Options | Access control deny logging.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings are all client side, so must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\AccessControl\ACLogDeny=dword:0`

Allowed Values

Key value	Description
<code>ACLogDeny=dword:0</code>	Disables access control logging. (Default)
<code>ACLogDeny=dword:1</code>	Log all denials from the access control system.
<code>ACLogDeny=dword:2</code>	Log all denials from the access control system and the rule that resulted in the denial. (2 lines per denied attempt.)

Related Topics

HTTP Proxies

Limit Transfer Timeout

Maintenance Retry Attempts

Maintenance Retry Interval

Data Compression

Audit Trail Log Maximum File Size

Maintenance Retry Attempts (MaintenanceRetryAttempts)

This setting controls the processing of DesignSync requests when the DesignSync server is in read-only mode. The registry key can also be set by the "Retries to Server in Maintenance Mode" on the SyncAdmin's General Performance pane.

By default, if a server is in maintenance mode, therefore read-only, the client's communication to a server is lost. The "MaintenanceRetryAttempts" is enabled by setting the value to the number of retries the client attempts before failing the write operation.

Note: The number of attempts value should be specified in hexadecimal.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\MaintenanceRetryAttempts=dword:0
```

Allowed Values

Key value	Description
-----------	-------------

MaintenanceRetryAttempts=dword:0	Disables the maintenance retry attempts feature. (Default)
MaintenanceRetryAttempts=dword:1	Enables the maintenance retry attempts rate and sets it to 1 retry. (Minimum Value)
MaintenanceRetryAttempts=dword:<value>	Enables the maintenance retry attempts rate and sets it to the hexadecimal value specified for number of attempts.
MaintenanceRetryAttempts=dword:1869f	Enables the maintenance retry attempts rate and sets it to 99,999 retries. (Maximum Value)

Related Topics

HTTP Proxies (ProxyNamePort)

Limit Transfer Timeout (TransferTimeout)

Maintenance Retry Interval (MaintenanceRetryInterval)

Data Compression (TransferCompressed)

Delta Transfer Compression (TransferDelta)

Maintenance Retry Interval (MaintenanceRetryInterval)

When the Retries to Server in Maintenance Mode setting is turned on, this key can be used to set the wait time between retry attempts. The registry key can also be set by the "Number of Retry Attempts" value in SyncAdmin's General Performance pane.

By default, this key is set to 60 seconds (dword:3C). If this value is set, and the "Maintenance Retry Attempts" is not enabled, this key value is ignored.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

DesignSync Data Manager Administrator's Guide

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\MaintenanceRetryInterval=dword:0

Allowed Values

Key value	Description
MaintenanceRetryInterval=dword:0	Disables maintenance retry.
MaintenanceRetryInterval=dword:1	Sets the maintenance retry interval to 1 second. (Minimum Value)
MaintenanceRetryInterval=dword:3C	Sets the maintenance retry interval to 60 seconds. (Default)
MaintenanceRetryInterval=dword:<value>	Sets the maintenance retry interval to the specified hexadecimal value.

Related Topics

HTTP Proxies (ProxyNamePort)

Limit Transfer Timeout (TransferTimeout)

Maintenance Retry Attempts (MaintenanceRetryAttempts)

Data Compression (TransferCompressed)

Delta Transfer Compression (TransferDelta)

HTTP Proxies (ProxyNamePort)

An HTTP proxy can be set up to access remote SyncServers from within a DesignSync session.

Specify the proxy IP address and port by setting `ProxyNamePort` to a string of the form "`<ProxyName>:<Port#>`".

For example, `ProxyNamePort="linus:80"`, where `linus` is the machine running the proxy server on port 80.

Alternatively, you can define the environment variable `ProxyNamePort` instead of editing the registry file; for example:

```
setenv ProxyNamePort linus:80
```

For more information about using the HTTP Proxy, see [HTTP Proxy](#).

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Directories\ProxyNamePort="<ProxyName>:<Port#>"
```

Related Topics

Limit Transfer Timeout (TransferTimeout)

Maintenance Retry Attempts (MaintenanceRetryAttempts)

Maintenance Retry Interval (MaintenanceRetryInterval)

Data Compression (TransferCompressed)

Delta Transfer Compression (TransferDelta)

Data Compression (TransferCompressed)

DesignSync has the ability to compress data before it is transferred between the client and SyncServer. The registry key can also be set by "Enable data compression between client and server" option in SyncAdmin's General Communications.

For more information on when to use data compression, see SyncAdmin's Communications tab.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the

`syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\TransferCompressed=dword:0

Allowed Values

Key value	Description
TransferCompressed=dword:0	Disables data transfer compression. (Default)
TransferCompressed=dword:1	Enables data transfer compression

Related Topics

HTTP Proxies (ProxyNamePort)

Limit Transfer Timeout (TransferTimeout)

Maintenance Retry Attempts (MaintenanceRetryAttempts)

Maintenance Retry Interval (MaintenanceRetryInterval)

Delta Transfer Compression (TransferDelta)

Delta Transfer Compression (TransferDelta)

File delta transfer compression is a compression mechanism that relies on determining and transferring the file differences between the file version in the workspace and the one on the server. Enabling delta transfer compression can improve the performance of checkin, checkout, and populate operations. The registry key can also be set by "Enable delta file transfer between client and server" in SyncAdmin's General Communications.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the `sregistry` command `set`. The settings are all client side, so must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\TransferDelta=dword:0

Allowed Values

Key value	Description
TransferDelta=dword:0	Disables delta transfer compression. (Default)
TransferDelta=dword:1	Enables delta transfer compression.

Related Topics

HTTP Proxies

Limit Transfer Timeout

Maintenance Retry Attempts

Maintenance Retry Interval

Data Compression

Limit Transfer Timeout (TransferTimeout)

This setting controls the length of time DesignSync clients wait for their server connection to be restored before timing out. The registry key can also be set by the "Limit Transfer Timeout" field on the SyncAdmin's General Performance pane.

By default, the client operation never times out if a client's communication to a server is lost. The "Limit Transfer Timeout" is enabled by specifying the timeout interval as the value of the key.

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

DesignSync Data Manager Administrator's Guide

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\TransferTimeout=dword:0

Allowed Values

Key value	Description
TransferTimeout=dword:0	Disables the limit transfer timeout feature. (Default)
TransferTimeout=dword:1	Enables the limit transfer timeout rate and sets it to 1 second. (Minimum Value)
TransferTimeout=dword:<time>	Enables the limit transfer timeout rate and sets it to value specified for time. The time value should be specified in hexadecimal.
TransferTimeout=dword:1f4	Enables the limit transfer timeout rate and sets it to 500 seconds. (Maximum Value)

Related Topics

HTTP Proxies (ProxyNamePort)

Maintenance Retry Attempts (MaintenanceRetryAttempts)

Maintenance Retry Interval (MaintenanceRetryInterval)

Data Compression (TransferCompressed)

Delta Transfer Compression (TransferDelta)

Trusted Agents and Processes (TrustedAgents)

Client/Server communication through a web browser requires a handshake between the two. In many cases, this is done with an embedded security code to insure that the request is a desired, legitimate, and safe operation for the server. Some clients, however can be authorized as "Trusted" meaning that all requests coming from the client are safe. The DesignSync GUI is an example of a trusted client. When the server receives a request from the DesignSync GUI, for example, for an object datasheet, it does not require a trusted authentication cookie. The administrator can designate custom applications as "Trusted."

This registry setting pertains to communication between DesignSync clients and a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings are all client side, so must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to

the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\Security\Truste
dAgents\TrustedAgents@=none
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\Security\Truste
dAgents\Agent1="<client>"
```

...

Example

This is an example of setting the Trusted Agent key for a trusted client tool. This example uses the open source tool, "curl."

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\Security\Truste
dAgents\TrustedAgents\@=none
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\Security\Truste
dAgents\AgentID="curl/*"
```

Allowed Values

Key Value	Description
<code>TrustAgents\@=none</code>	A placeholder for the registry key.
<code>AgentID="<i><client></i>"</code>	<p>AgentId can be any string and/or incremented string.</p> <p>Tip: For ease of use, using a standard prefix with an increment number can help you uniquely identify the agent. For example: If the desired prefix was Agent, then the first client defined would be Agent1, the second Agent2, etc.</p> <p>IMPORTANT: DesignSync uses the prefix TrustedAgent for all included trusted agents.</p> <p>The client value url path should uniquely identify the sending client. It does not need to include</p>

	URL protocol.
--	---------------

Registry Settings for DesignSync Clients

DesignSync Client Startup Registry Settings

Name of log file (AutoCreateLogFile and DefaultLogFile)

Specifies whether to create a new and unique log file for each session (Default) or reuse the same log file name for each session. This option can also be set with the "Initial log file" option in DesignSync Administrator's Logging pane.

By default, new and unique log files are used.

If a single log file is used, you also have the option to set the name of the log file used.

Otherwise the log file name is auto-generated with the following format:

dss_<date>_<time>.log.

Notes

- If the log file name is set, but you are creating unique logs for each session, the log file name is ignored.
 - While you can technically set a path as a log file, DesignSync recommends that you use a simple file name for the DefaultLogFile key and use the DefaultLogDir to set the path.

For more information on setting log file settings, see the DesignSync Administrator's Logging pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\General\Logging\AutoCreateLogFile=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\General\Logging\DefaultLogFile="<dss_session.log>"

Allowed Values

Key value	Description
AutoCreateLogFile=dword:0	Reuses the same log file name for all DesignSync client sessions.
AutoCreateLogFile=dword:1	Uses a unique generated file name for each DesignSync client session log. (Default)
DefaultLogFile="<dss_session.log>"	Sets the name of the log file when AutoCreateLogFile is disabled (dword:0). The default log file is "dss_session.log."

Related Topics

Automatically Log (LogAtStartup)

Information to log (LogDetailedOutput)

Location of Logs (DefaultLog Dir)

DesignSync client log size (MaxClientLogSize)

Startup script (RunInitScript and InitScript)

Initial folder (InitialDir, InitialDirSpecify)

Logging Options

Startup Options

Location of Logs (DefaultLogDir)

Specifies the directory location to store users' DesignSync log files. This option can also be set with "Put log files in this directory" in DesignSync Administrator's Logging pane.

By default, log files are stored in the user's directory defined in DefaultLogDir.

DesignSync Data Manager Administrator's Guide

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\General\CmdTable\DefaultLogDir\@="\$HOME"

Allowed Values

Key value	Description
DefaultLogDir\@="\$HOME"	Sets the log directory to the default home directory for the user. For UNIX users, that is as defined as \$HOME. For Windows users there is no HOME environment variable, the user profile directory is used instead.
DefaultLogDir\@="<c>:/<directory>"	Sets the log directory to a specific directory. The user writing to the log must have write access to the directory, or writes to the log file will fail silently. (Windows)
DefaultLogDir\@="/<unix>/<path>"	Sets the log directory to a specific directory. The user writing to the log must have write access to the directory, or writes to the log file will fail silently. (UNIX)

Related Topics

Automatically Log (LogAtStartup)

Information to log (LogDetailedOutput)

Name of log file (AutoCreateLog File and DefaultLogFile)

DesignSync client log size (MaxClientLogSize)

Startup script (RunInitScript and InitScript)

Initial folder (InitialDir, InitialDirSpecify)

Logging Options

Startup Options

Information to log (LogDetailedOutput)

Determines whether to write both the commands executed and the command results, as displayed in DesignSync's output window, to the log file, or only the command executed. This option can also be set with "Record both commands and command results" in DesignSync Administrator's Logging pane. By default, both executed commands and results are saved in the log file.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

`HKEY_CURRENT_USER\Software\Synchronicity\General\Logging\LogDetailedOutput=dword:1`

Allowed Values

Key value	Description
<code>LogDetailedOutput=dword:0</code>	Disables command results logging.
<code>LogDetailedOutput=dword:1</code>	Enables command results logging. (Default)

Related Topics

Automatically Log (LogAtStartup)

Location of Logs (DefaultLog Dir)

Name of log file (AutoCreateLog File and DefaultLogFile)

DesignSync client log size (MaxClientLogSize)

Startup script (RunInitScript and InitScript)

Initial folder (InitialDir, InitialDirSpecify)

Logging Options

Startup Options)

Automatically Log (LogAtStartup)

Determines whether DesignSync automatically begins logging users' sessions when they invoke a DesignSync client. The registry key can also be set by "Logging should automatically begin at startup" in DesignSync Administrator's Logging pane. This option is enabled by default.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Logging\LogAtStartup=dword:1
```

Allowed Values

Key value	Description
LogAtStartup=dword:0	Disables DesignSync client logging.
LogAtStartup=dword:1	Enables DesignSync client logging. (Default)

Related Topics

Information to log (LogDetailedOutput)

Location of Logs (DefaultLog Dir)

Name of log file (AutoCreateLog File and DefaultLogFile)

DesignSync client log size (MaxClientLogSize)

Startup script (RunInitScript and InitScript)

Initial folder (InitialDir, InitialDirSpecify)

Logging Options

Startup Options

DesignSync client log size (MaxClientLogSize)

The setting controls the maximum size of the DesignSync client log. The DesignSync client log is named `dss_<Date>_<Time>.log` and is located in the default user directory. If the maximum size is reached, the active client log is saved to: `dss_<Date>_<Time>.bak.log`, the existing log file is zeroed out, and client session logging continues.

Note: Every time the session log limit is reached the backup file is replaced, so it essentially conforms to the same maximum size as the log file. This means if your users tend to have long sessions, where they do not close the client, experience a volume of logged client transactions, or are running with synctrace, you may not see the complete log.

The dword value is specified in hexadecimal.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Logging\MaxClientLogSize=dword:0
```

Allowed Values

Key	Description
MaxClientLogSize=dword:0	Removes the maximum size limitation of the log file.

	The log file will continue to grow as long as the session that spawned the log is in process. (Default)
MaxClientLogSize=dword:#	Sets the maximum client size to the specified number of MB in hexadecimal.
MaxClientLogSize=dword:1	Sets the maximum client log size to 1 MB.
MaxClientLogSize=dword:A	Sets the maximum client log size to 10MB. For typical client sessions, without debug mode engaged, this is a reasonable size that prevents the log from growing too large.

See Also

About DesignSync Client Log Files

Name of log file (AutoCreateLog File and DefaultLogFile)

Location of Logs (DefaultLog Dir)

Information to log (LogDetailedOutput)

Automatically Log (LogAtStartup)

Startup script (RunInitScript and InitScript)

Specifies whether DesignSync should run a script at client startup and specifies the name of the script.

This option can also be set with "Run this startup script" in the DesignSync Administrator's Startup pane is used.

By default, DesignSync does not run a script on startup.

Note: If start initial script is **RunInitScript** is enabled, but the specified script does not exist, the error is silently ignored and the client still starts.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\General\Startup\RunInitScript=dword:0

HKEY_CURRENT_USER\Software\Synchronicity\General\Startup\InitScript="dsinit.dss"

Allowed Values

Key value	Description
RunInitScript=dword:0	Disables running an initial script on DesignSync client startup. (Default)
RunInitScript=dword:1	Enables running an initial script on DesignSync client startup.
InitScript="dsinit.dss"	Name of the script to run at DesignSync startup. The default value is "dsinit.dss"
InitScript=<value>	Changes the name of the script to run at DesignSync startup to the specified value.

Related Topics

Automatically Log (LogAtStartup)

Information to log (LogDetailedOutput)

Location of Logs (DefaultLog Dir)

Name of log file (AutoCreateLog File and DefaultLogFile)

Initial folder (InitialDir, InitialDirSpecify)

Logging Options

Startup Options

DesignSync Client Commands Registry Settings**Check-in detection of copied workspace (AllowVaultRelocation)**

The `AllowVaultRelocation` key controls how data is checked in if the vault URL associated with the files in a workspace do not match the value for the parent folder. By default, `ci` fails if the vault URL associated with a workspace's files does not match the

vault for the parent folder. This mismatch can occur when managed data is copied between workspaces. A subsequent check-in would result in the data being checked into the vault location specified in the workspace from which the data was copied, likely differing from the parent vault folder associated with the current directory.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\AllowVaultRelocation=dword:0
```

Allowed Values

Key	Description
<code>AllowVaultRelocation=dword:0</code>	Disallows vault relocation of the data being checked in. (Default)
<code>AllowVaultRelocation=dword:1</code>	Allows vault relocation of the data being checked in, thereby skipping the vault URL check.

Related Topics

Fetch Read Only (`FetchReadOnly`)

Default Fetch State (`DefaultFetchType`)

Client Vault Fetch State (`ClientVaultGetOnly`)

Check-in Minimum Comment Length (`RequireCiComment` and `MinCiCommentLength`)

Fetching retired data (`CoFetchRequired`)

Exclude Lists (`SiteFilter` and `Filter`)

Exclude Warnings (`NotifyExcludeByFilter`)

Retain Defaults (`FileRetainTime`)

Empty Directory Defaults (PopulateNoEmptyDirs)
Save Local Versions (SaveLocal)
Populate Full/Incremental (PopulateUseIncremental)
From Location Default (FetchFromLocal)
Keep Last-Version Information (RmVaultKeepVid)
Commit Module Changes Immediately (CommitImmediate)
Process locked objects only (IfLock)
Resolve paths for mirrors (EnableRealMirrorPaths)
CR/LF processing (FilterCrLf)
Use Checksum (NoChecksumMod)
Set Help Mode (HelpMode)
Checkin Error Retry Attempts (ModuleFailureRetryAttempts)
Checkin Error Retry Interval (ModuleFailureRetryInterval)
General Options
Default Fetch State Options
Exclude Lists
Command Defaults
Keyword Expansion

Client Vault Fetch State (ClientVaultGetOnly)

The `ClientVaultGetOnly` controls the default behavior of client vaults, which typically are for private, not shared data, and so may not want to use the same default fetch mode as typical server-based vaults.

The `ClientVaultGetOnly` registry key overrides the `DefaultFetchType` registry key. The `ClientVaultGetOnly` registry key also overrides any fetch mode specified to a command, whether that is via the command line or the DesignSync GUI.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\ClientVaultGetOnly=dword:0
```

Allowed Values

Key	Description
<code>ClientVaultGetOnly=dword:0</code>	Fetch read/write copies into the workspaces associated with a client vault.
<code>ClientVaultGetOnly=dword:1</code>	Fetch read-only copies into the workspaces associated with a client vault.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)
 From Location Default (FetchFromLocal)
 Keep Last-Version Information (RmVaultKeepVid)
 Commit Module Changes Immediately (CommitImmediate)
 Process locked objects only (IfLock)
 Resolve paths for mirrors (EnableRealMirrorPaths)
 CR/LF processing (FilterCrLf)
 Use Checksum (NoChecksumMod)
 Set Help Mode (HelpMode)
 Checkin Error Retry Attempts (ModuleFailureRetryAttempts)
 Checkin Error Retry Interval (ModuleFailureRetryInterval)
 General Options
 Default Fetch State Options
 Exclude Lists
 Command Defaults
 Keyword Expansion

Fetching retired data (CoFetchRetired)

The `CoFetchRetired` key controls whether checkout will fetch objects of a branch if the branch has been retired. By default, `co` of data on `<branch>:Latest` does not fetch objects if the branch is retired.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the

DesignSync Data Manager Administrator's Guide

`syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\CoFetchRetired=dword:0
```

Available Values

Key	Description
<code>CoFetchRetired=dword:0</code>	Does not allow populating the contents of a retired vault with <code><branch>:Latest</code> . (Default)
<code>CoFetchRetired=dword:1</code>	Allows populating the contents of a retired vault with <code><branch>:Latest</code> .

Related Topics

Fetch Read Only (`FetchReadOnly`)

Default Fetch State (`DefaultFetchType`)

Client Vault Fetch State (`ClientVaultGetOnly`)

Check-in Minimum Comment Length (`RequireCiComment` and `MinCiCommentLength`)

Check-in detection of copied workspace (`AllowVaultRelocation`)

Exclude Lists (`SiteFilter` and `Filter`)

Exclude Warnings (`NotifyExcludeByFilter`)

Retain Defaults (`FileRetainTime`)

Empty Directory Defaults (`PopulateNoEmptyDirs`)

Save Local Versions (`SaveLocal`)

Populate Full/Incremental (`PopulateUseIncremental`)

From Location Default (`FetchFromLocal`)

Keep Last-Version Information (`RmVaultKeepVid`)

Commit Module Changes Immediately (`CommitImmediate`)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Commit Module Changes Immediately (CommitImmediate)

This setting controls whether module changes are committed to memory immediately or not. The `CommitImmediate` registry key is set when SyncAdmin's Command Defaults pane is used to set the "Commit module changes immediately (-immediate/-noimmediate)" option.

The default is for the commands (remove and mvmember) to behave as though the **noimmediate** option was specified.

Note: This option preference is also used by DSDFII GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\CommitImmediate=dword:0

Allowed Values

Key	Description
CommitImmediate=dword:0	Specifies that the command behave as though the -noimmediate option was specified. (Default)
CommitImmediate=dword:1	Specifies that the command behave as though the -immediate option was specified.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Default Fetch State (DefaultFetchType)

The `DefaultFetchType` registry key is used to set the default fetch state for objects in the workspace when they are checked in or checked out. This value can also be set by SyncAdmin's Default Fetch State pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Options\Default
FetchType="<value>"
```

Allowed Values

Key	Description
<code>DefaultFetchType="get"</code>	Leaves unlocked copies of files in your local

	work area. This is equivalent to the "Unlocked Copies" option on the SyncAdmin Default Fetch State panel. (Default)
<code>DefaultFetchType="reference"</code>	Leaves a reference (or pointer) to the version of the object in the vault. These files are only visible to DesignSync. This is equivalent to the "References to versions" option on the SyncAdmin Default Fetch State panel. Note: Do not select this option when working with Cadence or Synopsys data.
<code>DefaultFetchType="share"</code>	This option creates a link from your work area to objects in the file cache. This is equivalent to the "Links to cache" option on the SyncAdmin Default Fetch State panel.
<code>DefaultFetchType="mirror"</code>	This option creates a link from your work area to objects in the mirror. This is equivalent to the "Links to Mirror" option on the SyncAdmin Default Fetch State panel.

Related Topics

Fetch Read Only (FetchReadOnly)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Use File Editor for Comment Entry (EnableCommentEditor)

This setting controls whether DesignSync uses the defined file editor for entering comments for command that provide an interactive comment interface if a comment isn't entered (for example: checkin, checkout, rollback, etc.). The file editor is defined in **SyncAdmin | General | File editor**. This setting can also be set with the **Use file editor for comment entry** checkbox

Note: If you plan to enter multibyte characters, use a editor that supports UTF-8 characters.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and

restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\Enable  
CommentEditor=dword:1
```

Allowed Values

Key	Description
<code>EnableCommentEditor=dword:0</code>	Do not use the defined editor to edit comments. This uses the old style interactive comments where you can enter a comment followed by a period (".").
<code>EnableCommentEditor=dword:1</code>	Uses the editor defined in the File Editor field on the SyncAdmin General panel to enter comments for DesignSync commands that prompt for comments. (Default)

Related Topics

SyncAdmin General Pane

Resolve paths for mirrors (EnableRealMirrorPaths)

This setting controls whether DesignSync uses the absolute or relative path to the mirror directory. By default, the path to a mirror is stored exactly as specified by the `setmirror` command. If your mirror directory is on an auto-mounted directory, you may instead want `setmirror` to resolve the path.

For example, with `EnableRealMirrorPaths=dword:1`, meaning that the path is resolved at access time, if you perform the command:

```
stcl> setmirror /s/data/testMirror .
```

`setmirror` resolves the auto-mounted path:

```
stcl> url mirror .  
file:///space/data/syncmgr/data/testMirror
```

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command `set`. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an

individual user may be made to that user's

\$SYNC_USER_CFGDIR/UserRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\EnableRealMirrorPaths=dword:0

Allowed Values

Key	Description
EnableRealMirrorPaths=dword:0	Uses the full resolved path of the mirror at the time the mirror was created to locate the mirror. (Default)
EnableRealMirrorPaths=dword:1	Uses a relative path which is resolved each time the mirror is accessed.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

DesignSync Data Manager Administrator's Guide

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

From Location Default (FetchFromLocal)

This setting controls whether `co` and `populate` commands fetch from the vault or from local. The `FetchFromLocal` registry key is set when SyncAdmin's Command Defaults pane is used, for whether to "Fetch from mirror or cache when possible".

The default is for `co` and `populate` to behave as though **-from local** was specified.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$(SYNC_CUSTOM_DIR)/site/config/SiteRegistry.reg` file or a `$(SYNC_PROJECT_CFGDIR)/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$(SYNC_USER_CFGDIR)/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\Client\General\Optimizations\FetchFromLocal=dword:1

Allowed Values

Key	Description
FetchFromLocal=dword:0	Specifies that co and populate to behave as though - from vault was specified.
FetchFromLocal=dword:1	Specifies that co and populate to behave as though - from local was specified. (Default)

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

DesignSync Data Manager Administrator's Guide

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Fetch Read Only (FetchReadOnly)

The `FetchReadOnly` setting controls whether checking out files (checkout without lock) places read/write or read-only copies in your work area. This option also affects check-in operations when you select the **Keep an unlocked copy** option in DesignSync. The `FetchReadOnly` registry key is set when DesignSync is installed. The value can be modified using SyncAdmin's General pane. The default value is "Check out read only when not locking", represented by a `FetchReadOnly` value of `dword:1`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\FetchReadOnly=dword:1
```

Allowed Values

Key	Description
FetchReadOnly=dword:0	Check out read/write copies in your work area, regardless of whether the checkout specifies lock.
FetchReadOnly=dword:1	Check out read only copies in your work area unless checkout lock is specified. (Default)

Related Topics

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Retain Defaults (FileRetainTime)

This setting determines whether to "retain last-modification timestamps" when SyncAdmin's Command Defaults pane is used.

Note: This option preference is also used by DSDFII GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\FileRetainTime=dword:0
```

Allowed Values

Key	Description
<code>FileRetainTime=dword:0</code>	Specifies that DesignSync commands should, by default, behave as if the <code>-noretain</code> option was specified. (Default)
<code>FileRetainTime=dword:1</code>	Specifies that DesignSync commands should, by default, behave as if the <code>-retain</code> option was specified.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

CR/LF processing (FilterCrLf)

This setting controls whether CR/LF is enabled or not. The `FilterCrLf` registry key is set when SyncAdmin's General Pane is used on Windows, to "Enable CR/LF Processing".

The default behavior is for a SyncServer to substitute CR/LF characters when transferring data to/from a DesignSync client on Windows.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNCH_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNCH_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNCH_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Options\FilterCrLf=dword:1
```

Allowed Values

Key	Description
<code>FilterCrLf=dword:0</code>	Specifies that DesignSync should disable CR/LF substitution.
<code>FilterCrLf=dword:1</code>	Specifies that DesignSync should enable CR/LF substitution. (Default)

Related Topics

Fetch Read Only (`FetchReadOnly`)

Default Fetch State (`DefaultFetchType`)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Hide Command in the GUI (ForbidCmds)

This setting controls which commands are displayed in the GUI. This can also be set from the Commands pane in SyncAdmin.

By default, no commands are restricted.

IMPORTANT: Because we do not document the specific strings that the command names use, you should use the SyncAdmin Commands interface to hide commands in the GUI. This reduces the possibility of introducing errors into your user or site registry files.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidCmds=""
```

Allowed Values

Key	Description
ForbidCmds=""	Show all commands in the GUI interface (Default)
ForbidCmds="<command> [command]>"	Hide specified commands, separated by a space, in the GUI interface to disallow their use by the user. Note: Hidden commands are still allowed in the command line interface in the GUI.

Hide or Move Command Options in the GUI (Forbid*Flags & Advanced*Flags)

These setting controls which commands options are displayed in the GUI and in which tab or section (General or Advanced) the option appears. This can also be set from the Command Options pane in SyncAdmin. The Forbidden key controls whether the option is displayed or hidden. The Advanced key controls whether the option is included in General or Advanced.

IMPORTANT: Because we do not document the specific strings that the command names and options use, you should use the SyncAdmin Command Options interface to move, show, or hide commands options in the GUI. This reduces the possibility of introducing errors into your user or site registry files.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedAddMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidAddMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedCancelFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidCancelFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedCiFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidCiFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedCoFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidCoFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedCompareFlags=""
```

DesignSync Data Manager Administrator's Guide

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidCompareFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedContentsFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidContentsFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedLockBranchFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidLockBranchFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedMcacheFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidMcacheFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedMkBranchFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidMkBranchFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedMkHrefFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidMkHrefFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedMkModFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidMkModFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedMvMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidMvMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedRetireFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidRetireFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedRmMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidRmMemberFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedSetFilterFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidSetFilterFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedStatusFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidStatusFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedTagFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidTagFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedUnlockFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidUnlockFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedVaultInfoFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidVaultInfoFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\AdvancedVhistoryFlags=""
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\ForbidVhistoryFlags=""
```

Allowed Values

The sample values included in this table are intended for reference use only. DesignSync strongly recommends using the Command Options interface to set or change these display values.

Key	Description
Advanced<command>Flags="<option> [option]"	Show the specified options for the listed command in the key in a space separated list. For example: AdvancedCiFlags="branch datatype hrefversions iflock keys retain retry tag trigarg "
Forbid<command>Flags="<option [option]>"	Hide specified command options, separated by a space, in the GUI interface to disallow their use by the user. For example: ForbidCiFlags="skip" Note: Hidden command options are still allowed in the command line interface in the GUI.

Set Help Mode (HelpMode)

DesignSync features the ability to tailor the ENOVIA Synchronicity Command Reference to your site usage by selecting a working methodology. When you select a working methodology, the Command Reference displayed when you select help within the command interface is tailored to method you selected. This corresponds to Command Reference Help Mode on the General tab.

Note: This does not affect the help you see when links within the help system are linked. The help links always link to the All version.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\Help\
HelpMode=all

Allowed Values

Key	Description
HelpMode=all	Links help and webhelp commands in the client interfaces to the All version of the ENOVIA Synchronicity Command Reference. This version contains all commands and options for DesignSync. (Default)
HelpMode=module	Links help and webhelp commands in the client interfaces to the Module version of the ENOVIA Synchronicity Command Reference. This version contains only the commands and options for DesignSync modules usage.
HelpMode=file	Links help and webhelp commands in the client interfaces to the File version of the ENOVIA Synchronicity Command Reference. This version contains only the commands and options for DesignSync file/vault-based usage.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

DesignSync Data Manager Administrator's Guide

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Process locked objects only (IfLock)

This setting controls the behavior of the `ci` command. The `IfLock` registry key is set when SyncAdmin's Command Defaults pane is used to set the "Checkin will process Locked Mode objects only (-iflock/-noiflock)" option.

The default is for `ci` to behave as though the **-noiflock** option was specified.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\IfLock=dword:0

Allowed Values

Key	Description
IfLock=dword:0	Specifies that the ci command behave as though the -noflock option was specified. (Default)
IfLock=dword:1	Specifies that the ci command behave as though the -iflock option was specified.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCIComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

DesignSync Data Manager Administrator's Guide

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

When retry is enabled and the `retryOnModuleCiFailureHook` trigger is not enabled, the `ModuleFailureRetryAttempts` setting determines the number of auto-retry attempts of the module checkin after failing with a communication connect failure. The value must be a number between one and 999, specified in hexadecimal. The value defaults to 60 retry attempts. Retry is enabled by launching a checkin with the `retry enabled` option. This option corresponds to options set in the "Retry on Module Checkin Failure" section on the Modules tab of SyncAdmin.

This setting is used in conjunction with the `ModuleFailureRetryInterval` registry setting. After it is determined that all retry attempts have not been exhausted, the next retry will occur after the time interval defined in the retry interval setting.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\Checkin\ModuleFailureRetryAttempts=dword:3c

Allowed Values

Key Value	Description
ModuleFailureRetryAttempts=dword:1	Sets number of retry attempts to 1. (Minimum Value)
ModuleFailureRetryAttempts=dword:3c	Sets number of retry attempts to 60. (Default)
ModuleFailureRetryAttempts=dword:<value>	Sets number of retry attempts to the specified value.
ModuleFailureRetryAttempts=dword:999	Sets number of retry attempts to 2457. (Maximum Value)

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

DesignSync Data Manager Administrator's Guide

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Checkin Error Retry Interval (ModuleFailureRetryInterval)

After checking the `ModuleFailureRetryAttempts` value, if a module checkin is to be retried, the `ModuleFailureRetryInterval` setting determines the wait time between retry attempts. The value must be a number between 100 and 9999, specified in hexadecimal. The value is specified in seconds and defaults to 900 seconds (dword:384). This option corresponds to options set in the "Retry on Module Checkin Failure" section on the Modules tab of SyncAdmin.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\Checkin\ModuleFailureRetryInterval=dword:384
```

Allowed Values

Key value	Description
ModuleFailureRetryInterval=dword:64	Sets the retry interval to 100. (Minimum Value)
ModuleFailureRetryInterval=dword:384	Sets the retry interval to 900. (Default)
ModuleFailureRetryInterval=dword:<value>	Sets the retry interval to the specified number of seconds in hexadecimal.
ModuleFailureRetryInterval=dword:270F	Sets the retry interval to 9999. (Maximum Value)

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

DesignSync Data Manager Administrator's Guide

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Use Checksum (NoChecksumMod)

DesignSync can register whether an object is locally modified using either the modification date or the file size and checksum information. By default, DesignSync uses the file size and CheckSum information to determine whether the file contents themselves have been modified.

Note: DesignSync clients prior to V6R2010x do not create Checksum metadata in the workspace. If compare is run on a workspace populated by a V6R2010x (or higher) client and then subsequently populated by older client the fetched version number and the checksum values will be out of sync with what the command expects. To avoid this situation, repopulate your workspace with a newer client, or disable checksum comparison.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\General\Options\NoChecksumMod

Allowed Values

Key	Description
NoChecksumMod=dword:0	Specifies that DesignSync should use the file size and CheckSum information to determine whether the file contents have been modified. (Default)
NoChecksumMod=dword:1	Specifies that DesignSync should consider any files with a date modification as locally modified

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

DesignSync Data Manager Administrator's Guide

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Exclude Warnings (NotifyExcludeByFilter)

This setting controls whether to warn the user when object is excluded when SyncAdmin's General pane is used. The registry key can also be set by the "Retries to Server in Maintenance Mode" on the SyncAdmin's General Performance pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notifications\NotifyExcludeByFilter=dword:1
```

Allowed Values

Key	Description
NotifyExcludeByFilter=dword:0	Disables warnings when an object is excluded

	by filter.
NotifyExcludeByFilter=dword:1	Enables warnings when an object is excluded by filter. (Default)

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Resetting the Populate Selector with the Version Option (PopulateModuleAllowSetSelector)

This setting determines how the version switch operates on the persistent module selector for a top-level module. When this setting is enabled (default), when the workspace is populated with the version switch, the persistent selector is updated to reflect the new version selected.

Note: Submodules are not affected by this option. Submodule version changes should be performed with the swap command set.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

`HKEY_CURRENT_USER\Software\Synchronicity\General\Options\PopulateNoEmptyDirs=dword:1`

Allowed Values

Key	Description
<code>PopulateNoEmptyDirs=dword:0</code>	Specifies that DesignSync commands should, by default, behave as if the <code>-emptydirs</code> option was specified.
<code>PopulateNoEmptyDirs=dword:1</code>	Specifies that DesignSync commands should, by

	default, behave as if the -noemptydirs option was specified. (Default)
--	--

Related Topics

Empty Directory Defaults (PopulateNoEmptyDirs)

Populate Full/Incremental (PopulateUseIncremental)

Empty Directory Defaults (PopulateNoEmptyDirs)

This setting determines whether to "populate empty directories" when SyncAdmin's Command Defaults pane is used. The `PopulateNoEmptyDirs` registry key is set when SyncAdmin's Command Defaults pane is used, for whether to "Populate empty directories".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Options\PopulateNoEmptyDirs=dword:1
```

Allowed Values

Key	Description
<code>PopulateNoEmptyDirs=dword:0</code>	Specifies that DesignSync commands should, by default, behave as if the -emptydirs option was specified.
<code>PopulateNoEmptyDirs=dword:1</code>	Specifies that DesignSync commands should, by default, behave as if the -noemptydirs option was specified. (Default)

Related Topics

Fetch Read Only (FetchReadOnly)

DesignSync Data Manager Administrator's Guide

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

There are two registry keys that control enforcement of a minimum checkin comment length. One, `RequireCiComment` indicates that a comment is required and the other, `MinCiCommentLength`, specifies the minimum length of the comment. These can also set by specifying "Minimum comment length," on SyncAdmin's General pane.

By default, checkin comments are not required. This default behavior is represented by a `RequireCiComment` value of `dword:0`. To require a minimum comment, set a `RequireCiComment` value of `dword:1`. Specify the length of the minimum comment as the `MinCiCommentLength` value, in hexadecimal.

For example, a minimum comment length of 20 characters is represented by:

```
MinCiCommentLength=dword:14
```

Note: If you set `RequireCiComment=dword:1` and do not specify a `MinCiCommentLength` value, the built-in default `MinCiCommentLength` value of 20 will be used.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\RequireCiComment=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\MinCiCommentLength=dword:<n>
```

Allowed Values

Key	Description
RequireCiComment=dword:0	Disables the requirement to enter a comment when performing a checkin operation. (Default)
RequireCiComment=dword:1	Enables a requirement to enter a comment when performing a checkin operation.
MinCiCommentLength=dword:<n>	Sets the minimum comment length of a comment in hexadecimal. If <code>RequireCiComment</code> is enabled and this value is not set, the minimum comment length is 20 (dword:14). If <code>RequireCiComment</code> is disabled, this key is ignored.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Exclude Restricted Characters (RestrictedCharacters and EnforceRestrictedCharacters)

This setting controls the list of characters that cannot be used in the natural path of a DesignSync object. This can also be set from the Exclude Lists pane in SyncAdmin. The characters that can be optionally restricted are used by DesignSync and other applications.

By default, no characters are restricted.

IMPORTANT:

- DSDFile users and administrators should not restrict the use of the # character.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

DesignSync Data Manager Administrator's Guide

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\EnforceRestrictedCharacters=dword:0
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\RestrictedCharacters="<ListOfCharacters>"
```

Allowed Values

Key	Description
EnforceRestrictedCharacters =dword:0	Do not restrict the characters which can be specified as the natural path for an object. (Default)
EnforceRestrictedCharacters =dword:1	Restrict the characters which can be specified as the natural path for an object, by disallowing users to create files or folders containing the characters on the list in the RestrictedCharacters registry key.
RestrictedCharacter=""	Does not restrict the characters which can be specified as the natural path for an object. (Same result as setting EnforceRestrictedCharacters=dword:0). (Default)
RestrictedCharacter="~! ?@# \$%^&* () ; : ` \ ' \" = [] / \ \ < > "	Restricts all special characters from being used in the natural path for an object. Note: The escape character and the quotation characters are escaped to avoid incorrect processing.
RestrictedCharacter="~! ?@# \$%^&* () ; : ` \ ' \" = [] / \ \ < > "	Restricts all special characters except for the # required for DSDFII users.

Keep Last-Version Information (RmVaultKeepVid)

This setting controls how the `rmvault` and `rmfolder` behave. The `RmVaultKeepVid` registry key is set when SyncAdmin's Command Defaults pane is used, for whether to "Keep last-version information when deleting vaults".

The default is for `rmvault` and `rmfolder` to behave as though the `-keepvid` option was specified.

Note: This option preference is also used by DSDFII GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

HKEY_CURRENT_USER\Software\Synchronicity\General\Options\RmVaultKeepVid=dword:1

Allowed Values

Key	Description
RmVaultKeepVid=dword:0	Specifies that the rmvault and rmfolder behave as though the -nokeepvid option was specified.
RmVaultKeepVid=dword:1	Specifies that the rmvault and rmfolder behave as though the -keepvid option was specified. (Default)

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

DesignSync Data Manager Administrator's Guide

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Save Local Versions (SaveLocal)

This setting controls how the populate command should behave for local saves. The `SaveLocal` registry key is set when SyncAdmin's Command Defaults pane is used, for how to handle local versions.

The default is for populate to behave as though **-savelocal fail** ("Fail if local versions exist") was specified.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

\$SYNC_USER_CFGDIR/UserRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\SaveLocal=dword:0

Allowed Values

Key	Description
SaveLocal=dword:0	Specifies that the populate command should behave, by default, as though -savelocal fail ("Fail if local versions exist") was specified. (Default)
SaveLocal=dword:1	Specifies that populate commands should, by default, behave as though -savelocal save ("Save local versions") was specified.
SaveLocal=dword:2	Specifies that populate commands should, by default, behave as though -savelocal delete ("Delete local versions") was specified.

Related Topics

Fetch Read Only (FetchReadOnly)

Default Fetch State (DefaultFetchType)

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Lists (SiteFilter and Filter)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

DesignSync Data Manager Administrator's Guide

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Exclude Lists (SiteFilter and Filter)

This setting controls the list of directories and files that should always be excluded from revision-control operations that support exclude filters (the **-exclude** option for DesignSync command-line commands and the **Exclude Filter** field for GUI operations). The registry key can also be set on the Exclude Lists pane in SyncAdmin.

Notes:

- The mkmod command while it does not have a **-exclude** option, does utilize the **Exclude Lists**, for the mkmod command's **-checkin** option.
- This registry key is applicable to the GUI clients. It does not include any command defaults set in the command line defaults system for the **-exclude** option.

The global exclude lists apply to revision control operations such as ci, co, populate, tag, and mkbranch. The global exclude lists do not apply by default to commands that list objects, such as ls, vhistory (**Tools=>Reports=>Version History**), compare (**Tools=>Reports=>Compare**), and contents (**Tools=>Reports=>Contents**). For these types of listing commands, exclude objects in the global exclude lists by specifying a null string ("") using the **-exclude** option or the **Exclude Filter** field. To exclude objects from the List View, select **View=>Hide Excluded Objects**)

By default, no site/project or user exclude lists are defined. To specify an exclude list, specify comma-separated file types as the `SiteFilter` or `Filter` value. For example:

```
SiteFilter="*.o,*.log"
```

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\SiteFilter="<value>"
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\Filter="<value>"
```

Allowed Values

Key	Description
<code>SiteFilter=""</code>	Specify a default site exclude filter.
<code>Filter=""</code>	Specify a list of recommended exclude filters which can be override at the user level.

Related Topics

Fetch Read Only (`FetchReadOnly`)

Default Fetch State (`DefaultFetchType`)

DesignSync Data Manager Administrator's Guide

Client Vault Fetch State (ClientVaultGetOnly)

Check-in Minimum Comment Length (RequireCiComment and MinCiCommentLength)

Check-in detection of copied workspace (AllowVaultRelocation)

Fetching retired data (CoFetchRequired)

Exclude Warnings (NotifyExcludeByFilter)

Retain Defaults (FileRetainTime)

Empty Directory Defaults (PopulateNoEmptyDirs)

Save Local Versions (SaveLocal)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Keep Last-Version Information (RmVaultKeepVid)

Commit Module Changes Immediately (CommitImmediate)

Process locked objects only (IfLock)

Resolve paths for mirrors (EnableRealMirrorPaths)

CR/LF processing (FilterCrLf)

Use Checksum (NoChecksumMod)

Set Help Mode (HelpMode)

Checkin Error Retry Attempts (ModuleFailureRetryAttempts)

Checkin Error Retry Interval (ModuleFailureRetryInterval)

General Options

Default Fetch State Options

Exclude Lists

Command Defaults

Keyword Expansion

Enable/Disable Exclusion Files (UseSyncExclude)

This setting controls whether the exclusion files feature is enabled or disabled. When the feature is disabled, any defined exclusion files or exclusion command operations are disabled. For more information using exclusion files, see DesignSync Data Manager User's Guide: Working with Exclude Files.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\UseSync
cExclude=dword:1
```

Allowed Values

Key	Description
UseSyncExclude=dword:0	Disabled the exclusion files feature.
UseSyncExclude=dword:1	Enables the exclusion files feature. (Default)

DesignSync Client Environment Registry Settings

Auto-creation of workspace root directory (AllowAutoRootCreation)

This setting controls automatic creation of workspace root directories. When enabled, auto-creation of the workspace root at specified location, specified with the workspace root path is allowed. The workspace root is used to establish context in the workspace when design data is created or populated into the workspace.

When not checked, the workspace root is not created when new data is placed in the workspace.

Note for modules: If there is no existing workspace root directory for the specified directory for a new module, the creation operation fails.

DesignSync Data Manager Administrator's Guide

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\AllowAutoRootCreation=dword:0
```

Allowed Values

Key value	Description
<code>AllowAutoRootCreation=dword:0</code>	Do not auto-create workspace root. (Default)
<code>AllowAutoRootCreation=dword:1</code>	Auto-create workspace root.

Related Topics

Workspaces

Workspace root path (`DefaultAutoRootPath`)

Enable the `setvault` command to set the module root directory (`EnsureRoot`)

[ENOVIA Synchronicity Command Reference: setroot](#)

[ENOVIA Synchronicity Command Reference: url root](#)

Cached file uniqueness (`CacheUseHostPost`)

Specified whether to include the host and port of the server when calculating the unique filename of a cached file or omit the host and port from the hashed filename calculation.

If you move servers or projects often, omitting the host and port from the hashed filename calculation will save users from having to update their workspaces, because the hashed filenames will not have changed. However, if more than one server has a particular vault path, then the same vault path on multiple servers will hash to the same cached filename.

By default, the host and port server are included.

Note: If you set the `CacheUseHostPort` registry key value to `<dword:0>`, you must not use the same vault path on more than one server.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\CacheUseHostPort=dword:1
```

Allowed Values

Key	Description
<code>CacheUseHostPort=dword:0</code>	Omit the server name and port when calculating the unique filename of a cached file.
<code>CacheUseHostPort=dword:1</code>	Include the server name and port when calculating unique filename of a cached file. (Default)

Related Topics

Default cache (WebObjectCache)

Project caches (Projects)

Symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode)

Enable Merged Mode (SavePreMergedFile)

Workspace links to Cache (PBFCEnabled)

Disable hard links when cache owner populates the cache (PBFCAllowUserToOwnFile)

General Options

Projects Options

Symbolic Links Options

Workspace root path (DefaultAutoRootPath)

DesignSync Data Manager Administrator's Guide

This setting controls the workspace root path. This key can also be set from the Workspaces pane.

When the Auto-creation of workspace root directory is enabled, the `DefaultAutoRootPath` registry key is used from SyncAdmin, to set the workspace root path.

By default, the root path is set to `..` indicating one level above where the top-level folder for the design data resides in the workspace. To set a different workspace root path, specify the path. Path names can be absolute or relative.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\DefaultAutoRootPath=".."
```

Allowed Values

Key value	Description
<code>DefaultAutoRootPath=".."</code>	Indicates one level above workspace module. (Default)
<code>DefaultAutoRootPath=<value></code>	Changes workspace root path to specified path value.

Related Topics

Auto-creation of workspace root directory (`AllowAutoRootCreation`)

Workspaces

[ENOVIA Synchronicity Command Reference: setroot](#)

[ENOVIA Synchronicity Command Reference: url root](#)

Enable the `setvault` command to set the module root directory (`EnsureRoot`)

This setting controls whether the `setvault` command, when run on a module workspace, sets the module workspace root. When enabled, the `setvault` command sets the workspace root one level higher than the workspace folder that `setvault` is run against.

This registry setting pertains to DesignSync servers and clients. To modify registry values programmatically, use the `sregistry` command set.

The settings can be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the `SyncServer` is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\EnsureRoot=dword:1
```

Allowed Values

Key value	Description
<code>EnsureRoot=dword:0</code>	Do not set a workspace root when <code>setvault</code> is run on a module.
<code>EnsureRoot=dword:1</code>	When <code>setvault</code> is run on a module, automatically designate the containing folder for the module as the module workspace root. (Default)

Related Topics

[Workspaces](#)

[Auto-creation of workspace root directory \(AllowAutoRootCreation\)](#)

[Workspace root path \(DefaultAutoRootPath\)](#)

ENOVIA Synchronicity Command Reference: setroot

ENOVIA Synchronicity Command Reference: url root

Disable hard links when cache owner populates the cache (PBFCAllowUserToOwnFile)

The `PBFCAllowUserToOwnFile` setting controls whether the user who fetched the cached versions can link to them with hard links. There are no performance implications either way, but when the owner is viewing the workspace containing the cache hard links, there are no obvious indications that these are cache links, not actual files. This can lead to problems if the owner then tries to manipulate the file.

By default, this setting is disabled, meaning that when the owner of a cache file creates a workspace object linking to this file in the cache, the cache link is created as a symbolic link, rather than a hard link.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\Cache\PBFCAllowUserToOwnFile=dword:0
```

Allowed Values

Key	Description
<code>PBFCAllowUserToOwnFile=dword:0</code>	Allows owner of a cache file to create a workspace object linking to this file in the cache as a symbolic link, rather than a hard link. (Default)
<code>PBFCAllowUserToOwnFile=dword:1</code>	Allows the owner of a cache file to create a workspace object linking to this file in the cache as hard link to the cache.

Related Topics

Default cache (`WebObjectCache`)

Project caches (Projects)

Cached file uniqueness (CacheUseHostPost)

Symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode)

Symbolic link handling (ManagedLinkInMirrorMode)

Enable Merged Mode (SavePreMergedFile)

Workspace links to Cache (PBFCEnabled)

General Options

Projects Options

Symbolic Links Options

Workspace links to Cache (PBFCEnabled)

The `PBFCEnabled` setting is used to enable creation of hard links, instead of symbolic links, to file caches. For more information on using symbolic or hard links, see [Links Options](#).

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\Cache\PBFCEnabled=dword:0
```

Allowed Values

Key value	Description
<code>PBFCEnabled=dword:0</code>	Enables the creation of symbolic links instead of hard links. (Default)
<code>PBFCEnabled=dword:1</code>	Enables the creation of hard links.

Related Topics

Default cache (`WebObjectCache`)

Project caches (`Projects`)

Cached file uniqueness (`CacheUseHostPost`)

Symbolic link handling (`SymbolicLinkMode` and `DirSymbolicLinkMode`)

Symbolic link handling (`ManagedLinkInMirrorMode`)

Enable Merged Mode (`SavePreMergedFile`)

Disable hard links when cache owner populates the cache (`PBFCAAllowUserToOwnFile`)

General Options

Projects Options

Symbolic Links Options

Project Registry Keys (`ProjectCacheTclScript`) and (`Projects`)

A DesignSync project is an infrastructure designed to support a real-world design project. A project can be a module or a collection of vault objects. Projects provide the ability to use caching. The project definition stores key information, such as the Vault URL for the project and the cache location. This information can be stored in one of two different ways for both modules and file-based projects:

- Using a cache script to store the definition
- Defining the project in the registry settings

Using a Cache Script to Store the Definition

Users define or calculate the Vault URL and cache directory in a TCL script which is specified by the `ProjectCacheTclScript` registry key. The project cache mappings in the script are loaded into memory on an as-needed basis; when an appropriate command is run with the `-share` option. This provides performance improvements over defining the keys directly in the registry for systems with a large number of projects defined.

Workspaces associated with a project's `ProjectVault` will use the project's `ProjectCache` for **-share** operations.

By default, no projects are defined.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\Cache\ProjectCacheTclScript="Tcl-Procedure"
```

Allowed Values

Key Value	Description
<code>ProjectCacheTclScript="Tcl-Procedure"</code>	Defines the script name containing the project cache mappings. If the script is defined and there are also projects with the same name defined using the registry keys below, the registry keys definition takes precedence over the script definition. The file should be placed in the appropriate custom directory. For more information on defining the TCL script, see How DesignSync Determines the Correct Project Cache

Example

There is an example script provided in the examples directory:

```
$<SYNC_DIR>/share/examples/doc/stclguide/syncLocateProjectCache.tcl
```

Defining the project in the Registry Settings

Users define the project properties directly in the registry key files. These options to control project location information can also be set on SyncAdmin's Project pane.

Workspaces associated with a project's `ProjectVault` will use the project's `ProjectCache` for **-share** operations.

DesignSync Data Manager Administrator's Guide

By default, no projects are defined.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss

Registry Keys

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\<project_name>\@=none
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\<project_name>\ProjectVault="project vault URL"
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\<project_name>\ProjectCache="cache path"
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\<project_name>\ProjectDescription="project description"
```

Allowed Values

Key Value	Description
<code><project_name>\@=none</code>	Defines the project name. The value of project_name must be constant for all the related keys.
<code><project_name>\ProjectVaultURL=" project vault URL"</code>	Defines the SyncURL associated with the project. This is used when
<code><project_name>\ProjectCache=" cache path"</code>	Defines location of project cache. Applicable only for UNIX.
<code><project_name>\ProjectDescription=" project description"</code>	Defines project description seen by users.

Example

This is an example of setting the Project registry keys for a files-based project named "Sportster."

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\Sportster\@=none
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\Sportster\ProjectVault="sync://ABCo.com:2647/Projects/Sportster"
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\Sportster\ProjectCache="/home/syncmgr/caches/Sportster"
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Projects\Sportster\ProjectDescription="Cache for Sportster project"
```

Related Topics

What Is a Project?

How DesignSync Determines the Correct Project Cache

Enable Merged Mode (SavePreMergedFile)

The `SavePreMergedFile` setting is used to enable saving backup copies of files in the workspace when a merge is performed. By default, DesignSync saves a backup file if there is a merge conflict.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\SavePreMergedFile=dword:1
```

Allowed Values

Key	Description
<code>SavePreMergedFile=dword:0</code>	Never save a backup file when performing a merge operation.
<code>SavePreMergedFile=dword:1</code>	Save a backup file if there is a merge conflict during a merge operation. (Default)
<code>SavePreMergedFile=dword:2</code>	Save a backup file when performing a merge operation.

Related Topics

DesignSync Data Manager Administrator's Guide

Default cache (WebObjectCache)

Project caches (Projects)

Cached file uniqueness (CacheUseHostPost)

Symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode)

Workspace links to Cache (PBFCEnabled)

Disable hard links when cache owner populates the cache (PBFCAllowUserToOwnFile)

General Options

Projects Options

Symbolic Links Options

Symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode)

Specifies how symbolic links are handled. The options to control symbolic link handling can also be controlled through "Revision Control Symbolic Links" on SyncAdmin's Symbolic Links pane.

The default behavior, which corresponds to "Treat links as copies of the files/directories to which they point" in SyncAdmin, is represented by default values of `SymbolicLinkMode=dword:0` for links to files and `DirSymbolicLinkMode=dword:0` for links to directories.

The SyncAdmin option to "Store links to files as links and remember where they point to on check out. Treat links to directories as copies of the directories to which they point" is represented by values of `SymbolicLinkMode=dword:1` for links to files and `DirSymbolicLinkMode=dword:0` for links to directories.

The SyncAdmin option to "Store both links to files and directories as links and remember where they point to on check out" is represented by values of `SymbolicLinkMode=dword:1` for links to files and `DirSymbolicLinkMode=dword:1` for links to directories.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use

the `syncdadadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\SymbolicLinkMode=dword:0`

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\DirSymbolicLinkMode=dword:0`

Allowed Values

Key	Description
<code>SymbolicLinkMode=dword:0</code>	Treat links to files as copies of the files. (Default)
<code>SymbolicLinkMode=dword:1</code>	Treat links to files as links to their original locations.
<code>DirSymbolicLinkMode=dword:0</code>	Treat links to directories as copies of the directories to which they point. (Default)
<code>DirSymbolicLinkMode=dword:1</code>	Treat links to directories as links to their original locations.

Related Topics

Default cache (`WebObjectCache`)

Project caches (`Projects`)

Cached file uniqueness (`CacheUseHostPost`)

Symbolic link handling (`ManagedLinkInMirrorMode`)

Enable Merged Mode (`SavePreMergedFile`)

Workspace links to Cache (`PBFCEnabled`)

Disable hard links when cache owner populates the cache (`PBFCAllowUserToOwnFile`)

General Options

Projects Options

Symbolic Links Options

Default cache (`WebObjectCache`)

DesignSync Data Manager Administrator's Guide

Changes the default cache directory. The default cache directory is initially set during DesignSync client installation. This option can also be set by the "Default cache directory" option in SyncAdmin's General pane.

Specify the *<path>* as an absolute path, available to all DesignSync users on your LAN. For example:

```
WebObjectCache="/home/syncmgr/sync_cache"
```

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Directories\  
WebObjectCache="<path>"
```

Allowed Values

Key Value	Description
<i><User's_Home_Directory>/sync_cache</i>	Default cache value for UNIX systems.
<code>WebObjectCache=c:\Users\<UserName>\AppData\Roaming\Synchronicity\cache</code>	Default cache value for Windows systems.

Related Topics

Project caches (Projects)

Cached file uniqueness (CacheUseHostPost)

Symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode)

Enable Merged Mode (SavePreMergedFile)

Workspace links to Cache (PBFCEnabled)

Disable hard links when cache owner populates the cache (PBFCAAllowUserToOwnFile)

General Options

Projects Options

Symbolic Links Options

Modules Registry Settings

Added module roots (ModuleRoots)

This setting controls the path to the module root directory. The `ModuleRoots` registry key is set when the Modules Roots pane is used from SyncAdmin, to set "Module Roots."

The `RootSubKey` value is the url-encoded URL of the module root directory.

The `RootDirectoryPath` is the path to the module root directory for all of the module instances within that directory hierarchy.

For example:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\ModuleRoots\
file%3A%2F%2F%2Fc%7C%2FBUILD%2FDesignSync_src\ModuleRootPath="c
:\BUILD\DesignSync_src"
```

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\ModuleRoots\RootSubKey\
ModuleRootPath="RootDirectoryPath"
```

Related Topics

Always show tags (`AlwaysShowTags`)

Auto-creation of module root directory (`AllowAutoRootCreation`)

Enable Legacy Mode (`UseLegacyModules`)

Href Mode Behavior (`HrefModeChangeWithTopStaticSelector`)

DesignSync Data Manager Administrator's Guide

Ignore invalid Module Cache paths (IgnoreInvalidPaths)

Legacy modules without hrefs (FetchModuleNoHrefsAsModule)

Link to submodules in the module cache (AllowIntermediateLinks)

Maximum number of versions displayed (MaxVersionsDisplayed)

Module cache mode (Mode)

Module cache paths (Paths)

Module root path (DefaultAutoRootPath)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Link to submodules in the module cache (AllowIntermediateLinks)

The `AllowIntermediateLinks` registry key is used to control how hierarchically linked submodules are handled when populating an mcache. It can also be set using SyncAdmin's Modules pane "Allow Linking to Submodules in the mcache".

By default, this option is disabled, meaning that hierarchical referenced submodules are not populated into the mcache.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\Mcache\AllowIntermediateLinks=dword:0
```

Allowed Values

Key value	Description
AllowIntermediateLinks=dword:0	Hierarchical referenced submodules are not populated into the mcache. (Default)
AllowIntermediateLinks=dword:1	Hierarchical referenced submodules are populated into the mcache.

Related Topics

Always show tags (AlwaysShowTags)

Enable Legacy Mode (UseLegacyModules)

Href Mode Behavior (HrefModeChangeWithTopStaticSelector)

Ignore invalid Module Cache paths (IgnoreInvalidPaths)

Legacy modules without hrefs (FetchModuleNoHrefsAsModule)

Maximum number of versions displayed (MaxVersionsDisplayed)

Module cache mode (Mode)

Module cache paths (Paths)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Always show tags (AlwaysShowTags)

The AlwaysShowTags registry key stores a comma separated tag list. You can use this key to manually maintain a list of tags for modules or module members that, if they exist, should always be displayed. Each module instance has a showtags property with a comma separated list of tags. These two lists, together with any blended selector (combining a base tag with members from other module versions) defined for the workspace are considered "tags of interest" for the module members. This key is always manually updated. There is no corresponding field in SyncAdmin. For more information on blended selectors, see the *DesignSync Data Manager User's Guide: Module Member Tags*.

Note: You can manually update showtags property for a module instance with the url setprop command, but note that url setprop does not append to the value of the key, but

replaces it. To be sure you are updating it correctly, use `url getprop` to get the list of interesting tags, and use that list as the basis for your changes.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesignSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\AlwaysShowTags=<"tag>[,<tag>...]"

Related Topics

Enable Legacy Mode (`UseLegacyModules`)

Href Mode Behavior (`HrefModeChangeWithTopStaticSelector`)

Ignore invalid Module Cache paths (`IgnoreInvalidPaths`)

Legacy modules without hrefs (`FetchModuleNoHrefsAsModule`)

Link to submodules in the module cache (`AllowIntermediateLinks`)

Maximum number of versions displayed (`MaxVersionsDisplayed`)

Module cache mode (`Mode`)

Module cache paths (`Paths`)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(`PopulateDoVersionExtendedInfo`\)](#)

Legacy modules without hrefs (`FetchModuleNoHrefsAsModule`)

The `FetchModuleNoHrefsAsModule` registry key determines whether legacy module without hierarchical references are displayed as modules or folders in the workspace. It can also be set using SyncAdmin's Modules pane "Fetch legacy modules with no hierarchical references as modules".

By default, fetched legacy modules that do not have any hierarchical references (at the time they are fetched) are represented in the local workspace as modules.

For those fetched modules to instead be represented in the local workspace as regular (non-module) folders, set `FetchModuleNoHrefsAsModule` to `dword:0`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\FetchModuleNoHrefsAsModule=dword:1
```

Allowed Values

Key value	Description
<code>FetchModuleNoHrefsAsModule=dword:0</code>	Fetched legacy modules that do not have any hierarchical references (at the time they are fetched) are represented in the local workspace as folders.
<code>FetchModuleNoHrefsAsModule=dword:1</code>	Fetched legacy modules that do not have any hierarchical references (at the time they are fetched) are represented in the local workspace as modules.(Default)

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

[Enable Legacy Mode \(UseLegacyModules\)](#)

[Href Mode Behavior \(HrefModeChangeWithTopStaticSelector\)](#)

Ignore invalid Module Cache paths (IgnoreInvalidPaths)

Link to submodules in the module cache (AllowIntermediateLinks)

Maximum number of versions displayed (MaxVersionsDisplayed)

Module cache mode (Mode)

Module cache paths (Paths)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Href Mode Behavior (HrefModeChangeWithTopStaticSelector)

The `HrefModeChangeWithTopStaticSelector` registry key controls how the module hierarchy is processed by DesignSync operations when using the hrefmode "Normal," with a static selector on the selected module. It can also be set using SyncAdmin "Change traversal mode with static selector on top level module" in the Modules pane.

By default, this setting is enabled. When the setting is enabled, the populate operation will start the hierarchy traversal (from the top level module) in static mode if the selector for the top level module is a static selector (such as a version tag).

When the setting is disabled, the traversal will be in 'normal' mode (from the top level), regardless of the type of selector of the top level module (static or dynamic)

For more information and an example of how a module hierarchy is populated into a workspace, see the *ENOVIA Synchronicity DesignSync Data Manager User's Guide*, Module Hierarchy: An Alternate Method of Module Hierarchy Traversal..

Note: This key is applicable at both the user and server level, however for DesignSync Web UI hierarchical queries and email subscriptions, you must set the registry key at the server level. When the SyncAdmin interface is run in Site mode, it automatically sets the registry key for the site (therefore at the server level, rather than the user level.)

This registry setting pertains to DesignSync clients as well as SyncServers. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's

`$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a

`$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\HrefModeChangeWithTopStaticSelector=dword:1`

Allowed Values

Key value	Description
<code>HrefModeChangeWithTopStaticSelector=dword:0</code>	Using hrefmode normal, the hrefs for the selected module are evaluated. The selector for the selected module is not considered when processing hrefs.
<code>HrefModeChangeWithTopStaticSelector=dword:1</code>	Using hrefmode normal, if the selected module uses a static selector, all hrefs are followed statically. (Default)

Related Topics

Module Hierarchy

Always show tags (`AlwaysShowTags`)

Enable Legacy Mode (`UseLegacyModules`)

Ignore invalid Module Cache paths (`IgnoreInvalidPaths`)

Legacy modules without hrefs (`FetchModuleNoHrefsAsModule`)

Link to submodules in the module cache (`AllowIntermediateLinks`)

Maximum number of versions displayed (`MaxVersionsDisplayed`)

Module cache mode (`Mode`)

Module cache paths (`Paths`)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Ignore invalid Module Cache paths (IgnoreInvalidPaths)

When populate encounters an invalid module cache path, there are two possible ways to handle it; it can stop processing the command and return an error, or it can skip the object that experiencing the error and continue processing the command. This property can also be controlled from the SyncAdmin Module Cache panel.

By default, `IgnoreInvalidPaths` registry key is disabled, meaning that when populate encounters an invalid module cache path, it stops processing the command and returns an error.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\Mcache\IgnoreInvalidPaths=dword:0
```

Allowed Values

Key value	Description
<code>IgnoreInvalidPaths=dword:0</code>	Stop processing command on module cache path error. (Default)
<code>IgnoreInvalidPaths=dword:1</code>	Continuing command processing on other objects after receiving a module cache path error.

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

[Enable Legacy Mode \(UseLegacyModules\)](#)

[Href Mode Behavior \(HrefModeChangeWithTopStaticSelector\)](#)

Legacy modules without hrefs (FetchModuleNoHrefsAsModule)

Link to submodules in the module cache (AllowIntermediateLinks)

Maximum number of versions displayed (MaxVersionsDisplayed)

Module cache mode (Mode)

Module cache paths (Paths)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Maximum number of versions displayed (MaxVersionsDisplayed)

The `MaxVersionsDisplayed` registry key sets how many versions are shown on a module branch when browsing the server. It can also be set using SyncAdmin's Modules pane "Maximum number of versions displayed".

By default, the last 50 versions on a module branch are shown, when browsing a module on a server.

Specify the `MaxVersionsDisplayed` value in hexadecimal. The default value of 50 decimal is represented in hexadecimal as `dword:32`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\MaxVersionsDisplayed=dword:32
```

Allowed Values

Key Value	Description
MaxVersionsDisplayed=dword:32	Sets number of versions shown on branch to 50. (Default)
MaxVersionsDisplayed=dword:<value>	Set desired number of versions shown on branch in hexadecimal.

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

[Enable Legacy Mode \(UseLegacyModules\)](#)

[Href Mode Behavior \(HrefModeChangeWithTopStaticSelector\)](#)

[Ignore invalid Module Cache paths \(IgnoreInvalidPaths\)](#)

[Legacy modules without hrefs \(FetchModuleNoHrefsAsModule\)](#)

[Link to submodules in the module cache \(AllowIntermediateLinks\)](#)

[Module cache mode \(Mode\)](#)

[Module cache paths \(Paths\)](#)

[Modules Options](#)

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Module cache mode (Mode)

Identifies the method (mode) that the **populate** operation uses to fetch modules when users do not specify the **-mcachemode** option. The Mode registry key is set when SyncAdmin's Modules Cache pane is used, to select the "Default module cache mode".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client

(exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_CURRENT_USER\Software\Synchronicity\Client\Mcache\Mode="link"

Allowed Values

Key value	Description
Mode="link"	<p>For each module it finds in the module cache, the populate operation sets up a symbolic link from your work area to the base directory of the module in the module cache.</p> <p>Default for UNIX platforms.</p> <p>This corresponds to the "Link to modules in the module cache" set for Default module cache mode.</p>
Mode="copy"	<p>For each legacy module it finds in the module cache, the populate operation copies the module to your work area.</p> <p>Default for Windows platforms.</p> <p>This mode is ignored for non-legacy modules, which are fetched from the server.</p> <p>This corresponds to the "Copy modules from the module cache" set for Default module cache mode.</p>
Mode="server"	<p>Fetch modules from the server. Causes the populate operation to fetch modules from the server.</p> <p>This corresponds to the "Fetch modules from the server" set for Default module cache mode.</p>

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

[Enable Legacy Mode \(UseLegacyModules\)](#)

[Href Mode Behavior \(HrefModeChangeWithTopStaticSelector\)](#)

[Ignore invalid Module Cache paths \(IgnoreInvalidPaths\)](#)

[Legacy modules without hrefs \(FetchModuleNoHrefsAsModule\)](#)

DesignSync Data Manager Administrator's Guide

[Link to submodules in the module cache \(AllowIntermediateLinks\)](#)

[Maximum number of versions displayed \(MaxVersionsDisplayed\)](#)

[Module cache paths \(Paths\)](#)

[Modules Options](#)

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

[Populate Version Extended Info \(PopulateDoVersionExtendedInfo\)](#)

Module cache paths (Paths)

This setting controls which module cache paths are used.

The `Paths` registry key is set when the Modules pane is used from SyncAdmin, to set "Default module cache paths".

By default, no module cache paths are defined. To set default module cache paths, specify the `<value>` as a list of space-separated path names. Path names can be absolute or relative.

For example:

```
Paths="/home/syncmgr/mcaches ~/mcaches"
```

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\Mcache\Paths="<value> [...]"
```

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

Enable Legacy Mode (UseLegacyModules)

Href Mode Behavior (HrefModeChangeWithTopStaticSelector)

Ignore invalid Module Cache paths (IgnoreInvalidPaths)

Legacy modules without hrefs (FetchModuleNoHrefsAsModule)

Link to submodules in the module cache (AllowIntermediateLinks)

Maximum number of versions displayed (MaxVersionsDisplayed)

Module cache mode (Mode)

Modules Options

ENOVIA Synchronicity Command Reference: url setprop

ENOVIA Synchronicity Command Reference: url getprop

Populate Version Extended Info (PopulateDoVersionExtendedInfo)

Populate Version Extended Info (PopulateDoVersionExtendedInfo)

This setting controls whether the appropriate command version has been entered when a populate command is entered. The setting is set to `dword:1` by default, which generates an informational message if the incorrect version command is entered.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Keys

General\Commands\General\PopulateDoVersionExtendedInfo=dword:1

Allowed Values

Key value	Description
-----------	-------------

<code>PopulateDoVersionExtendedInfo=dword:1</code>	Enables a check of the populate command for whether the correct version command has been used. If the version command is incorrect an informational message explaining the issue will be displayed.
<code>PopulateDoVersionExtendedInfo=dword:0</code>	Disables the populate command version check.

Related Topics

Always show tags (`AlwaysShowTags`)

Enable Legacy Mode (`UseLegacyModules`)

Href Mode Behavior (`HrefModeChangeWithTopStaticSelector`)

Ignore invalid Module Cache paths (`IgnoreInvalidPaths`)

Legacy modules without hrefs (`FetchModuleNoHrefsAsModule`)

Link to submodules in the module cache (`AllowIntermediateLinks`)

Maximum number of versions displayed (`MaxVersionsDisplayed`)

Module cache mode (`Mode`)

Module cache paths (`Paths`)

Modules Options

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

Show Hrefs Status in Report Normal Mode (`ShowHrefsNeedCheckinStatus`)

The `ShowHrefsNeedCheckinStatus` registry key controls whether the report mode for the `showstatus` command shows the hierarchical reference "- needs checkin" status in the workspace.

By default, this setting is disabled meaning that the hierarchical reference status "needs checkin" is only displayed when `show status` is run in the verbose report mode. When the setting is enabled, the `show status` operation includes the "needs checkin" status in both normal and verbose report modes.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\ShowHrefsNeedCheckinStatus=dword:0`

Allowed Values

Key value	Description
<code>ShowHrefsNeedCheckinStatus=dword:0</code>	When using the Show Status command (<code>showstatus</code>), the status "needs checkin" for hrefs for the selected module is included in the verbose report mode only. (Default)
<code>ShowHrefsNeedCheckinStatus=dword:1</code>	When using the Show Status command (<code>showstatus</code>), all href status messages for the selected module are included in both normal and verbose report modes.

Related Topics

`showstatus` command (*DesignSync Command Reference*)

Href Mode Behavior (`HrefModeChangeWithTopStaticSelector`)

Enable Legacy Mode (`UseLegacyModules`)

The `UseLegacyModules` registry key is used to enable/disable legacy module mode. It can also be set using SyncAdmin's Modules pane "Activate legacy hcm command set."

By default, the legacy command set is disabled. For more information on using legacy modules, see the *ENOVIA Synchronicity DesignSync HCM User's Guide*. To enable legacy module mode, which includes full editing for legacy modules, set `UseLegacyModules` to `dword:1`.

Note: You may require a license that includes legacy module support.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\UseLegacyModules=dword:0
```

Allowed Values

Key value	Description
<code>UseLegacyModules=dword:0</code>	Disable legacy module commands. Legacy modules can be still viewed and used within a module hierarchy. (Default)
<code>UseLegacyModules=dword:1</code>	Enable legacy module commands. Legacy modules can be created and modified.

Related Topics

[Always show tags \(AlwaysShowTags\)](#)

[Href Mode Behavior \(HrefModeChangeWithTopStaticSelector\)](#)

[Ignore invalid Module Cache paths \(IgnoreInvalidPaths\)](#)

[Legacy modules without hrefs \(FetchModuleNoHrefsAsModule\)](#)

[Link to submodules in the module cache \(AllowIntermediateLinks\)](#)

[Maximum number of versions displayed \(MaxVersionsDisplayed\)](#)

[Module cache mode \(Mode\)](#)

[Module cache paths \(Paths\)](#)

[Modules Options](#)

[ENOVIA Synchronicity Command Reference: url setprop](#)

[ENOVIA Synchronicity Command Reference: url getprop](#)

Populate Version Extended Info (PopulateDoVersionExtendedInfo)**Vendor Objects Registry Settings****Cadence DFII Integration Registry Settings****Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)**

This setting determines what is set as the default comment. The AutoCheckInComment key is used for silent check-in operations initiated by the Cadence GDM system.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\ExtensionTypes\Cadence\AutoCheckInComment=autocheckin
```

Allowed Values

Key Value	Description
AutoCheckInComment=autocheckin	The default value is "autocheckin" which meets a comment length minimum of 10 characters, a typical minimal comment length.
AutoCheckInComment="string value"	You may enter any string value comment you like to use as the default comment. If you use spaces in the comment, you must put quotes around the string. Note: You cannot specify a return character or a line feed character in the comment field.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

DesignSync Data Manager Administrator's Guide

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Non-view folders in cell directories (Cadence NonView Folder)

The `Cadence NonView Folder` determines whether recursive operations traverse into view level directories within Cadence cells that do not contain views (views as defined in the Cadence data registry).

With `Cadence NonView Folder` set to `dword:1`, recursive operations, such as `ci`, `co`, `populate`, `tag`, `ls`, `cancel`, and `unlock` will traverse into these folders and operate on the files therein.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\AllowRecursion\  
Cadence NonView Folder=dword:0
```

Allowed Values

Key Value	Description
Cadence NonView Folder=dword:0	Recursive operations to not traverse into Cadence view folders if the folder does not contain a view. (Default)
Cadence NonView Folder=dword:1	Recursive operations traverse into Cadence view folders to operate on the files contained within them.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Cadence Design Systems' Collections (CadenceView)

The `CadenceView` key indicates whether Cadence object recognition is enabled or disabled. Cadence object recognition can be enabled during client installation or enabled or disabled by SyncAdmin's Third Party Integration pane. By default, Cadence object recognition is disabled.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In

DesignSync Data Manager Administrator's Guide

order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\ExtensionTypes\CadenceView=dword:0
```

Allowed Values

Key Value	Description
CadenceView=dword:0	Cadence object recognition is disabled. (Default)
CadenceView=dword:1	Cadence object recognition is enabled.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Cadence non-collection members (Cadence View Folder)

The `Cadence View Folder` key determines whether revision control operations recurse into Cadence cell view folders and, by default, operate on non-collection members. This setting affects only Cadence collections. By default, revision control operations do not recurse into Cadence cell view folders.

With `Cadence View Folder` set to `dword:1`, `ci`, `co`, `populate`, `tag`, `ls`, `cancel`, and `unlock` operations will traverse into view folders and operate on files that are not part of the Cadence collection. See [Managing Non-Collection Objects](#) for more information.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stlc` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\AllowRecursion\
Cadence View Folder=dword:0
```

Allowed Values

Key Value	Description
<code>Cadence View Folder=dword:0</code>	Revision control operations do not recurse into Cadence cell view folders. (Default)
<code>Cadence View Folder=dword:1</code>	Revision control operations recurse into Cadence cell view folders and operate on non-collection members.

Related Topics

SCC Plug-in Auto-Checkin Default Comment ([DSVS\AutoCheckInComment](#))

Cadence Auto-Checkin Default Comment ([Cadence\AutoCheckInComment](#))

Cadence Design Systems' Collections ([CadenceView](#))

Cadence View Depth ([MaxDepth](#))

[fisrvDoesOp optimization \(FisrvDoesOp\)](#)

DesignSync Data Manager Administrator's Guide

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Library Manager Display Additional Columns (ExtendCadenceLMColumns)

The `ExtendCadenceLMColumns` key controls the display of DesignSync specific information in the Library Manager. By default, these columns are enabled meaning that the Library Manager displays these four columns:

- **WS Status - Workspace status.** For a full description of available workspace status values, see *ENOVIA Synchronicity Command Reference: ls* command: Report Options.
- **Server Status - Server Status.** For a full description of available server status values, see *ENOVIA Synchronicity Command Reference: ls* command: Report Options.
- **Tags - Lists version tags of a managed object.** The column lists tags in the order of Most-Recent to Oldest (the reverse order of when the tags were added).
- **Member Of - Displays the instance name for module members.** If an object does not belong to a module, this field is blank.

Note: There is some overlap between the WS Status information and the Server Status information, however both have distinct information, so you may need both values to see a proper representation of the object status.

Because the status requires information from the server, there is a performance implication for displaying these fields.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In

order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Because this registry setting controls how DesignSync interacts with the Cadence Library Manager, you may also need to restart the DSDFI client for the changes to take effect.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\Extend
CadenceLMColumns=dword:1
```

Allowed Values

Key Value	Description
ExtendCadenceLMColumns=dword:0	Cadence library manager does not display the DesignSync specific fields listed above.
ExtendCadenceLMColumns=dword:1	Cadence library manager displays the DesignSync information fields listed above. (Default)

Related Topics

Non-view folders in cell directories (Cadence NonView Folder)

Cadence Design Systems' Collections (CadenceView)

Cadence non-collection members (Cadence View Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

The `fisrvDoesOp` key determines whether operations (such as checkins and checkouts) go through the Cadence GDM system or spawn a separate `stcl` process. Using the `fisrvDoesOp` optimization can improve performance and reliability. It is enabled by default. This optimization option can also be modified from the SyncAdmin's Third Party Integration pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In

DesignSync Data Manager Administrator's Guide

order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes\Cadence\FisrvDoesOp=dword:1
```

Allowed Values

Key Value	Description
FisrvDoesOp=dword:0	Do not use the performance optimization.
FisrvDoesOp=dword:1	Use the FisrvDoesOp performance optimization. (Default)

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Cadence client compatibility mode setting (ManageViewManifest)

The `ManageViewManifest` key determines whether compability with DesignSync 4.x clients is enabled or disabled. If this option is enabled, a `.<view>.sync.cds.synccmd` file is maintained for Cadence cell views by the DesignSync 5.0hf+ client operations, which keeps the vault data usable by 4.x clients in other workspaces.

Note: This compatibility mode is only necessary if your use model requires DesignSync 4.x and 5.0+ clients access to the same vault data, and the data includes Cadence cell view data. Mirrors may be affected by this requirement.

For more information about Cadence compatibility mode, contact customer support.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimiz
ations\ManageViewManifest=dword:0
```

Allowed Values

Key Value	Description
<code>ManageViewManifest=dword:0</code>	Compatibility mode between 4.x and 5.x clients is not maintained. (Default)
<code>ManageViewManifest=dword:1</code>	Compatibility mode between 4.x and 5.x clients is maintained.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

DesignSync Data Manager Administrator's Guide

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Cadence View Depth (MaxDepth)

The `MaxDepth` key defines the maximum depth of Cadence libraries. This maximizes the efficiency of browsing by limiting how far to scan up a directory hierarchy if a folder is part of a Cadence library.

Note: If you are using ADE-XL viewtypes, set the value to 8.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Cadence\MaxDepth=dword:<n>
```

Allowed Values

Key Value	Description
<code>MaxDepth=dword:04</code>	Depth of Cadence library browsing is 4; 1 level below the view directory. (Default)
<code>MaxDepth=dword:##</code>	Depth of Cadence library browsing. Can be any integer value up to 99.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Synopsys Custom Designer Integration Registry Settings**Synopsys Custom Designer Collections (CDOA)**

If Custom Designer object recognition is enabled during client installation, the CDOA registry key is set, to `dword:1`. Custom Designer object recognition can also be enabled via SyncAdmin's Third Party Integration pane. By default, Custom Designer object recognition is disabled.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to

DesignSync Data Manager Administrator's Guide

restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes  
\CDOA=dword:0
```

Allowed Values

Key Value	Description
CDOA=dword:0	Synopsys Custom Designer Collection object recognition is off. (Default)
CDOA=dword:1	Synopsys Custom Designer Collection object recognition is on.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Custom Designer Exclude List (Exclude)

This setting specifies an exclude list, specify comma-separated file types as the `Exclude` value. By default, the `Exclude` value is set to:

```
"*%,*-,*.cdslck*"
```

The Custom Designer object exclude list can also be enabled via SyncAdmin's Third Party Integration pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stlc` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes\CDOA\Exclude="*%,*-,*.cdslck*"
```

Allowed Values

Key Value	Description
<code>Exclude=""</code>	Nothing is excluded.
<code>Exclude="<value>[,<value>...]"</code>	General format showing how to exclude procedures.
<code>Exclude="*%,*-,*.cdslck*"</code>	Recommended list of values to exclude. (Default)

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

Custom Designer Exclude Proc (ExcludeProc)

This setting specifies a procedure that defines what objects to exclude. By default, the Exclude value is undefined, meaning no proc is selected.

The Custom Designer object exclude proc can also be enabled via SyncAdmin's Third Party Integration pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes\CDOA\ExcludeProc=""
```

Allowed Values

Key Value	Description
<code>ExcludeProc=""</code>	No procedures are excluded.(Default)
<code>ExcludeProc="<value>"</code>	General format showing how to exclude procedures. The value is the name of a <i>single</i> TCL procedure. It is used to define a procedure which will be called for each file in the "view" directory, with a single argument that is the path to the file. The

	function must return a value of 1 if the given file should NOT be included in the view collection object. Any other value will mean that the file is included in the collection.
--	--

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Third Party Integration Options

General Options

Microsoft Windows Visual Studio and SCC Integration Registry Keys**SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)**

This setting controls minimum checkin comment length as well as the default comment.

The AutoCheckInComment key is used for silent check-in operations initiated by the DesignSync SCC.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the

DesignSync Data Manager Administrator's Guide

client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_CURRENT_USER\Software\Synchronicity\General\ExtensionTypes\DSVS\AutoCheckInComment=autocheckin

Allowed Values

Key Value	Description
AutoCheckInComment=autocheckin	The default value is "autocheckin" which meets a comment length minimum of 10 characters, a typical minimal comment length.
AutoCheckInComment=<string value>	You may enter any string value comment you like to use as the default comment. If you use spaces in the comment, you must put quotes around the string. Note: You cannot specify a return character or a line feed character in the comment field.

Related Topics

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection size (Units)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

MathWorks Simulink Integration Registry Keys

Simulink Registry Keys Overview

Most of the operations performed into the DesignSync Simulink interface do not permit configurable options at command run time. To allow configurable options, DesignSync has provided a series of customizations in the form of DesignSync registry keys that control how the Simulink interface operates. Most of these keys control individual command operation, and one key determines whether DesignSync echoes commands to the Simulink output window;

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

DesignSync Data Manager Administrator's Guide

Show Status options in Simulink (Simulink\Show\Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Add object options in Simulink (Simulink\Add\File)

This setting controls the Simulink options for adding files to a module. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference*: add.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Add\File="-nodefaults -report brief"
```

Allowed Values

Key Value	Description
<code>File="<options>"</code>	Use the Command Reference to determine which add options to supply within the quoted string.
<code>File="-nodefaults -report brief"</code>	Adds the following options to the <code>ci</code> command. (Default) <ul style="list-style-type: none"><code>-nodefaults</code> - disables the command line-based command defaults system. This removes any unexpected settings.<code>-report brief</code> - sets the report mode to "brief." For more information on report options, see the <code>add</code> command.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

This setting controls the Simulink options for check a module into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: ci*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

DesignSync Data Manager Administrator's Guide

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes  
\Simulink\Checkin\ModuleInstance="-nodefaults -keep -report  
brief"
```

Allowed Values

Key Value	Description
Selected="<options>"	Use the Command Reference to determine which checkin options to supply within the quoted string.
Selected="- nodefaults -keep - report brief"	Adds the following options to the ci command. (Default) <ul style="list-style-type: none">• -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings.• -keep - specifies whether to keep a local copy of objects after the checkin operation .• -report brief - sets the report mode to "brief." For more information on report options, see the ci command.

Related Topics

[Simulink echoes DesignSync Commands to Output Window \(Simulink\OutputCmds\)](#)

[Add object options in Simulink \(Simulink\Add\File\)](#)

[Checkin Module options in Simulink \(Simulink\Checkin\ModuleInstance\)](#)

[Checkin objects options in Simulink \(Simulink\Checkin\Selected\)](#)

[Populate Lock options in Simulink \(Simulink\Lock\Selected\)](#)

[MkMod options in Simulink \(Simulink\Mkmod\Workspace\)](#)

[Populate Initial Workspace options in Simulink \(Simulink\Populate\InitialWs\)](#)

[Populate Module options in Simulink \(Simulink\Populate\ModuleInstance\)](#)

[Populate Revert options in Simulink \(Simulink\Populate\Revert & Simulink\Cancel\Selected\)](#)

[Populate Objects options in Simulink \(Simulink\Populate\Selected\)](#)

[Populate Revert option in Simulink \(Simulink\Populate\Version\)](#)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

This setting controls the Simulink options for checking in selected objects, not the entire module, into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference*: ci.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Checkin\Selected="-nodefaults -keep -report brief"
```

Allowed Values

Key Value	Description
Selected="<options>"	Use the Command Reference to determine which checkin options to supply within the quoted string.
Selected="-nodefaults -keep -report brief"	<p>Adds the following options to the ci command. (Default)</p> <ul style="list-style-type: none"> • <code>-nodefaults</code> - disables the command line-based command defaults system. This removes any unexpected settings. • <code>-keep</code> - specifies whether to keep a local copy of objects after the checkin operation . • <code>-report brief</code> - sets the report mode to "brief." For more information on report options, see the ci

	command.
--	----------

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Lock options in Simulink (Simulink\Lock\Selected)

This setting controls the Simulink options for populating locked versions selected objects, not the entire module, into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference*: populate.

Tip: The -nodefaults option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the

client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Lock\Selected="-nodefaults -lock -merge -replace -
report brief"
```

Allowed Values

Key Value	Description
Selected="<options>"	Use the Command Reference to determine which populate options to supply within the quoted string.
Selected="- nodefaults -lock - merge -replace - report brief"	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -lock - fetch locked copies. • -merge - merges the changes from the server with the changes made in the workspace. • -replace - replaces unmodified files and changes to the module structure. For more information, see the populate command. • -report brief - sets the report mode to "brief." For more information on report options, see the populate command. <p>Note: If a folder is selected, the operation defaults to running in folder (not module) recursive mode.</p>

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

DesignSync Data Manager Administrator's Guide

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

This setting controls the Simulink options for creating a new module (MkMod). The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: mkmod*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Mkmod\Workspace="-nodefaults -nocomment -report brief"
```

Allowed Values

Key Value	Description
-----------	-------------

Workspace="<options>"	Use the Command Reference to determine which MkMod options to supply within the quoted string.
Workspace="- nodefaults -nocomment -report brief"	<p>Adds the following options to the MkMod command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -nocomment - removes the requirement for a comment. • -report brief - sets the report mode to "brief." For more information on report options, see the mkmod command.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

This setting controls whether the Simulink integration echoes DesignSync commands to the Command output window. This setting is enabled by default.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes  
\Simulink\OutputCmds=dword:1
```

Allowed Values

Key Value	Description
<code>OutputCmds=dword:0</code>	Does not send DesignSync messages to the Simulink command output window.
<code>OutputCmds=dword:1</code>	Sends DesignSync messages to the Simulink command output window. (Default)

Related Topics

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

This setting controls the Simulink options for the initial (or first) populate of a module into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: populate*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stlc` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Populate\InitialWs="-nodefaults -get -report brief -
recursive -new"
```

Allowed Values

Key Value	Description
<code>InitialWs="<options>"</code>	Use the Command Reference to determine which populate options to supply within the quoted string.
<code>InitialWs="-nodefaults -get -report brief -recursive -new"</code>	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. -get - fetch unlocked copies.

	<ul style="list-style-type: none">• -report brief - sets the report mode to "brief." For more information on report options, see the populate command.• -recursive - populates the module hierarchy.• -new - get any new objects that have been exist on the server, but have never been populated into the workspace.
--	--

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

This setting controls the Simulink options for populating a module into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: populate*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Populate\ModuleInstance="-nodefaults -get -replace -
report brief -merge -recursive -new"
```

Allowed Values

Key Value	Description
ModuleInstance="<options>"	Use the Command Reference to determine which populate options to supply within the quoted string.
ModuleInstance="-nodefaults -get -replace -report brief -merge -recursive -new"	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> • <code>-nodefaults</code> - disables the command line-based command defaults system. This removes any unexpected settings. • <code>-get</code> - fetch unlocked copies. • <code>-replace</code> - replaces unmodified files and changes to the module structure. For more information, see the populate command. • <code>-report brief</code> - sets the report mode to "brief." For more information on report options, see the populate command. • <code>-merge</code> - merges the changes from the server with the changes made in the workspace. • <code>-recursive</code> - populates the module hierarchy. • <code>-new</code> - get any new objects that have been exist on the server, but have never been populated into the

	workspace.
--	------------

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

These settings controls the Simulink options for removing local changes, releasing locks and refreshing the module in the workspace with the server version. One of the following two keys is invoked by the "**Revert Local Changes and Release Lock**" command in Simulink. If there are no locked objects in the workspace, the Populate\Revert key is used to set the options. If there are locked files in the workspace, the Cancel\Selected key is used. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference*: ,mmmmmmmmmmmmmmmmmmmm and cancel.

Tip: The -nodefaults option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Keys

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Populate\Revert="-nodefaults -get -report brief -
force"
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Cancel\Selected="-nodefaults -keep -force"
```

Allowed Values

Key Value	Description
Revert=" <i><options></i> "	Use the Command Reference to determine which populate options to supply within the quoted string.
Revert="-nodefaults -get -report brief - force"	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -get - fetch unlocked copies. • -report brief - sets the report mode to "brief." For more information on report options, see the populate command.
Selected=" <i><options></i> "	Use the Command Reference to determine which cancel options to supply within the quoted string.
Selected="- nodefaults -keep - force"	<p>Adds the following options to the Cancel command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unforeseen settings. • -keep - specifies whether to keep a local copy of objects after canceling a lock operation . • -force - force overwrite of modified objects.

DesignSync Data Manager Administrator's Guide

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWVs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Objects options in Simulink (Simulink\Populate\Selected)

This setting controls the Simulink options for populating selected objects, not the entire module, into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: populate*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Populate\Selected="-nodefaults -get -replace -report
brief -merge -recursive -new"
```

Allowed Values

Key Value	Description
Selected="<options>"	Use the Command Reference to determine which populate options to supply within the quoted string.
Selected="- nodefaults -get - replace -report brief -merge - recursive -new"	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexecuted settings. • -get - fetch unlocked copies. • -replace - replaces unmodified files and changes to the module structure. For more information, see the populate command. • -report brief - sets the report mode to "brief." For more information on report options, see the populate command. • -merge - merges the changes from the server with the changes made in the workspace. • -recursive - populates the directory (folder) hierarchy. • -new - get any new objects that exist on the server, but have never been populated into the workspace.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

DesignSync Data Manager Administrator's Guide

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Populate Revert option in Simulink (Simulink\Populate\Version)

This setting controls the Simulink options for populating a specified version into the workspace. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: populate*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Populate\Version="-nodefaults -get -report brief -
replace"
```

Allowed Values

Key Value	Description
<code>Version="<options>"</code>	Use the Command Reference to determine which populate options to supply within the quoted string.

<pre>Version="- nodefaults -get - report brief - replace"</pre>	<p>Adds the following options to the Populate command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -get - fetch unlocked copies. • -report brief - sets the report mode to "brief." For more information on report options, see the populate command. • -replace - replaces unmodified files and changes to the module structure. For more information, see the populate command.
---	---

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Remove member options in Simulink (Simulink\Remove\Selected)

This setting controls the Simulink options for removing selected objects from the module. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: remove*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

Note: Remove cannot be run from the Simulink integration in `-noimmediate` mode. You must immediately commit the changes.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Remove\Selected="-nodefaults -keep -force -immediate -
recursive"
```

Allowed Values

Key Value	Description
Selected="<options>"	Use the Command Reference to determine which remove options to supply within the quoted string.
Selected="- nodefaults -keep - force -immediate - recursive"	<p>Adds the following options to the <code>ci</code> command. (Default)</p> <ul style="list-style-type: none"> • <code>-nodefaults</code> - disables the command line-based command defaults system. This removes any unexpected settings. • <code>-force</code> - removes the specified object, even if it has been locally modified. • <code>-keep</code> - leaves local copies of the removed objects and empty subfolders. These objects become unmanaged objects in the workspace. • <code>-immediate</code> - immediately commits the operation to the server. This is a required option.

- | | |
|--|---|
| | <ul style="list-style-type: none"> • -recursive - removes all objects in the folder and any subfolders. This option is ignored if the argument is not a module folder. |
|--|---|

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Show Status options in Simulink (Simulink\Show>Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Show Remove Comment (Simulink\Remove>ShowRmComment)

This setting controls the whether the user can specify a comment when removing a file from the project. By default, no comment is allowed during object removal.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stclc session, or if you are

DesignSync Data Manager Administrator's Guide

using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes\Simulink\Remove>ShowRmComment=dword:0
```

Allowed Values

Key Value	Description
ShowRmComment=dword:0	Does not allow a comment to be specified when the Remove command is used. (Default)
ShowRmComment=dword:1	Allows a comment to be specified when the Remove command is used. The comment is committed to the server when the object is removed from the module. If the object is added, then removed, and has never been committed to the module, the comment is added to the module during the next checkin.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show\Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Show Status options in Simulink (Simulink\Show\Status)

This setting controls the Simulink options for showing the module status. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: showstatus*.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Show\Status="-nodefaults -objects -hrefs -report
brief"
```

Allowed Values

Key Value	Description
Status=" <code><options></code> "	Use the Command Reference to determine which <code>showstatus</code> options to supply within the quoted string.
Status=" <code>nodefaults -objects -hrefs -report brief</code> "	<p>Adds the following options to the <code>showstatus</code> command. (Default)</p> <ul style="list-style-type: none"> <code>-nodefaults</code> - disables the command line-based command defaults system. This removes any unexpected settings. <code>-objects</code> checks the status of the workspace objects against the server versions to determine status. <code>-hrefs</code> follows the hierarchical references to determine if the reported status is current.

	<ul style="list-style-type: none">• -report brief - sets the report mode to "brief." For more information on report options, see the showstatus command.
--	--

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

This setting controls the Simulink options for tagging a module as part of the initial MkMod operation. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference: tag*.

Tip: The -nodefaults option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Tag\InitialBranch="-nodefaults -nocomment -mutable -
report brief"
```

Allowed Values

Key Value	Description
<code>InitialBranch="<options>"</code>	Use the Command Reference to determine which tag options to supply within the quoted string.
<code>InitialBranch="-nodefaults -nocomment -mutable -report brief"</code>	<p>Adds the following options to the tag command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -nocomment - removes the requirement for a comment. • -mutable - indicates that the module's tag is to be mutable, meaning it can be moved to a different module version. • -report brief - sets the report mode to "brief." For more information on report options, see the tag command.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show>Status)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

Tag options in Simulink (Simulink\Tag\ModuleVersion)

This setting controls the Simulink options for tagging a module version. The Registry Key section shows the default values. The options are described in the *ENOVIA Synchronicity Command Reference*: tag.

Tip: The `-nodefaults` option disables the Command Default system for the command. This is highly recommended for any clients launched from a GUI interface.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\ExtensionTypes
\Simulink\Tag\ModuleVersion="-nodefaults -mutable -replace -
report brief"
```

Allowed Values

Key Value	Description
-----------	-------------

InitialBranch="<options>"	Use the Command Reference to determine which Tag options to supply within the quoted string.
<pre>InitialBranch="- nodefaults -nocomment - mutable -report brief"</pre>	<p>Adds the following options to the tag command. (Default)</p> <ul style="list-style-type: none"> • -nodefaults - disables the command line-based command defaults system. This removes any unexpected settings. • -mutable - indicates that the module's tag is to be mutable, meaning it can be moved to a different module version. • -replace - indicates that it should move the tag, if the tag name is already in use on the module. • -report brief - sets the report mode to "brief." For more information on report options, see the tag command.

Related Topics

Simulink echoes DesignSync Commands to Output Window (Simulink\OutputCmds)

Add object options in Simulink (Simulink\Add\File)

Checkin Module options in Simulink (Simulink\Checkin\ModuleInstance)

Checkin objects options in Simulink (Simulink\Checkin\Selected)

Populate Lock options in Simulink (Simulink\Lock\Selected)

MkMod options in Simulink (Simulink\Mkmod\Workspace)

Populate Initial Workspace options in Simulink (Simulink\Populate\InitialWs)

Populate Module options in Simulink (Simulink\Populate\ModuleInstance)

Populate Revert options in Simulink (Simulink\Populate\Revert & Simulink\Cancel\Selected)

Populate Objects options in Simulink (Simulink\Populate\Selected)

Populate Revert option in Simulink (Simulink\Populate\Version)

Remove member options in Simulink (Simulink\Remove\Selected)

Show Status options in Simulink (Simulink\Show\Status)

Initial Tag options in Simulink (Simulink\Tag\InitialBranch)

Collection members (MapMemberOps)

The MapMemberOps key is set when SyncAdmin's General Options pane is used, to "Map operations on collection members to owner".

By default, attempting to apply a revision control operation to a collection member warns that the member file is not a versionable object. If you set MapMemberOps to dword:1, then when a member file is specified to operate on, the revision control commands ci, co, tag, cancel, unlock and retire will determine what collection object the member file belongs to, and operate on that collection object.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\MapMemberOps=dword:0
```

Allowed Values

Key Value	Description
MapMemberOps=dword:0	Does not treat individual collection members as versionable objects. (Default)
MapMemberOps=dword:1	Allows you to apply revision control operations to a selected member which then performs on the containing collection object.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)
 fisrvDoesOp optimization (FisrvDoesOp)
 Cadence non-collection members (Cadence View Folder)
 Non-view folders in cell directories (Cadence NonView Folder)
 Cadence client compatibility mode setting (ManageViewManifest)
 Collection size (Units)
 Synopsys Custom Designer Collections (CDOA)
 Custom Designer Exclude List (Exclude)
 Custom Designer Exclude Proc (ExcludeProc)
 Third Party Integration Options
 General Options

Collection size (Units)

The `Units` key is set when SyncAdmin's General Options pane is used, for whether to "Show collection size in bytes".

By default, the size shown for a collection is its number of member files. To instead show the collection object size as the sum of the size of its member files, set `Units="Bytes"` .

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Collections\Units="Files"
```

Allowed Values

Key Value	Description
Units="Files"	Collection size is shown as the number of member files.(Default)
Units="Bytes" .	Collection size is shown as the sum of the size of its member files in bytes.

Related Topics

SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment)

Cadence Auto-Checkin Default Comment (Cadence\AutoCheckInComment)

Cadence Design Systems' Collections (CadenceView)

Cadence View Depth (MaxDepth)

fisrvDoesOp optimization (FisrvDoesOp)

Cadence non-collection members (Cadence View Folder)

Non-view folders in cell directories (Cadence NonView Folder)

Cadence client compatibility mode setting (ManageViewManifest)

Collection members (MapMemberOps)

Synopsys Custom Designer Collections (CDOA)

Custom Designer Exclude List (Exclude)

Custom Designer Exclude Proc (ExcludeProc)

Third Party Integration Options

General Options

DesignSync GUI Registry Settings

Automatically refresh (AutoRefreshTimer)

The `AutoRefreshTimer` key is used to determine whether to automatically refresh the GUI display. It can also be set when SyncAdmin's GUI Options pane is used, to "Automatically refresh after how many minutes". For best performance, this option is disabled by default.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\
AutoRefreshTimer=dword:0
```

Allowed Values

Key Value	Description
AutoRefreshTimer=dword:0	Never shows the "stale" state for an object in GUI. (Default)
AutoRefreshTimer=dword:a	Sets the GUI to refresh automatically after 10 minutes. This is a suggested value.
AutoRefreshTimer=dword:<value>	Sets the GUI to refresh automatically after the specified number of minutes has passed. Specify the number of minutes in hexadecimal.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

DesignSync Data Manager Administrator's Guide

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Web browser (Browser)

The `Browser` key sets the web browser used when an action within DesignSync launches a web browser. This can include using the `webhelp` command, launching help from the GUI, or viewing data sheets. The default value is set when the DesignSync client is configured. The `Browser` default value can be changed using SyncAdmin's General Pane.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the

client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\Browse  
r="<value>"
```

Allowed values

The command to launch the web browser. If the command is not on your path, you must include the full path. There is no validation performed on this key.

Related Topics

File editor (FileEditor)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Columns (Columns)

The `Column` keys are used to determine which columns display in the List View Pane. It is also set when SyncAdmin's Columns pane is used, to select which columns appear in the list view.

By default, all of the columns represented by the registry keys are visible. That default behavior is represented by the registry values of `dword:1`. To hide any of the above columns, set the registry value for the column to: `dword:ffffffe`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\Columns\Branch=dword:1
```

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\Columns\Cache URL=dword:1
```

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Description=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Locker=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Member of=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Modified=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Project URL=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Reference Target=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Relative Path=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Result=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Selector=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Server URL=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Size=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Status=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Type=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Version=dword:1

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\
Columns\Version Tags=dword:1

Allowed Values

Key Value	Description
\Columns\ <i><specified_column></i> =dword:1	Allows the specified column to be visible. (Default)
\Columns\ <i><specified_column></i> =dword:ffffffe	Hides specified column.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargelcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Confirm use of "populate -force" (ConfirmPopForce)

The `ConfirmPopForce` key is used to determine if you need to confirm a populate action when the `-force` option is specified. This key can also be set by the SyncAdmin's GUI Options pane, "Confirm use of Populate 'Force overwrite of local modifications' option".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notifications\ConfirmPopForce=dword:1
```

Allowed Values

Key Value	Description
<code>ConfirmPopForce=dword:0</code>	Does not launch a confirmation screen when populate operations are run with the <code>-force</code> switch.
<code>ConfirmPopForce=dword:1</code>	Launches a confirmation screen when populate operations are run with the <code>-force</code> switch. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

DesignSync Data Manager Administrator's Guide

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Show data sheet in frame (DataSheetInFrame)

The `DataSheetInFrame` key determines whether the data sheet is displayed in a View Pane of the GUI window or in the default web browser. This key can also be set when SyncAdmin's GUI Options pane is used, to "Show data sheet in frame".

By default, data sheets are shown in the View Pane. This is represented by a `DataSheetInFrame` value of `dword:1`. To instead display data sheets in a browser, set a `DataSheetInFrame` value of `dword:0`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\
DataSheetInFrame=dword:1
```

Allowed Values

Key Value	Description
<code>DataSheetInFrame=dword:0</code>	Launches the default web browser to display the data sheet.
<code>DataSheetInFrame=dword:1</code>	Displays the data sheet in a tab within the View Panel of the DesignSync GUI window. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

DesignSync Data Manager Administrator's Guide

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

File editor (FileEditor)

The `FileEditor` key sets the file editor used by the DesignSync GUI when **Edit** is clicked for a selected DesignSync object. The default value is set when the DesignSync client is configured. The `FileEditor` default value can also be changed using SyncAdmin's General Pane.

Notes:

- When SyncAdmin is invoked via the DesignSync GUI's **Tools=>Options**, an additional GUI Options pane appears, for the Command Bar. Its settings are highly dependent on individual users' display preferences, so are not listed here.
- Most of SyncAdmin's GUI Customization panes (Main Toolbar, Module Toolbar, Keyboard Shortcuts, Custom Tools, Commands, States, Command Options, and Custom Columns) write out multiple registry settings .

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\FileEditor="<value>"
```

Allowed values

The command to launch the editor. If the command is not on your path, you must include the full path. There is no validation performed on this key.

Related Topics

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

DesignSync Data Manager Administrator's Guide

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Font size (FontSize)

The `FontSize` key controls the font size, which can range from 10-18 point. The key is set when SyncAdmin's Display Options pane is used, to select a "Font size". The default font size is 12 pt. which is represented by "C" in hexadecimal.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the

client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stlc session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

`HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\FontSize=dword:<n>`

Allowed Values

Key Value	Description
<code>ConfirmPopForce=dword:<n></code>	Enter the font size in hexadecimal. Allowed font sizes range from 10 pt (<code>dword:a</code>) to 18 pt (<code>dword:12</code>).
<code>ConfirmPopForce=dword:c</code>	The default font size of 12pt in hexadecimal. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Icon size (TreeViewLargelcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

DesignSync Data Manager Administrator's Guide

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Age of history (Expire)

The `Expire` key is used to determine the how many days until the locations in the history expire. The key is also set when SyncAdmin's History pane is used, to specify the number of "Days until locations expire".

The `Expire` value is represented in hexadecimal.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\History\Expire=dword:<n>
```

Allowed Values

Key Value	Description
Expire=dword:<n>	Expiration time for the entries in the history in hexadecimal.
Expire=dword:1e	Expiration time for the entries in the history is set to 30 days. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Initial folder (InitialDir, InitialDirSpecify)

The InitialFolder registry keys controls whether an initial directory is selected when the GUI starts and which directory. The keys are also set when SyncAdmin's GUI Options Initial Folder pane is used, to specify whether a folder is initially selected in the tree view on startup.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\  
InitialDir=dword:<n>
```

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync/HLD\General\  
InitialDirSpecify="<value>"
```

Allowed Values

Key Value	Description
InitialDir=dword:0	No initial directory is set. (Default)
InitialDir=dword:1	Uses the current directory as the initial directory for the GUI.
InitialDir=dword:2	Uses the home directory as the initial directory for

	the GUI.
<code>InitialDir=dword:3</code>	Use the directory specified by the <code>InitialDirSpecify</code> key.
<code>InitialDirSpecify="<value>"</code>	Specify the full path of the directory the GUI uses as its initial starting directory. This key is only used when the <code>InitialDir</code> key value is 3.

Related Topics

File editor (`FileEditor`)

Web browser (`Browser`)

Synchronize GUI and CLI (`SyncGUI`)

Always refresh on tree selection (`RefreshOnTreeSelect`)

Show stale icon (`ObsoleteTimer`)

Automatically refresh (`AutoRefreshTimer`)

Show data sheet in frame (`DataSheetInFrame`)

Confirm use of "populate -force" (`ConfirmPopForce`)

Font size (`FontSize`)

Icon size (`TreeViewLargeIcons`)

Time format (`TimeFormat`)

Look & feel and Metal-style window frames (`LookAndFeel`, `Theme`)

Age of history (`Expire`)

Amount of history (`RecentMax`)

Columns (`Columns`)

Tag values (`TagList`)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Look & feel and Metal-style window frames (LookAndFeel, Theme)

The `LookAndFeel`, and `Theme` registry keys are used to determine the look of the DesignSync GUI interface. These options can also be set when SyncAdmin's Display Options pane is used, to select the "Look & feel", and whether to "Use Metal-style window frames".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\LookAndFeel=<value>
```

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\Theme=<value>
```

Allowed Values

Key Value	Description
<code>LookAndFeel="javax.swing.plaf.metal.MetalLookAndFeel"</code> <code>Theme=""</code>	Sets the look and feel to the Metal (Java) skin.
<code>LookAndFeel="javax.swing.plaf.metal.MetalLookAndFeel"</code>	Sets the look

Theme="com.synchronicity.lib.ui.MetalThemes\$SyncTheme"	and feel to the Metal (Synchronicity) skin.
LookAndFeel="com.sun.java.swing.plaf.motif.MotifLookAndFeel" Theme=""	Sets the look and feel to the CDE/Motif skin.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargelcons)

Time format (TimeFormat)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Show stale icon (ObsoleteTimer)

The `ObsoleteTimer` key determines when the GUI reports an object as stale. This key is also set when SyncAdmin's GUI Options pane is used, to "Show stale icon after how many minutes". By default, this option is disabled.

This option is disabled by default, represented by an `ObsoleteTimer` value of `dword:0`. To have the GUI indicate that information for an object may need to be refreshed, set an `ObsoleteTimer=dword:<value>`, where `<value>` is the number of minutes, represented in hexadecimal. For example, for a value of 10 minutes, specify `ObsoleteTimer=dword:a`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\
ObsoleteTimer=dword:0
```

Allowed Values

Key Value	Description
<code>ObsoleteTimer=dword:0</code>	Never shows the "stale" state for an object in GUI. (Default)

ObsoleteTimer=dword:a	Sets the object to stale, meaning that it needs to be refreshed, after 10 minutes. This is a suggested value.
ObsoleteTimer=dword:<value>	Sets the object to stale, meaning that it needs to be refreshed, after the specified number of minutes has passed. Specify the number of minutes in hexadecimal.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Amount of history (RecentMax)

The `RecentMax` key is used to determine how many recently visited locations are shown in the Go window. This key is also set when SyncAdmin's History pane is used, to specify the "Number of recent history items shown in Go menu".

The `RecentMax` value is represented in hexadecimal.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\History\RecentMax=dword:<n>
```

Allowed Values

Key Value	Description
<code>RecentMax=dword:<n></code>	Enter the number of last visited locations desired in hexadecimal.
<code>RecentMax=dword:a</code>	Lists the last 10 visited locations. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Always refresh on tree selection (RefreshOnTreeSelect)

The `RefreshOnTreeSelect` key determines whether the GUI refreshes information when returning to a folder in an open client. This key is also set when SyncAdmin's GUI Options pane is used, to "Always refresh on tree selection". By default, and for best performance, once a folder has been visited, the GUI does not refresh its displayed information when next visiting that folder.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\RefreshOnTreeSelect=dword:0
```

Allowed Values

Key Value	Description
<code>RefreshOnTreeSelect=dword:0</code>	Does not refresh the GUI when a folder is revisited within the open client. (Default)
<code>RefreshOnTreeSelect=dword:1</code>	Automatically refreshes the GUI with current information every time a folder is visited .

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Synchronize GUI and CLI (SyncGUI)

The `SyncGUI` key controls whether selection and current directory are synchronized between the tree and list views and the command bar. The value is also set when SyncAdmin's GUI Options pane is used, to "Synchronize graphical and command-line interface".

The default, to synchronize the GUI's tree and list views and the command bar, is represented by a `SyncGUI` value of `dword:1`. To not synchronize the GUI and command bar, set a `SyncGUI` value of `dword:0`.

DesignSync Data Manager Administrator's Guide

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\General\SyncGUI=dword:1
```

Allowed Values

Key Value	Description
syncGUI=dword:0	Does not maintain synchronization between the GUI and the command bar.
syncGUI=dword:1	Maintains synchronization between the GUI and the command bar. (Default)

Related Topics

File editor (FileEditor)

Web browser (Browser)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargelcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Tag values (TagList)

The `TagList` key is used to define a list of tag name choices presented by the Tag dialog. This key is also set when SyncAdmin's Tags pane is used, to specify tag name choices to be listed in DesignSync's Tag dialog.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

DesignSync Data Manager Administrator's Guide

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Options\TagList  
=<tag values>
```

Allowed Values

Key Value	Description
TagList="<tag>[<tag>...]"	Specify the tags to show as a space-separated list of tag names.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargeIcons)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Time format (TimeFormat)

The `TimeFormat` key is used to set the time format. The key can also be set when SyncAdmin's Display Options pane is used, to select a "Time format".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\TimeFormat="<value>"
```

Allowed Values

Key Value	Description
<code>TimeFormat="07/13/2003 15:37:04"</code>	Displays time format as MM, DD, YYYY followed by 24 hour time. (Default)
<code>TimeFormat="07/13/2003 03:37:04 PM"</code>	Displays time format as MM, DD, YYYY followed by 12 hour time and period (AM/PM).
<code>TimeFormat="13/07/2003 15:37:04"</code>	Displays time format as DD, MM, YYYY followed by 24 hour time.
<code>TimeFormat="13/07/2003 03:37:04 PM"</code>	Displays time format as DD, MM, YYYY followed by 12 hour time and period (AM/PM).
<code>TimeFormat="Jul 13,</code>	Displays time format as a three letter abbreviation for

DesignSync Data Manager Administrator's Guide

2003 15:37:04"	month, DD, YYYY followed by 24 hour time.
TimeFormat="Jul 13, 2003 3:37:04 PM"	Displays time format as a three letter abbreviation for month, DD, YYYY followed by 12 hour time and period.
TimeFormat="13, Jul 2003 15:37:04"	Displays time format as DD, three letter abbreviation for month, YYYY followed by 24 hour time.
TimeFormat="13, Jul 2003 3:37:04 PM"	Displays time format as DD, three letter abbreviation for month, YYYY followed by 12 hour time and period.
TimeFormat="2003-07-13 15:37:04"	Displays time format as YYYY-MM-DD followed by 24 hour time.

Related Topics

File editor (FileEditor)

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Icon size (TreeViewLargelcons)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

Icon size (TreeViewLargeIcons)

The `TreeViewLargeIcons` key determines whether to display the icons as large or small. The key can also be set when SyncAdmin's Display Options pane is used, to "Display large icons".

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\TreeViewLargeIcons=dword:0
```

Allowed Values

Key Value	Description
<code>TreeViewLargeIcons=dword:0</code>	Displays small icons in the GUI. (Default)
<code>TreeViewLargeIcons=dword:1</code>	Displays large icons in the GUI.

Related Topics

File editor (FileEditor)

DesignSync Data Manager Administrator's Guide

Web browser (Browser)

Synchronize GUI and CLI (SyncGUI)

Always refresh on tree selection (RefreshOnTreeSelect)

Show stale icon (ObsoleteTimer)

Automatically refresh (AutoRefreshTimer)

Show data sheet in frame (DataSheetInFrame)

Confirm use of "populate -force" (ConfirmPopForce)

Initial folder (InitialDir, InitialDirSpecify)

Font size (FontSize)

Time format (TimeFormat)

Look & feel and Metal-style window frames (LookAndFeel, Theme)

Age of history (Expire)

Amount of history (RecentMax)

Columns (Columns)

Tag values (TagList)

General Options

GUI Options

Initial Folder Options

Display Options

History Option

Column Options

Command Bar Options

Tag Options

DesignSync diff Display Registry Settings

Diff Tool Ancestor (Ancestor)

When performing a three way diff, the diff tool needs to identify the ancestor. The built-in diff tool uses the `-a` option to identify the ancestor. If you are not using the built-in diff tool, your tool may require a different option to identify the ancestor. Consult the documentation for the diff tool to determine if 3-way diff is supported, and if so, what flag is used to identify which is the ancestor version so you can set the registry key to that flag.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\DiffTool\Flags\Ancestor="-a"
```

Allowed Values

Key value	Description
<code>Ancestor=""</code>	The empty value indicates that the diff utility does not support 3 day diff, and in this case the ancestor value will not be passed into the diff utility invoked.
<code>Ancestor="(arg) "</code>	Indicates that no flag is sent to identify the ancestor argument. This should be used when the diff utility supports 3 day diff and uses the position of the argument to identify the common ancestor
<code>Ancestor="-a"</code>	Sends <code>-a <ancestor></code> to the GUI diff utility invocation to identify the ancestor for a 3-way diff. (Default)
<code>Ancestor="<value>"</code>	The specified value is used as a switch to be passed into the diff utility along with the value of the common ancestor. This should be used when the diff utility supports a 3 way diff, and uses a switch (such as <code>'-a'</code>) to identify that the value that follows the switch is the common ancestor.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Binary Compare (Binary)

The `Binary` value should be set to the value used by the Diff graphical utility to allow binary compare. If there is no value, do not set this key.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stlc` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
Software\Synchronicity\General\Commands\General\DiffTool\Flags\Binary=<value>
```

Allowed Values

Key value	Description
<code>Binary=""</code>	Indicates no value.
<code>Binary=<value></code>	Set the "value" to the expected binary flag for your diff

utility.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Diff Format Options

Diff Options (BlackBg)

Those settings are highly dependent on individual users' display preferences and individual registry keys are not documented because it is understood that users will generally configure their own their Diff tool look-and-feel using SyncAdmin. The one exception is the `BlackBg` registry key is set when SyncAdmin's Diff Options pane is used, to "Use black background". This may be something an administrator would choose to set more globally.

The complete list of Diff options is available on the SyncAdmin Diff Options pane .

Those settings are highly dependent on individual users' display preferences, so are not listed here.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_CURRENT_USER\Software\Synchronicity\DesignSync\HLD\Client\DiffView\BlackBg=dword:0

Allowed Values

Key value	Description
BlackBg=dword:0	Sets the background of the diff window to white. (Default)
BlackBg=dword:1	Sets the background of the diff window to black.

Related Topics

Diff Format (Diff2Format)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Diff Tool (Command)

By default, DesignSync uses its built-in graphical Diff tool for graphical diff and conflict resolution. If there is a preferred Diff tool in use at your site, you can use this registry key to specify the location of the desired graphical Diff utility.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the

syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\DiffTool\Command="<value>"
```

Example

This example shows a sample registry entry for the TkDiff utility on Windows. It defines both the location of the tkdiff executable and the library locations needed to support it.

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\DiffTool\Command="set TK_LIBRARY=c:\Program Files\Synchronicity\DesignSyncV6R2010\share\tk\library && "c:\Program Files\Synchronicity\DesignSyncV6R2010\bin\wish.exe" "c:\Program Files\tkdiff-4.1.4-unix\tkdiff.tcl""
```

Allowed Values

Any path to a valid diff tool; must also include any supporting files and their locations. If your path contains spaces it must be enclosed in quotations.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Diff Format (Diff2Format)

DesignSync Data Manager Administrator's Guide

The `Diff2Format` setting controls whether which diff tool to launch when launching a 2 or 3-way diff tool. The `Diff2Format` registry key is set when DesignSync is installed. The `Diff2Format` registry key is set when SyncAdmin's General Diff Format pane is used, to select the "Format to use for both 2-way and 3-way diffs". The default value is "Use Graphical Diff Tool", represented by a `DiffFormat` value of `dword:5`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\Diff2Format=dword:<n>
```

Allowed Values

Key value	Description
<code>Diff2Format=dword:0</code>	Revised diff format (2-way diff only)
<code>Diff2Format=dword:1</code>	Standard diff format
<code>Diff2Format=dword:2</code>	Unified diff format
<code>Diff2Format=dword:3</code>	Classic Syncdiff format
<code>Diff2Format=dword:4</code>	Annotated Diff format
<code>Diff2Format=dword:5</code>	Use graphical tool (Default)

Related Topics

[Diff Options \(BlackBg\)](#)

[Diff Tool \(Command\)](#)

[Diff Tool Ancestor \(Ancestor\)](#)

[Embedded Whitespace Characters \(Embedded\)](#)

[Leading/Trailing Whitespace Characters \(IgnoreWhite\)](#)

[Ignore Case \(IgnoreCase\)](#)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Embedded Whitespace Characters (Embedded)

The `Embedded` value should be set to the value used by the Diff graphical utility to indicate that whitespace within lines should be ignored in file comparisons. If there is no value, enter a null string for the value (`""`).

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNCH_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNCH_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNCH_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\DiffTool\Flags\Embedded="<value>"
```

Allowed Values

Key value	Description
<code>Embedded=""</code>	Indicates no value.
<code>Embedded="<value>"</code>	Set the "value" to the expected embedded whitespace ignore flag for your diff utility.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Ignore Case (IgnoreCase)

The `IgnoreCase` value should be set to the value used by the Diff graphical utility to indicate that file comparisons should ignore case. If there is no value, enter a null string for the value ("").

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
Software\Synchronicity\General\Commands\General\DiffTool\Flags\IgnoreCase="<value>"
```

Allowed Values

Key value	Description
<code>IgnoreCase=""</code>	Indicates no value.
<code>IgnoreCase="<value>"</code>	Set the "value" to the expected case ignore flag for your diff utility.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

Ignore RSC Keys (IgnoreKeys)

The `IgnoreKeys` value should be set to ignore the \$keys used by RSC system when performing file comparisons in the Diff graphical utility. If there is no value, enter a null string for the value (`""`).

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's

`$$SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
Software\Synchronicity\General\Commands\General\DiffTool\Flags\IgnoreKeys="<value>"
```

Allowed Values

Key value	Description
<code>IgnoreKeys=""</code>	Indicates no value.
<code>IgnoreKeys="<value>"</code>	Set the "value" to ignore the RSC keys in your diff utility.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Leading/Trailing Whitespace Characters (IgnoreWhite)

Ignore Case (IgnoreCase)

Binary Compare

Diff Format Options

Leading/Trailing Whitespace Characters (IgnoreWhite)

The `IgnoreWhite` value should be set to the value used by the Diff graphical utility to indicate that leading or trailing whitespace characters should be ignored in file comparisons. If there is no value, enter a null string for the value ("").

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. Settings specific to an individual user may be made to that user's `$_SYNC_USER_CFGDIR/UserRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\DiffTool\Flags\IgnoreWhite="<value>"
```

Allowed Values

Key value	Description
<code>IgnoreWhite=""</code>	Indicates no value.
<code>IgnoreWhite="<i><value></i>"</code>	Set the "value" to the expected leading/trailing whitespace ignore flag for your diff utility.

Related Topics

Diff Format (Diff2Format)

Diff Options (BlackBg)

Diff Tool (Command)

Diff Tool Ancestor (Ancestor)

Embedded Whitespace Characters (Embedded)

Ignore Case (IgnoreCase)

Ignore RSC Keys (IgnoreKeys)

Binary Compare

Diff Format Options

DesignSync Client Optimizations Registry Settings

Available drive types (AllowableDriveTypes)

This setting allows you to control the type of drives that DesignSync navigates on the Windows platform. This setting is not controlled by SyncAdmin and must be created manually.

The default hexadecimal dword value of 1F indicates that the client should include all drives, including mounted network drives. The dword value is created by using bitwise OR to combine the possible options. The allowed hexadecimal values are:

- 01 - fixed media only; for example, a hard drive or flash drive.
- 02 - remote (network) drives only.
- 04 - removable media only; for example, floppy drives, USB flash drives.
- 08 - CD-ROM drives only.
- 10 - RAM storage only.

These bits can also be combined to set the desired value. For example, the value `dword:1D` excludes network drives.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

DesignSync Data Manager Administrator's Guide

HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimizations\AllowableDriveTypes=dword:1F

Allowed Values

Key Value	Description
AllowableDriveTypes=dword:1F	Navigate all drive types (Default)
AllowableDriveTypes=dword:1D	Navigate all drives except network drives.
AllowableDriveTypes=dword:01	Navigate only fixed media.
AllowableDriveTypes=dword:02	Navigate only network drives.
AllowableDriveTypes=dword:04	Navigate only removable media.
AllowableDriveTypes=dword:08	Navigate only CD-ROM drives.
AllowableDriveTypes=dword:10	Navigate only RAM storage.

Related Topics

Keyword Expansion (EnableKeywordExpansion)

Multi-threading (MaxWorkerThreads)

Cache reference counting (CacheDisableRefCount)

Direct fetch (EnableDirectFetch)

Pre-checkin disk space check (Precheckin)

Link-in of files to the server (EnableLinkIn)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (DefaultFetchType)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Cache reference counting (CacheDisableRefCount)

The `CacheDisableRefCount` registry key is set when SyncAdmin's General Performance pane is used, to "Enable cache optimizations".

By default, DesignSync keeps track of how many users have links to cached files, automatically removing file versions from the cache that are no longer referenced by anyone. Disabling cache reference counting is a performance optimization. When this cache optimization is enabled, caches should periodically be cleaned manually. To enable this cache optimization, set a `CacheDisableRefCount` value of `dword:1`

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\CacheDisableRefCount=dword:0
```

Allowed Values

Key Value	Description
<code>CacheDisableRefCount=dword:0</code>	Allows automatic cleaning of un-referenced cache files. (Default)
<code>CacheDisableRefCount=dword:1</code>	Disables cache reference counting and removal of un-referenced cache files.

Related Topics

Keyword Expansion (`EnableKeywordExpansion`)

Multi-threading (`MaxWorkerThreads`)

Direct fetch (`EnableDirectFetch`)

Pre-checkin disk space check (`Precheckin`)

Link-in of files to the server (`EnableLinkIn`)

Available drive types (`AllowableDriveTypes`)

DesignSync Data Manager Administrator's Guide

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (DefaultFetchType)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Direct fetch (EnableDirectFetch)

The `EnableDirectFetch` registry key is set when SyncAdmin's General Performance pane is used, to "Enable direct fetch".

By default, DesignSync creates a temporary file in the workspace metadata, when fetching a file into the workspace. To skip that step, set a `EnableDirectFetch` value of `dword:1`. Trade-offs when setting `EnableDirectFetch` are discussed in SyncAdmin's Performance Options.

Note: `EnableDirectFetch` is automatically disabled for Populate with merge operations.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimiz  
ations\EnableDirectFetch=dword:0
```

Allowed Values

Key Value	Description
<code>EnableDirectFetch=dword:0</code>	Creates temporary workspace file while fetching a file. (Default)

<code>EnableDirectFetch=dword:1</code> Skips temporary file creation.

Related Topics

Keyword Expansion (`EnableKeywordExpansion`)

Multi-threading (`MaxWorkerThreads`)

Cache reference counting (`CacheDisableRefCount`)

Pre-checkin disk space check (`Precheckin`)

Link-in of files to the server (`EnableLinkIn`)

Available drive types (`AllowableDriveTypes`)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (`DefaultFetchType`)

Populate Full/Incremental (`PopulateUseIncremental`)

From Location Default (`FetchFromLocal`)

Keyword Expansion (`EnableKeywordExpansion`)

The `EnableKeywordExpansion` registry key is set when SyncAdmin's General Performance pane is used, to "Enable keyword expansion".

By default, DesignSync expands keywords in ascii files, except when certain vault types are used. Disabling keyword expansion is a performance optimization, because keyword expansion requires that data be transferred from the vault. To disable keyword expansion, set a `EnableKeywordExpansion` value of `dword:0`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes

DesignSync Data Manager Administrator's Guide

made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the `syncdadmin` command to restart syncd). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimizations\EnableKeywordExpansion=dword:1

Allowed Values

Key Value	Description
EnableKeywordExpansion=dword: 0	Value "0" disables keyword expansion.
EnableKeywordExpansion=dword: 1	Value "1" enables keyword expansion. (Default)

Related Topics

Multi-threading (MaxWorkerThreads)

Cache reference counting (CacheDisableRefCount)

Direct fetch (EnableDirectFetch)

Pre-checkin disk space check (Precheckin)

Link-in of files to the server (EnableLinkIn)

Available drive types (AllowableDriveTypes)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (DefaultFetchType)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Link-in of files to the server (EnableLinkIn)

The `EnableLinkIn` registry key is set when SyncAdmin's General Performance pane is used, to "Enable Link-in if possible".

By default, when possible, DesignSync will create hard links to large files in the server's vault from a user's check-in. This performance optimization only applies to Unix clients. The "pre-checkin disk space check" described above must be enabled, for the Link-In optimization to be attempted. For details on the Link-In optimization, see Linking Large Files.

To disable the Link-In optimization, set an `EnableLinkIn` value of `dword:0`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimizations\EnableLinkIn=dword:1
```

Allowed Values

Key Value	Description
<code>EnableLinkIn=dword:0</code>	Disables hard linking to server vault from check-in.
<code>EnableLinkIn=dword:1</code>	Allows creation of hard links to files in server vault from check-in. (Default)

Related Topics

Keyword Expansion (`EnableKeywordExpansion`)

Multi-threading (`MaxWorkerThreads`)

Cache reference counting (`CacheDisableRefCount`)

Direct fetch (`EnableDirectFetch`)

DesignSync Data Manager Administrator's Guide

Pre-checkin disk space check (Precheckin)

Available drive types (AllowableDriveTypes)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (DefaultFetchType)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Multi-threading (MaxWorkerThreads)

The `MaxWorkerThreads` registry key is set when SyncAdmin's General Performance pane is used, to "Use client threads".

By default, DesignSync uses five client worker threads, to operate on five objects in parallel. The `MaxWorkerThreads` value can range from 1 thread (`dword:1` hexadecimal) to 10 threads (`dword:a` hexadecimal). To disable multi-threading, so that DesignSync uses only its main thread, set a `MaxWorkerThreads` value of `dword:0`.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\General\Optimizations\MaxWorkerThreads=dword:5
```

Allowed Values

Key Value	Description
<code>MaxWorkerThreads=dword:0</code>	Disables client threading.

MaxWorkerThreads=dword:<n>	Set the desired number of client threads up to a maximum of 10.
MaxWorkerThreads=dword:5	Sets number of client threads to 5. (Default)
MaxWorkerThreads=dword:10	Sets number of client threads to 10. This is the maximum value allowed.

Related Topics

Keyword Expansion (EnableKeywordExpansion)

Cache reference counting (CacheDisableRefCount)

Direct fetch (EnableDirectFetch)

Pre-checkin disk space check (Precheckin)

Link-in of files to the server (EnableLinkIn)

Available drive types (AllowableDriveTypes)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (DefaultFetchType)

Populate Full/Incremental (PopulateUseIncremental)

From Location Default (FetchFromLocal)

Pre-Checkin disk space check (EnablePreCheckin/PreCheckinThreshold)

The registry keys below are set when SyncAdmin's General Performance pane is used, to "Check disk space before every checkin".

By default, DesignSync checks whether there is sufficient disk space on the server, before attempting to checkin a file that is 256 Megabytes or larger. The performance optimization is to fail immediately if there isn't enough space. Otherwise, the file would

be sent to the server with the checkin request, which would then fail, but only after running out of disk space.

To modify the file size used to determine whether a pre-checkin disk space check is performed, set `PreCheckinThreshold` to `dword:<value>`, where the value is the number of Megabytes, represented in hexadecimal. The default value of 256 Megabytes is represented by a hexadecimal value of `dword:100`. To disable the pre-checkin disk space check, set an `EnablePrecheckin` value of `dword:0`.

These registry settings pertain to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Settings

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimiz
ations\EnablePreCheckin=dword:1
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimiz
ations\PreCheckinThreshold=dword:100
```

Allowed Values

Key Value	Description
<code>EnablePreCheckin=dword:0</code>	Disables pre-checkin disk space check.
<code>EnablePreCheckin=dword:1</code>	Allows pre-checkin disk space check. (Default)
<code>PreCheckinThreshold=dword:100</code>	Representative value of file size threshold for pre-checkin disk space check. (Default)
<code>PreCheckinThreshold=dword:<value></code>	Sets representative value of file size threshold for pre-checkin disk space check to the specified hexadecimal value.

Related Topics

Keyword Expansion (`EnableKeywordExpansion`)

Multi-threading (`MaxWorkerThreads`)

Cache reference counting (`CacheDisableRefCount`)

Direct fetch (`EnableDirectFetch`)

Link-in of files to the server (`EnableLinkIn`)

Available drive types (`AllowableDriveTypes`)

Performance Options

About Vault Types

Cleaning a Cache

Linking Large Files

Default Fetch State (`DefaultFetchType`)

Populate Full/Incremental (`PopulateUseIncremental`)

From Location Default (`FetchFromLocal`)

Validate Reference Workspace (`ValidateReferenceWorkspace`)

The `ValidateReferenceWorkspace` registry key provides a pre-check for locked, swapped and locally modified objects in workspaces that are being duplicated using the `duplicatews` command.

By default, DesignSync verifies that the workspace does not contain locked, swapped, or modified files before executing the `duplicatews` command. (`dword:1`). To skip that validation, set a `ValidateReferenceWorkspace` value of `dword:0`.

Note: The registry setting is used as the default. If the `-validate/-novalidate` option is specified on the command, the command option takes precedent over this default.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimizations\ValidateReferenceWorkspace=dword:1

Allowed Values

Key Value	Description
ValidateReferenceWorkspace=dword:0	Does not perform any pre-check validation on a workspace being duplicated. If there are locked or modified files in the workspace, the operation fails during execution and an appropriate error message is generated to explain the issue.
ValidateReferenceWorkspace=dword:1	Performs a pre-check on a workspace being duplicated to verify that no files within the workspace are locked or modified before sending the command to the server. (Default)

Workspace Metadata Registry Settings

Attempts to acquire metadata lock (ClientMetadataLockNumTries)

The default `ClientMetadataLockNumTries` value is the maximum integer available on the system, represented in hexadecimal. The minimum value that DesignSync allows is 10 attempts (`dword:a`). Any value lower than 10 (`dword:a`) is interpreted as 10 (`dword:a`).

Specify the `ClientMetadataLockNumTries` value in hexadecimal.

Note: The values of `ClientMetadataLockNumTries` and `ClientMetadataLockTimeout` work in combination to influence the wait time to acquire the metadata lock. If you change the default value for either, make sure their combined new values allow enough time for operations to acquire the metadata lock. For example, suppose you change `ClientMetadataLockNumTries` to 10 (tries) and `ClientMetadataLockTimeout` to 100 (milliseconds, or 0.1 second). Their combined values allow a wait time of 1 second total (10*100). If a user has a lock on the metadata for an operation that lasts 5 seconds, there is not enough time for the operation to complete and the next operation to acquire the metadata lock. So the next operation fails because it could not acquire the lock.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

HKEY_CURRENT_USER\Software\Synchronicity\General\Locks\ClientMetadataLockNumTries=dword:a

Allowed Values

Key value	Description
ClientMetadataLockNumTries=dword:"1"... "a"	Sets the number of tries to 10. All values below 10 round to 10. (Minimum Value)
ClientMetadataLockNumTries=dword:<value>	Sets the number of tries to the specified value.
ClientMetadataLockNumTries=dword:<max_value>	Sets the number of tries to the maximum available hexadecimal value on the system. (Default/Maximum Value)

Related topics

Amount of time to wait for metadata lock (ClientMetadataLockTimeout)

Orphaned lock timeout (OrphanTimeout)

Troubleshooting Metadata Errors

Amount of time to wait for metadata lock (ClientMetadataLockTimeout)

The default `ClientMetadataLockTimeout` value is 1000 milliseconds (1 second), represented in hexadecimal as `dword:3E8`. The minimum that DesignSync allows is 100 milliseconds (0.1 seconds), represented in hexadecimal as `dword:64`. Any value lower than 100 (`dword:64`) is interpreted as 100 (`dword:64`).

Specify the `ClientMetadataLockTimeout` value in milliseconds, and as a hexadecimal number.

Note: The values of `ClientMetadataLockNumTries` and `ClientMetadataLockTimeout` work in combination to influence the wait time to acquire the metadata lock. If you change the default value for either, make sure their combined new values allow enough time for operations to acquire the metadata lock. For example, suppose you change `ClientMetadataLockNumTries` to 10 (tries) and `ClientMetadataLockTimeout` to 100 (milliseconds, or 0.1 second). Their combined values allow a wait time of 1 second total (10*100). If a user has a lock on the metadata for an operation that lasts 5 seconds, there is not enough time for the operation to complete and the next operation to acquire the metadata lock. So the next operation fails because it could not acquire the lock.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Locks\  
ClientMetadataLockTimeout=dword:3E8
```

Allowed Values

Key value	Description
<code>ClientMetadataLockTimeout=dword:64</code>	Sets the lock timeout to 100 milliseconds. (Minimum Value)
<code>ClientMetadataLockTimeout=dword:3E8</code>	Sets the lock timeout to 1000 milliseconds. (Default)
<code>ClientMetadataLockTimeout=dword:<value></code>	Sets the lock timeout to the specified value.

Related topics

Attempts to acquire metadata lock (`ClientMetadataLockNumTries`)

Orphaned lock timeout (`OrphanTimeout`)

Troubleshooting Metadata Errors

Orphaned lock timeout (OrphanTimeout)

This setting sets the wait time before removing a leftover lock. Leftover locks can be generated when a DesignSync client session was not cleanly exited, there may be a leftover metadata lock. By default, DesignSync waits 300 seconds (5 minutes), before presuming the leftover metadata lock to be orphaned, and removing that leftover lock. This default behavior is represented by an `OrphanTimeout` value of `dword:12C`.

Specify the `OrphanTimeout` value in seconds, and as a hexadecimal number.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to either the client installation's `$_SYNCH_CUSTOM_DIR/site/config/SiteRegistry.reg` file or a `$_SYNCH_PROJECT_CFGDIR/ProjectRegistry.reg` file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your `DesSync`, `dssc`, or `stcl` session, or if you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`). An alternative to restarting the DesignSync client is to use the `sregistry reset` command to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\DotLock\Orphan
Timeout=dword:12c
```

Allowed Values

Key value	Description
<code>OrphanTimeout=dword:12c</code>	Sets presumption of orphaned metadata lock and subsequent removal to 5 minutes after an improper exit. (Default)
<code>OrphanTimeout=dword:<value></code>	Sets presumption of orphaned metadata lock and subsequent removal to the specified amount of minutes after an improper exit.

Related topics

Attempts to acquire metadata lock (ClientMetadataLockNumTries)

Amount of time to wait for metadata lock (ClientMetadataLockTimeout)

Troubleshooting Metadata Errors

Development Areas Registry Settings

Enabling Shared Development Areas (ShowSharedDevelopmentAreaOption)

When creating development areas, you can create shared development areas, which can be used by other users. By default, all users can create shared development areas.

If you do not want users to be able to create shared development areas, you can disable this option, which removes it from the Make Area tab in the DesignSync Development Area Management tool, and from the list of available options for the sda_mk command.

This registry setting pertains to DesignSync clients. To modify registry values programmatically, use the sregistry command set. The settings must be made to either the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file or a \$SYNC_PROJECT_CFGDIR/ProjectRegistry.reg file. In order for changes made to the client registry files to take effect, you must restart your DesignSync client (exit and restart your DesSync, dssc, or stcl session, or if you are using dss or stcl, use the syncdadmin command to restart syncd). An alternative to restarting the DesignSync client is to use the sregistry reset command to re-read the registry files. the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\sda\ShowSharedDevelopmentAreaOption=dword:1
```

Allowed Values

Key value	Description
ShowSharedDevelopmentAreaOption=dword:0	Remove the option to create shared developments from the make area commands.
ShowSharedDevelopmentAreaOption=dword:1	Allow make area actions to create the development as a shared development area. (Default)

Related Topics

Enterprise Servers

Registry Settings for SyncServers

Modules Registry Settings

Do Not Create Filtered Links (AllowFilteredLinks)

The `AllowFilteredLinks` key is used to prevent creation of mcache links during filtered populates.

When this key is enabled (1), DesignSync creates mcache links, when appropriate, when the populate command is executed with either the `-filter` or `-view` options.

When this key is disabled (0), populate using either the `-filter` or `-view` options does not create mcache links.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\Mcache\AllowFilteredLinks=dword:1
```

Allowed Values

Key Value	Descriptions
<code>AllowFilteredLinks=dword:0</code>	Disallows creation of mcache links during <code>-filter</code> or <code>-view</code> populate command executions.
<code>AllowFilteredLinks=dword:1</code>	Allows creation of mcache links during <code>-filter</code> or <code>-view</code> populate command executions. (Default)

Related Topics

Allow auto-merging (`AutoMerge`)

Automatically update Access Control file during module update (`AddAccessControls`)

Module Instance Name Format (`FindModuleByName`)

Modules Options

Allow auto-merging (`AutoMerge`)

The `AutoMerge` registry key is set when SyncAdmin's General Modules pane is used, for whether to "Allow auto-merging when creating a new module version on the server".

By default, auto-merging is allowed. To disallow auto-merging, set `AutoMerge` to `dword:0`.

This registry setting pertains to a `SyncServer`. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the `SyncServer` is to use the `sregistry` `reset` command in a server-side script to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\AutoMerge=dword:1
```

Allowed Values

Key Value	Description
<code>AutoMerge=dword:0</code>	Disallows auto-merging when creating new module versions on server.
<code>AutoMerge=dword:1</code>	Allows auto-merging when creating new module versions on server. (Default)

Related Topics

Do Not Create Filtered Links (`AllowFilteredLinks`)

Automatically update Access Control file during module update (`AddAccessControls`)

Module Instance Name Format (`FindModuleByName`)

Modules Options

Automatically update Access Control file during module update (`AddAccessControls`)

The `AddAccessControls` key is used to enable automatic modifications to the `AccessControl` files that occur when an object is upgraded (using `upgrade`) to the current module structure.

When this key is disabled, you should manually update your access controls to restrict edits to the objects being upgraded and the new module during module creation.

By default, the AccessControl files are updated automatically. To disable the auto update, set `AddAccessControl` to `dword:0`.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Key

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Modules\Upgrade\AddAccessControls-dword:1
```

Allowed Values

Key Value	Description
<code>AddAccessControls=dword:0</code>	Disables auto-update during module update.
<code>AddAccessControls=dword:1</code>	Allows auto-update during module update. (Default)

Related Topics

Do Not Create Filtered Links (`AllowFilteredLinks`)

Allow auto-merging (`AutoMerge`)

Module Instance Name Format (`FindModuleByName`)

Modules Options

Module Instance Name Format (`FindModuleByName`)

The `FindModuleByName` key is used to determine whether a module in a workspace must be identified by the name and instance number (`<Mod>%<Instance#>`) or may be identified by either the name or the name and instance number.

When this key is enabled (1), you may use either the name or instance reference to refer to a workspace module.

When this key is disabled (0), you must use the instance reference to refer to a module. If you specify a name, the name is assumed to be a workspace path. This is the default behavior.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Key

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\FindModuleByName=dword:0
```

Allowed Values

Key Value	Description
FindModuleByName=dword:0	Disables key, must use instance reference to refer to all modules. All searches are assumed to be workspace paths. (Default)
FindModuleByName=dword:1	Enables key, allows search by either name or instance reference.

Related Topics

Do Not Create Filtered Links (AllowFilteredLinks)

Allow auto-merging (AutoMerge)

Automatically update Access Control file during module update (AddAccessControls)

Modules Options

Data Storage and Maintenance Registry Settings

Backup Registry Settings

Earliest backup time (BackupEarliestTime)

Important: This value must be set in order for backups to run.

The time before which no backup will run. Set this time to the value of the first backup time. This allows you fine control over when the backups begin.

The default value is: "23:00"

Tip: To reduce load on the SyncServer, you should set this value to a specific time when the load tends to be light. For example, if your lightest hours are between 2 and 5 am, you can set this value to "02:00" and your first backup will begin then. Using the `BackupFrequencyKey`, you can then repeat the backup at the same time every day.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupEarliestTime="HH:MM"
```

Related Topics

Vault type used for data (`VaultType`)

Alarm trigger frequency (`MaintenanceTimeout`)

PostgreSQL maintenance (`PGMaintenanceTime`)

Apache (`httpsd`) process size (`ProcessSizeLimit`)

NFS Disk Quota Check Timeout (`NFSTimeout`)

Disk Quota Checking (`DiskLimits`)

Server disk space required (`FreeSpaceAbsErr`)

Server disk space required for the Postgres database (`FreeSpaceRelErr`)

Size of the transaction log (`TransactionLogRetainRecords`)

When database checkpointing occurs (`DBCcheckpointTime`)

Backup Frequency (`BackupFrequency`)

Number of incremental backups (`BackupNumberOfIncrementals`)

Length of time to retain backups (`BackupRetentionTime`)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Backup Frequency (BackupFrequency)

The number of hours between automatically launching a new backup run. If the value is set to 0 (zero), no backups run. The default value is 18 (24 hours) meaning backups run once a day.

Tip: To launch backups at a specific time each day, use the BackupEarliestTime key to set the first time for the backup, and set the BackupFrequency key to 18.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupFrequency=dword:18
```

Allowed Values

Key Value	Description
BackupFrequency=dword:0	Disallows automatic backup.
BackupFrequency=dword:18	Sets backup interval to 24 hours. (Default)
BackupFrequency=dword:<value>	Accepts any specified hexadecimal value for backup interval.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Schedule for Full Backups (BackupFullBackupDay)

This setting allows you to pre-schedule which days of the week are designated for running the full backup. DesignSync recommends running a full backup once a week and daily incremental backups. This methodology of explicitly specifying the data of the full backup replaces the Number of incremental backups (BackupNumberOfIncrementals) methodology of setting the number of incremental backups to run between full backups.

Specify the day of the week as a numeric value in a comma separated list.

Day of the week	Number
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6
Sunday	7

Tip: For an optimally efficient backup system, select a single number which is the desired day.

If this key is set, but a full backup has not successfully completed for three weeks, DesignSync will attempt to run a full backup until a full backup successfully completes, and then will resume the schedule defined by the key.

Note:The Backup Frequency (BackupFrequency) key must be set to 24 hours, or a multiple of 24 hours. (48, 72 etc.)

Important: The system maintains the information about when the backups were run in two files in the backup area, Backup.sync. The files are .LastBackup and .LastIncremental. Do not move, delete, or modify these files. They are required for the automatic backups to operate properly.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupFullBackupDay="7"
```

Key Value	Description
BackupFullBackupDay="7"	Runs the full backup once a week, on Sunday.
BackupFullBackupDay="1, 6"	Runs the full backup twice a week, on

	Monday, and on Saturday.
BackupFullBackupDay=<value>	Sets specified days of the week to run a full backup. All other backup runs will be incremental backups.

Related Topics

Procedure for Defining Automatic Backups

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Length of time to retain backup logs (BackupLogRetentionTime)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

The calculated length of time, in hours, that backup log files should be maintained on disk. The default value is represented in hexadecimal as `dword:79` (121 hours). This value is based on the optimal scheduling of nightly incremental and weekly full backups, using this formula:

$$\text{BackupFrequency} * 5 + 1$$

When the periodic backup runs, any backup log files on the disk older than this value are deleted. The default value causes the system to retain the last 5 backup logs. If there are no automatic backups scheduled, no cleanups are ever performed.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

DesignSync Data Manager Administrator's Guide

HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupLogRetentionTime=dword:79

Allowed Values

Key value	Description
BackupLogRetentionTime=dword:79	Sets backup log retention time to 121 hours. (Default)
BackupLogRetentionTime=dword:<value>	Sets backup log retention time to specified value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Number of incremental backups (BackupNumberOfIncrementals)

The number of incremental backups between each full backup. If you set the value to 0 (zero), every backup is a full backup. The default value is 6. All values are in hexadecimal.

Tip: For an optimally efficient backup system, set to 6 and the value of BackupFrequency to 18, meaning that you run a full backup once a week and an incremental backup daily.

Alternatively, you can define the specific days of the week on which to run full backup. For more information, see Schedule for Full Backups (BackupFullBackupDay).

Important: The system maintains the information about when the backups were run in two files in the backup area, Backup.sync. The files are .LastBackup and .LastIncremental. Do not move, delete, or modify these files. They are required for the automatic backups to operate properly.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupNumberOfIncrementals=dword:6
```

Key Value	Description
BackupNumberOfIncrementals=dword:0	Sets all backups performed to be full backups.
BackupNumberOfIncrementals=dword:6	Sets 6 incremental backups to run between each full backup. (Default)
BackupNumberOfIncrementals=dword:<value>	Sets specified value as number of incremental backups between

	each full backup.
--	-------------------

Related Topics

Procedure for Defining Automatic Backups

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Schedule for Full Backups (BackupFullBackupDay)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Length of time to retain backups (BackupRetentionTime)

The calculated length of time, in hours, that backup files should be maintained on disk. The default value is represented in hexadecimal as `dword:a9` (169 hours). This value is based on the optimal scheduling of nightly incremental and weekly full backups, using this formula:

$$(\text{BackupNumberOfIncrementals}+1) * \text{BackupFrequency}+1$$

When the periodic backup runs, any backups on the disk older than this value are deleted. The default value causes the system to effectively keep all backups back to, and including, the last scheduled full backup. If there are no automatic backups scheduled, no cleanups are ever performed.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\BackupRetentionTime=dword:A9
```

Allowed Values

Key value	Description
BackupRetentionTime=dword:A9	Sets backup retention to 169 hours. (Default)
BackupRetentionTime=dword:<value>	Sets backup retention to specified value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Data Storage Registry Settings

When database checkpointing occurs (DBCheckpointTime)

This setting controls when database checkpointing occurs. By Default, daily checkpointing of the server's databases occurs at 1:00am, (DBCheckpointTime value of "01:00"). When changing the DBCheckpointTime value, use the server's local time in ISO 8601 format (24 hour time, seconds omitted) as shown in these examples:

DBCheckpointTime="never": Turn off automatic checkpointing using this special value.

DBCheckpointTime="Sat 03:00": Perform weekly checkpointing on Saturdays at 3:00 AM.

DBCheckpointTime="00:00": Perform daily checkpointing at midnight.

Before setting the DBCheckpointTime value, you can use Tcl to verify the value for the time. For example, if you want to schedule checkpointing for Saturdays at 3:00 AM, verify the time using these Tcl commands:

```
stcl> set t {sat 0300}
sat 0300
stcl> clock format [clock scan $t]
Sat Jul 15 03:00:00 EDT 2000
```

Note:The alarm trigger that drives the checkpointing process only happens every 15 minutes. Therefore, it is possible that if you set the checkpointing to run at 1:00AM, the checkpointing may not occur until 1:14AM.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\DBCheckpointTime="<hh>:<mm>"
```

Allowed Values

Key Value	Description
DBCheckpointTime="never"	Disallows checkpointing of server databases.
DBCheckpointTime=<day> <hh>:<mm>	Allows checkpointing of server databases at the specified day and time.
DBCheckpointTime="01:00"	
DBCheckpointTime=<hh>:<mm>	Allows checkpointing of server databases at the specified time.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Disk Quota Checking (DiskLimits)

When enabled, DesignSync provides advanced warning when reaching the soft quota limit, allowing you to adjust the quota or data storage location appropriately.

The default value disabled, (dword:0) . To enable disk quota checking, set dword:1 .

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\DiskLimits\CheckQuota=dword:0
```

Allowed Values

Key Value	Description
DiskLimits\CheckQuota=dword:0	Disallows disk quota checking. (Default)
DiskLimits\CheckQuota=dword:1	Allows disk quota checking.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Server disk space required (FreeSpaceAbsErr)

The `FreeSpaceAbsErr=dword:<value>` is the number of megabytes to reserve as the required minimum space, represented in hexadecimal. The server periodically checks for available disk space in the server vault file system, the database directory, and the tmp file system. When the disk space conditions are not satisfied, an error message is generated. This prevents database journal corruption due to out-of-space conditions.

Note: The registry keys controlling the minimum amount of database space are in Server disk space required for the Postgres database.

For example, the server generates the following message:

```
sync-E-34: Out of disk space. Only 0 megabytes are free on the
file system containing /tmp/. A minimum of 32 megabytes is
required.
```

The default value is `dword:20` (32 Mb). If a value of `dword:0` is specified, then the default of `dword:20` (32 Mb) is used.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either

DesignSync Data Manager Administrator's Guide

the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\DiskLimits\FreeSpaceAbsErr=dword:20
```

Allowed Values

Key value	Description
<code>FreeSpaceAbsErr=dword:0</code>	Reverts to default.
<code>FreeSpaceAbsErr=dword:20</code>	Sets minimum required disk space for the server to 32Mb. (Default)
<code>FreeSpaceAbsErr=dword:<value></code>	Sets minimum required disk space for the server to specified hexadecimal value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Server disk space required for the Postgres database (FreeSpaceRelErr/FreeSpaceAbsErr)

These settings work together to define the minimum percentage or minimum absolute disk space required to maintain Postgres database integrity. The server periodically checks for available disk space in the database directory. When the disk space conditions are not satisfied, an error message is generated. This prevents database journal corruption due to out-of-space conditions.

The `FreeSpaceAbsErr=dword:<value>` is the number of megabytes to reserve as the required minimum space, represented in hexadecimal. The default value is `dword:20` (32 Mb). If a value of `dword:0` is specified, then the default of `dword:20` (32 Mb) is used.

The `FreeSpaceRelErr=dword:<value>` is the minimum percentage of the system to reserve as the required minimum space, represented in hexadecimal. This value is not set by default. By default, the system uses `FreeSpaceAbsErr` to determine the minimum free space required in MB.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Settings

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\DiskLimits\PgData\FreeSpaceAbsErr=dword:20
```


DesignSync Data Manager Administrator's Guide

HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\DiskLimits\PgData\FreeSpaceRelErr=dword:<value>

Allowed Values

Key value	Description
FreeSpaceAbsErr=dword:0	Reverts to default.
FreeSpaceAbsErr=dword:20	Sets minimum required disk space for the Postgres database to 32Mb. (Default)
FreeSpaceAbsErr=dword:<value>	Sets minimum required disk space for the Postgres database to specified hexadecimal value.
FreeSpaceRelErr=dword:<value>	Sets minimum percentage of disk space for the Postgres database to specified hexadecimal value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

NFS Disk Quota Check Timeout (NFSTimeout)

The server periodically checks for available disk space on the NFS drives. This key sets the timeout value for queries. If the quota query reply does not come back within the timeout limit, DesignSync will cancel the query and check the quota later.

The default value is 10 seconds (dword:A).

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\DiskLimits\NFSTimeout=dword:<value>
```

Allowed Values

Key value	Description
<code>NFSTimeout=dword:a</code>	Sets disk quota check time-out to 10 seconds. (Default)
<code>NFSTimeout=dword:<value></code>	Sets disk quota check time-out to specified hexadecimal value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

DesignSync Data Manager Administrator's Guide

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Aliases Keyword Expansion (EnableAliasesKeyword)

This setting controls keyword expansion in ascii files. The `EnableAliasesKeyword` registry key is set when SyncAdmin's Site options pane is used, to "**Honor 'Aliases' Keyword Expansion**".

By default, DesignSync expands the Aliases keyword in ascii files, except when certain vault types are used. Disabling Aliases keyword expansion from the server is a performance optimization, because keyword expansion requires that data be traversed in the server vault to determine all applicable tags. To disable Aliases keyword expansion, set a `EnableAliasesKeyword` value of `dword:0`.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\Enable
AliasesKeyword=dword:1
```

Allowed Values

Key value	Description
EnableAliasesKeyword=dword:0	Disallows keyword expansion in ascii files.
EnableAliasesKeyword=dword:1	Allows keyword expansion in ascii files. (Default)

Alarm trigger frequency (MaintenanceTimeout)

The `MaintenanceTimeout` registry key is set when SyncAdmin's Site Options pane is used, to control the "Alarm trigger maintenance time-out".

The `<value>` is the number of seconds, in hexadecimal, defaulting to 384 (15 minutes). The `MaintenanceTimeout` cannot be disabled

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\Server\MaintenanceTimeo
ut=dword:384
```

Allowed Values

Key value	Description
MaintenanceTimeout=dword:384	Sets alarm trigger maintenance time-out to 15 minutes. (Default)
MaintenanceTimeout=dword:<value>	Sets alarm trigger maintenance time-out to specified hexadecimal value.

Related Topics

Site Options

ProjectSync User's Guide: Setting Up Email Reminders for Defects.

Enterprise Design Synchronization Queue

PostgreSQL maintenance (PGMaintenanceTime)

The `PGMaintenanceTime` registry key is set when SyncAdmin's Site Options pane is used, to "Perform PostgreSQL maintenance".

Daily maintenance is enabled by default, with a `PGMaintenanceTime` value of "2:00". To disable daily PostgreSQL maintenance, set a value of "never".

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\PGMaintenanceTime="02:00"
```

Key Value	Description
<code>PGMaintenanceTime="never"</code>	Disables daily PostgreSQL maintenance.
<code>PGMaintenanceTime="02:00"</code>	Sets maintenance time to 2:00AM. (Default)
<code>PGMaintenanceTime="<hh>:<mm>"</code>	Set maintenance time to specified time value.

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

Apache (httpd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Apache (httpd) process size (ProcessSizeLimit)

The `ProcessSizeLimit` registry key is set when SyncAdmin's Site Options pane is used, to "Limit server process size".

The `<value>` is the process size in Megabytes, represented in hexadecimal, defaulting to `3c` (60 Mb). To not limit the server's process size, set a `ProcessSizeLimit` value of `dword:0`

This registry setting pertains to a `SyncServer`. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the `SyncServer` is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\ProcessSizeLimit=dword:3c
```

Allowed Values

Key Value	Description
<code>ProcessSizeLimit=dword:0</code>	Sets server process size limit to unlimited.
<code>ProcessSizeLimit=dword:3c</code>	
<code>ProcessSizeLimit=dword:<value></code>	Sets server process size to described hexadecimal value.

Related Topics

Vault type used for data (`VaultType`)

Alarm trigger frequency (`MaintenanceTimeout`)

PostgreSQL maintenance (`PGMaintenanceTime`)

NFS Disk Quota Check Timeout (`NFSTimeout`)

Disk Quota Checking (`DiskLimits`)

Server disk space required (`FreeSpaceAbsErr`)

Server disk space required for the Postgres database (`FreeSpaceRelErr`)

Size of the transaction log (`TransactionLogRetainRecords`)

When database checkpointing occurs (`DBCcheckpointTime`)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Size of the transaction log (TransactionLogRetainRecords)

The transaction log file is described in Reducing the Size of the Transaction Log. By default, records are retained for 168 hours (seven days), represented in hexadecimal as `dword:A8`.

Reduce this setting to limit the number of hours. For example, to limit the period to three days (72 hours), you would specify `48` (the hexadecimal representation of 72) as the `dword` value.

Do not set the period to less than 48 hours (30 hexadecimal).

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\TransactionLogRetainRecords=dword:<value>
```


Allowed Values

Key value	Description
TransactionLogRetainRecords=dword:30	Sets transaction record retention period to 48 hours. (Minimum Value)
TransactionLogRetainRecords=dword:<value>	Sets transaction record retention period to specified value.
TransactionLogRetainRecords=dword:A8	Sets transaction record retention period to 168 hours (7 days). (Maximum Value)

Related Topics

Vault type used for data (VaultType)

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Temporary Directory for Upload

The UploadTmpDir registry setting allows you to specify a directory on the server which can be used by the upload command to expand the compressed archive, perform delta comparisons and upload the changes between the uploaded version and the current server version. DesignSync recommends that the temporary directory be able to hold roughly 2.5 times the size of the uploaded archive. Specifying a temporary directory provides the ability to designate a private area which has enough space for the operation. It might be preferable to use a specified directory rather than rely on the shared system /tmp directory. This setting can be overridden by the user when they upload the archive. This setting can also be set using the DesignSync Administrator tool, Site Options. For more information on using Upload, see Upload Archive.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\UploadTmpDir=/tmp
```

Allowed Values

Key value	Description
/tmp	This the standard value on most unix machines. (Default)
<path>	The directory path on the server. This path should be written in unix format with "/" as a path delimiter.

Vault type used for data (VaultType)

DesignSync Data Manager Administrator's Guide

The registry keys below are set when SyncAdmin's Site Options Vault Types pane is used. Or when DesignSync Web UI's Administer Server Settings Vault Types tab is used.

Each vault type mapping must have the five lines shown below, with a "UniqueKey" used to group the five lines for a vault type setting. SyncAdmin and DesignSync Web UI use "SiteSubType1", "SiteSubType2", "SiteSubType3", etc. for their unique key groupings.

The "vault_type" can be any of these three values:

Sync:som:RceVault (Default)

Sync:som:CopyVault

Sync:som:SmartVault

For more information on Vault Types, see About Vault Types

The "pattern_match" value determines which files are stored using the specified "vault_type". For example:

* (Default)

*.tgz

*.Z

The "init" value can be:

An empty string "" for uncompressed (not zipped) vault types, including SmartVault when there are no user defined parameters

"zipped" for Zipped CopyVault, or Zipped SmartVault when there are no user defined parameters

A comma separated value list with the user defined parameters for SmartVault, beginning with whether the SmartVault is zipped

For example:

Init=",262144,50,524288,0" (unzipped, default parameter values)

Init="zipped,262144,50,524288,0" (zipped, default parameter values)

SmartVault's parameters are described in the topic [Converting Vault Data](#). The "Rank" <value> is the priority of the vault type mapping. If more than one vault type mapping definition matches the object being newly checked in, then the vault type mapping with the lowest "Rank" will be used. The Rank <value> can range from 100000 to 900000.

The default vault type mappings are shown in the SyncAdmin Help topic [Add or Edit Vault Association](#).

The registry settings below pertain to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Settings

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\WOT\File\<UniqueKey>\@=none
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\WOT\File\<UniqueKey>\Match="pattern_match"
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\WOT\File\<UniqueKey>\VaultType="vault_type"
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\WOT\File\<UniqueKey>\Init="string"
```

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\WOT\File\<UniqueKey>\Rank=dword:<value>
```

Allowed Values

Key Value	Description
<UniqueKey>\@=none	Describes value of the <Unique_Key>
<UniqueKey>\Match="pattern_match"	Allows described file type to be stored by referencing the "vault_type".
<UniqueKey>\VaultType=Sync:som:RceVault	Specifies RCE type vault. (Default)
<UniqueKey>\VaultType=Sync:som:CopyVault	Specifies Copy type vault.

DesignSync Data Manager Administrator's Guide

<code><UniqueKey>\VaultType=Sync:som:SmartVault</code>	Specifies Smart type vault.
<code><UniqueKey>\Init="string"</code>	Defines SmartVault parameters, any valid parameter values are allowed.
<code><UniqueKey>\Rank=dword:<value></code>	Determines priority order of vault type mapping, any rank number 100000-900000 in hexadecimal is allowed.

Related Topics

Alarm trigger frequency (MaintenanceTimeout)

PostgreSQL maintenance (PGMaintenanceTime)

Apache (httpsd) process size (ProcessSizeLimit)

NFS Disk Quota Check Timeout (NFSTimeout)

Disk Quota Checking (DiskLimits)

Server disk space required (FreeSpaceAbsErr)

Server disk space required for the Postgres database (FreeSpaceRelErr)

Size of the transaction log (TransactionLogRetainRecords)

When database checkpointing occurs (DBCcheckpointTime)

Earliest backup time (BackupEarliestTime)

Backup Frequency (BackupFrequency)

Number of incremental backups (BackupNumberOfIncrementals)

Length of time to retain backups (BackupRetentionTime)

Length of time to retain backup logs (BackupLogRetentionTime)

Converting Vault Data

Reducing the Size of the Transaction Log

Changing Vault Types

Add or Edit Vault Association

Site Options

Allow Non-Secure Cache Populate (AllowNonSecureCachePopulate)

When enabled, DesignSync prevents data from being populated into a cache by older clients (pre-V6R2016x) to ensure the protection of non-cacheable objects. This option can also be set by the Allow older clients to access non-cacheable Objects on the Site panel of SyncAdmin.

The default value disabled, (dword:0) . To enable this feature, set dword:1 .

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Server\Options\AllowNonSecureCachePopulate=dword:0
```

Allowed Values

Key value	Description
AllowNonSecureCachePopulate=dword:0	Allows older clients to populate objects from the server into a cache.
AllowNonSecureCachePopulate=dword:1	Disables populating into a cache for all objects on the server from older clients.

Related Topics

[Understanding Excluding IP from the Cache](#)

[Adding or Removing an Object from Caching](#)

Enterprise Design Settings

Enterprise Access Control Delegation (DelegateAC)

The Enterprise Design system allows you to delegate access control for platform connected DesignSync objects to the Enterprise Platform. When Delete Access Checks for Enterprise Managed Models is enabled, modules which are Platform connected

have all the access checked delegated to the connected Platform. This option can be set from the Enterprise Servers tab of SyncAdmin.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Enterprise\DelegateAC=dword:0
```

Allowed Values

Key value	Description
DelegateAC=dword:0	Access Control checks are handled locally, by the DesignSync client. (Default)
DelegateAC=dword:1	Access Control checks are handled by the ENOVIA Platform server.

Related Topics

Enterprise Servers

Enterprise User Identity Mapping (enterpriseUserMapType)

The Enterprise Design system involves reconciling the user names on the Enterprise Application server with the user names on the DesignSync server. On the DesignSync side, this is accomplished partially by the enterpriseUserMapType registry key.

The value of the key indicates what type of mapping is used between the two servers. The default value, UN-UN, indicates that there is a one-to-one match between the DesignSync user name and the Enterprise Application user name.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted for the changes to take effect. An

alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\EDA\enterprise
UserMapType="UN-UN"
```

Allowed Values

Key value	Description
enterpriseUserMapType="UN-UN"	DesignSync username maps directly to the Enterprise Server user name. (Default)
enterpriseUserMapType="UN-EM"	DesignSync "username" is the user's email address which maps to a username on the Enterprise Server using the email associated with the Enterprise Server user. If there is a team of users associated with the development, that list is used to restrict the search for matching users. If there no DesignSync email address for the user, and the development has a set of team members associated with it, the DesignSync username is used and compared against the Enterprise Server email address list. If there is no list of team members associated with a development, and there is no email address associated with the DesignSync user, the mapping defaults to UN-UN.
enterpriseUserMapType="EM-UN"	Enterprise Application server "username" is an email address which maps to a username on the DesignSync server using the email associated with the DesignSync user profile. If there is a team of users associated with the development, that list is used to restrict the search for matching users. If there is no DesignSync email for the user and the development has a set of team member associated with it, the DesignSync server username is used

	<p>for the match. If the development does not have an associated team, and there is no matching email, the mapping defaults to UN-UN.</p>
<p><code>enterpriseUserMapType="EM-EM"</code></p>	<p>Email addresses on both the DesignSync and Enterprise Application servers must match.. If there is a team of users associated with the development, that list is used to restrict the search for matching users.</p>
<p><code>enterpriseUserMapType=<functionName></code></p>	<p>Name of the custom TCL function that performs the mapping. The function must always be available to the server, for example, autoloaded. The function must take two arguments:</p> <ul style="list-style-type: none"> • TCL list of pairs username/email address on the DesignSync server. If there is no email address associated with the user, DesignSync automatically uses the username for both elements • TCL list of pairs username/email address on the Enterprise Application server. This list contains the identities from the assignment members. If a development team has no members, this list is empty. <p>The function must return a list of pairs equivalent to the first argument but with the values mapped as desired.</p>

Related Topics

Setting Up Enterprise Design User Mappings

Store User Lists (NoAssignmentUserList)

The Enterprise Design system involves reconciling the user names on the Enterprise Application server with the user names on the DesignSync server. This user list can be stored on the DesignSync server and used to filter out the developments and assignments available to the the user using the SDA GUI. Because the user list can grow quite large, processing the list can slow down the performance of the SDA GUI.

DesignSync allows you to not store user lists to avoid impacting the performance of the SDA GUI. Users using the GUI on which storing user lists has been disabled will be able to see all developments and assignments available for the EDA server, even if access controls prevent them from operating on those developments or assignments.

Note: This setting is only read when the Enterprise User Identity Mapping (enterpriseUserMapType) is set to EM-EM.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the sregistry command set. The settings can be made to either the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, or the \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file in the server's installation. Settings in the SiteRegistry.reg file will apply to all servers in the installation. The server must be restarted for the changes to take effect. An alternative to restarting the SyncServer is to use the sregistry reset command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\EDA\NoAssignmentUserList=dword:0 (Default)
```

Allowed Values

Key value	Description
NoAssignmentUserList=dword:0	Store user lists on the DesignSync server. This allows the SDA GUI to filter the developments and assignments visible to the user to include only the ones accessible to the user.(Default)
NoAssignmentUserList=dword:1	Do not store user lists on the DesignSync server. This improves the performance of the SDA GUI.

Related Topics

Setting Up Enterprise Design User Mappings

Enabling Enterprise Synchronization (UpdateScheme)

The Enterprise Design system allows you to synchronize data between the DesignSync and Platform servers. When updates are made to modules in the DesignSync system, DesignSync can automatically synchronize those changes with the corresponding Enterprise Design Objects based on the setting you select. This option can be set from the Enterprise Servers tab of SyncAdmin.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Enterprise\UpdateScheme="none"
```

Allowed Values

Key value	Description
<code>UpdateScheme="none"</code>	Enterprise Synchronization is performed manually. (Default)
<code>UpdateScheme="alarmtrigger"</code>	Uses the alarm trigger mechanism to determine when to check for queued updates to send to the Enterprise server.

Related Topics

[Enterprise Servers](#)

DesignSync Web UI Registry Settings

RevisionControl note statistics (CountRevCtlAttachments)

The `CountRevCtlAttachments` registry key is set when the option to "Count objects operated on when calculating server statistics" is used, in either SyncAdmin's Site Options RC Notes pane or DesignSync Web UI's Administer Server Settings RC Notes tab. If you use DesignSync Web UI's Administer Server to modify RevisionControl note

generation settings, you do not need to restart the server (or reset its registry files) for the changes to take effect.

The default value is `dword:0`, with statistics based on the number of RevisionControl notes. Set the value to `dword:1` to base statistics on the number of objects recorded in RevisionControl notes.

This registry setting pertains to a DesignSync Web UI server. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\ProjectSync\Stats\CountRevCtlAttachments=dword:0
```

Allowed Values

Key value	Description
<code>CountRevCtlAttachments=dword:0</code>	Bases reported note statistics on number of RevisionControl notes.
<code>CountRevCtlAttachments=dword:1</code>	Bases reported note statistics on number of objects recorded in RevisionControl notes.

Related Topics

RevisionControl note generation (`GenerateNote`)

Hierarchical Query time-out (`TokenLifeTime`)

Setting RevisionControl Note Generation

Revision Control Notes Option

Enabling LDAP

ProjectSync User's Guide: Constructing Standard Queries

RevisionControl note generation (`GenerateNote`)

DesignSync Data Manager Administrator's Guide

These registry keys are set when SyncAdmin's Site Options RC Notes pane is used. Or when DesignSync Web UI's Administer Server Settings RC Notes tab is used. If you use DesignSync Web UI's Administer Server to modify RevisionControl note generation settings, you do not need to restart the server (or reset its registry files) for the changes to take effect.

The default value for all RevisionControl note generation settings is `dword:0` (disabled). Set the value to `dword:1` to enable RevisionControl note generation.

These registry settings pertain to a DesignSync Web UI server. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's

`$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Settings

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\CiGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\CoLockGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\CoNoLockGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\MkbranchGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\MvCommandsGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\RmCommandsGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\RetireGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\SwitchlockerGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\TagGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\UnlockGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\MkmodGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\RmmodGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\LockmodGenerateNote=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Notes\UpgrademoGenerateNote=dword:0
```

Allowed Values

Key value	Description
<note_specifier>GenerateNote=dword:0	Disallows RevisionControl note generation. (Default)
<note_specifier>GenerateNote=dword:1	Allows RevisionControl note generation.

Related Topics

RevisionControl note statistics (CountRevCtlAttachments)

Hierarchical Query time-out (TokenLifeTime)

Setting RevisionControl Note Generation

Revision Control Notes Option

Enabling LDAP

ProjectSync User's Guide: Constructing Standard Queries

Hierarchical Query time-out (TokenLifeTime)

This setting controls the timeout for a hierarchical query (from DesignSync Web UI's Standard Query) if the origin server (from which the query is invoked) detects more than a five minute difference in its system clock and that of a referenced server. This expiration time is intended to avoid a replay attack.

DesignSync Data Manager Administrator's Guide

The default `TokenLifeTime` value is 300 seconds, expressed in hexadecimal as `dword:12C`.

This registry setting pertains to a DesignSync Web UI server. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the `SyncServer` is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\TokenLifeTime=dword:12C
```

Allowed Values

Key Value	Description
<code>TokenLifeTime=dword:12C</code>	Sets query time-out as 300 seconds. (Default)
<code>TokenLifeTime=dword:<value></code>	Sets query time-out to specified value.

Related Topics

RevisionControl note generation (`GenerateNote`)

RevisionControl note statistics (`CountRevCtlAttachments`)

Setting RevisionControl Note Generation

Revision Control Notes Option

Enabling LDAP

ProjectSync User's Guide: Constructing Standard Queries

Enable ACAdmin (`ACAdminEnabled`)

The `ACAdminEnabled` registry key is used to enable a web interface to the Access Controls. Using `ACAdmin`, you can create, remove, modify, or reset the Access Controls for your servers. For more information on Access Controls, or `ACAdmin`, see the *ENOVIA Synchronicity Access Control Guide*.

By default, ACAdmin is disabled. To enable ACAdmin, set `ACAdminEnabled` to `dword:1`.

This registry setting pertains to a SyncServer. To modify registry values programmatically, use the `sregistry` command set. The settings can be made to either the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, or the `$$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file in the server's installation. Settings in the `SiteRegistry.reg` file will apply to all servers in the installation. The server must be restarted, for the changes to take effect. An alternative to restarting the SyncServer is to use the `sregistry reset` command in a server-side script to re-read the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\AccessControl\
ACAdminEnabled=dword:0
```

Allowed Values

Key value	Description
<code>ACAdminEnabled=dword:0</code>	Disables ACAdmin. (Default)
<code>ACAdminEnabled=dword:1</code>	Enables ACAdmin.

Registry Settings for Mirrors

Repository Server Registry Settings

Check tags for mirror update (`CheckChangeAffectsMirror`)

This setting controls when the mirror is updated after a `tag`, `ci`, or `mkbranch` operation.

When this key is enabled, (`dword:1`), the default value, the mirror is updated only when the selector defined for the mirror contains a tag affected by the `ci`, `tag`, or `mkbranch` operation. The affected tag is a version or branch tag added or removed with the `tag` or `mkbranch` operation, or for a `ci` operation, a version tag added to the new version and any branch tag on the affected branch. The key is checked by the MPD at the RS to determine if a change should be pushed to the MAS. On the MAS, the push is verified by the MAD to determine whether the mirror should be updated.

When this key is not enabled (`dword:0`), the mirrors are updated whenever the mirrored source vault is updated.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\CheckChangeAffectsMirror=dword:1
```

Allowed Values

Key value	Description
CheckChangeAffectsMirror=dword:0	Disables check; mirrors update whenever source vault is updated.
CheckChangeAffectsMirror=dword:1	Enables check for affected tags in the selector defined for the mirror before allowing it to update. (Default)

Related Topics

Failure notify time (MPDPushFailureNotifyTime)

Failure retries (MPDPushFailureRetries)

Failure notify interval (MPDNotifyInterval)

Log file size (LogFileMaxSize)

Set Do Not Cache Property on Transaction in the Mirror (IncludeDoNotCacheProperty)

This setting determines how transactions involving noncacheable objects are handled by the Repository Server. Non-cacheable objects are present within a URL that is defined as non-cacheable and cannot be placed in a module or file cache. When using scripted mirrors, DesignSync recommends that transaction not be generated for non-cacheable objects. For more information on Non-cacheable objects, see . For more information on setting up scripted mirrors, see Creating Scripted Mirrors.

When this key is enabled, (`dword:1`), the default value, the repository server generates transactions for uncacheable objects.

When this is key is not enabled (`dword:0`),the repository server does not generate transactions for uncacheable objects.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\RS\SendUncacheableTransaction=dword:1
```

Allowed Values

Key value	Description
<code>SendUncacheableTransaction=dword:0</code>	Does not generate transactions for uncacheable objects. This is a recommended for scripted mirrors.
<code>SendUncacheableTransaction=dword:1</code>	Generates transactions for objects marked as uncacheable.

Related Topics

Log file size (LogFileMaxSize)

This setting controls the maximum size of the RS's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/share/content/mirrors/logs/mpdlog.txt` file, in kilobytes, before rotating the existing log file. The `mpdlog.txt` file is displayed when you click on the `View All Mirrors Log` button, from DesignSync Web UI's `Show RS Status` panel. The default value is 1000 (1 megabyte), represented in hexadecimal as `dword:3E8`.

Note: The `LogFileMaxSize` value must be at least 100, represented in hexadecimal as `dword:64`.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\LogFileMaxSize=dword:3E8
```

Allowed Values

Key value	Description
LogFileMaxSize=dword:64	Sets the maximum size of the RS's log file to 100 kilobytes. (Minimum Value)
LogFileMaxSize=dword:3E8	Sets the maximum size of the RS's log file to 1000 (1 megabyte). (Default)
LogFileMaxSize=dword:<value>	Sets the maximum size of the RS's log file to specified value in kilobytes.

Related Topics

Failure notify time (MPDPushFailureNotifyTime)

Failure retries (MPDPushFailureRetries)

Failure notify interval (MPDNotifyInterval)

Check tags for mirror update (CheckChangeAffectsMirror)

Failure notify interval (MPDNotifyInterval)

This setting controls how often the RS administrator is notified, in hours, about changes failing to be sent to a MAS. The default value is 24, represented in hexadecimal as `dword:18`.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MPD\MPDNotifyInterval=dword:18`

Allowed Values

Key value	Description
MPDNotifyInterval=dword:18	Sets notification of changes failing to be sent to a MAS to 24 hours. (Default)
MPDNotifyInterval=dword:<value>	Sets notification of changes failing to be sent to a MAS to specified value.

Related Topics

Failure notify time (MPDPushFailureNotifyTime)

Failure retries (MPDPushFailureRetries)

Log file size (LogFileMaxSize)

Check tags for mirror update (CheckChangeAffectsMirror)

Failure notify time (MPDPushFailureNotifyTime)

This setting controls the amount of time (in minutes) before sending email to the RS administrator, when an update to the RS's vault fails to be sent to a Mirror Administration Server (MAS). The default value is 15, represented in hexadecimal as `dword:F`.

Note: The `MPDPushFailureRetries` value below must also be met, for failure notification to be sent.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MPD\MPDPushFailureNotifyTime=dword:F
```

Allowed Values

Key value	Description
<code>MPDPushFailureNotifyTime=dword:F</code>	Sets failure notification time to 15 minutes. (Default)
<code>MPDPushFailureNotifyTime=dword:<value></code>	Sets failure notification time to specified value.

Related Topics

Failure retries (MPDPushFailureRetries)

Failure notify interval (MPDNotifyInterval)

Log file size (LogFileMaxSize)

Check tags for mirror update (CheckChangeAffectsMirror)

Failure retries (MPDPushFailureRetries)

This setting controls the number of times the RS tries (unsuccessfully) to send changes to a MAS, before designating the attempt as a failure. The default value is 2 tries, represented in hexadecimal as `dword:2`.

Note: The `MPDPushFailureNotifyTime` value above must also be met, for failure notification to be sent.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MPD\MPDPushFailureRetries=dword:2
```

Allowed Values

Key value	Description
<code>MPDPushFailureRetries=dword:2</code>	Sets number of unsuccessful RS tries before designating attempt as a failure to 2. (Default)
<code>MPDPushFailureRetries=dword:<value></code>	Sets number of unsuccessful RS tries before designating attempt as a failure to specified value.

related Topics

Failure notify time (`MPDPushFailureNotifyTime`)

Failure notify interval (`MPDNotifyInterval`)

Log file size (`LogFileMaxSize`)

Check tags for mirror update (`CheckChangeAffectsMirror`)

Send Uncachable Transactions to the Server (SendUncacheableTransaction)

This setting determines how transactions involving noncacheable objects are handled by the Repository Server. Non-cacheable objects are present within a URL that is defined as non-cacheable and cannot be placed in a module or file cache. When using

scripted mirrors, DesignSync recommends that transaction not be generated for non-cacheable objects. For more information on Non-cacheable objects, see . For more information on setting up scripted mirrors, see Creating Scripted Mirrors.

When this key is enabled, (dword:1), the default value, the repository server generates transactions for uncacheable objects.

When this is key is not enabled (dword:0),the repository server does not generate transactions for uncacheable objects.

This registry setting pertains to a SyncServer that is configured to be a Repository Server (RS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\RS\SendUncacheableTransaction=dword:1
```

Allowed Values

Key value	Description
SendUncacheableTransaction=dword:0	Does not generate transactions for uncacheable objects. This is a recommended for scripted mirrors.
SendUncacheableTransaction=dword:1	Generates transactions for objects marked as uncacheable.

Related Topics

Mirror Administration Server Registry Settings

Enable general disk space check (Active)

This setting activates the low disk space check for the disk used by the MAS for general information (such as MUP logs).

The default <value> is 1 (enabled). To disable the low disk space check, set the <value> to 0.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's

DesignSync Data Manager Administrator's Guide

`$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

`HKEY_LOCAL_MACHINE\Software\Synchronicity\MirrorGeneral\DiskLimits\Active=dword:1`

Allowed Values

Key value	Description
<code>Active=dword:0</code>	Disallows check for low disk space in disks used by the MAS.
<code>Active=dword:1</code>	Allows check for low disk space in disks used by the MAS. (Default)

Related Topics

Failure notify time (`MADUpdateFailureNotifyTime`)

Failure retries (`MADUpdateFailureRetries`)

Maximum MUPs running in Parallel (`MUPsMaxPerMAD`)

Ignore Locked Objects in the Mirror (`PopulateIgnoreLockedObjects`)

Daily update (`NumberHoursToPopulateAllMirrors`)

Log file size for mirrors (`LogFileMaxSize`)

Absolute value for general disk space check (`FreeSpaceAbsWarn`)

Relative value for general disk space check (`FreeSpaceRelWarn`)

Absolute value for mirror directory disk space check (`FreeSpaceAbsWarn`)

Relative value for mirror directory disk space check (`FreeSpaceRelWarn`)

Enable mirror directory disk space check (`Active`)

Secondary mirror fetch (`SMAAllowFetchFromRepository`)

Update command (`MUPNewVersForPopulate`)

Populate options for all mirrors on the MAS (`MUPNewVersForPopulate`)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Enable mirror directory disk space check (Active)

This setting controls activation of the low disk space check for mirror directory disks.

The default *<value>* is 1 (enabled). To disable the low disk space check, set the *<value>* to 0.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the registry command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\Mirrors\DiskLimits\Active=dword:1

Allowed Values

Key value	Description
Active=dword:0	Disallows check for low disk space in disks used by the MAS.
Active=dword:1	Allows check for low disk space in disks

	used by the MAS.(Default)
--	---------------------------

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

This setting controls whether an automatically generated mirror directory is deleted when it becomes empty.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
General\Mirrors\Definitions\MAS\AutoDeleteEmptyMirrors=dword:1
```

Allowed Values

Key value	Description
<code>AutoDeleteEmptyMirrors=dword:0</code>	Disallows automatic deletion of mirror directories.
<code>AutoDeleteEmptyMirrors=dword:1</code>	Allows automatic deletion of mirror directories. (Default)

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

DesignSync Data Manager Administrator's Guide

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Checkout options for all mirrors on the MAS (CoOptions)

This setting controls the options used when updating all mirrors on a MAS via the `co` command.

You can change the options used by the `co` command, when fetching data into the mirror. The default options used are shown in the `CoOptions` setting above.

This registry setting pertains to a `SyncServer` that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the

sregistry command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Note: to set the co options for a specific mirrors, see Co options for a specific mirror (CoOptions).

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MUP\CoOptions="-force -retain"
```

Allowed Values

The allowed values for this registry setting are located in Checking out design files.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

DesignSync Data Manager Administrator's Guide

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Enable Automatic Requeueing (AutoRequeue)

This setting controls enabling/disabling automatic requeueing and the number of attempts to requeue. A value of 0 (zero) disables automatic requeueing. Any integer value sets the number of retry attempts, to a maximum of 20 retries.

When using automatic requeue, you may want to verify that, if you log transactions, your SaveLogFiles registry setting is set to a value of 2, which records error on a per-mirror basis, allowing you to easily locate mirror failures.

Automatic Retry must be enabled for automatic transaction requeueing. For more information, see Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry).

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the registry command set. The settings must be made to the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\AutoRequeue=dword:0
```

Allowed Values

Key value	Description
AutoRequeue=dword:0	Disables automatic requeuing. (Default)
AutoRequeue=dword:#	Sets automatic requeue attempts to the value specified.
AutoRequeue=dword:5	For any failed transactions, will requeue the transaction up to 5 times. If after 5 attempts, the transactions still fails, the transaction will appear as a failure in the MAS status panel for manual requeuing after the error has been fixed.
AutoRequeue=dword:20	The maximum value of retry attempts. After 20 attempts, DesignSync will no longer attempt to retry the transaction, but the transaction can be removed or requeued manually after the error has been corrected.

Related Topics

Verifying that the Mirror System is Running (HeartBeatInterval)

Save all mup log files (SaveLogFiles)

Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry)

Checkout options for a specific mirror (CoOptions)

The options used when updating a specific mirror via the `co` command. In the example below the mirror is named `m1`.

You can change the options used by the `co` command, when fetching data into the mirror. The default options used are shown in the `CoOptions` setting above.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Note: To update the `co` options for all mirrors, see `Co options for all mirrors on the MAS (CoOptions)`.

Registry Setting

DesignSync Data Manager Administrator's Guide

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\<m1>\CoOptions="-retain -force"
```

Allowed Values

The allowed values for this registry setting are located in Checking out design files.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Absolute value for general disk space check (FreeSpaceAbsWarn)

This setting controls when the amount of free disk space on the disk used by the MAS for general information (such as MUP logs) falls below this threshold.

<value> is the number of Mb, represented in hexadecimal. The default *<value>* is 20, equating to 32Mb.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\MirrorsGeneral\DiskLimits\FreeSpaceAbsWarn=dword:20
```

Allowed Values

Key value	Description
FreeSpaceAbsWarn=dword:20	Sets threshold for low disk space warning to 32Mb. (Default)
FreeSpaceAbsWarn=dword:<value>	Sets threshold for low disk space warning to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

DesignSync Data Manager Administrator's Guide

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Relative value for general disk space check (FreeSpaceRelWarn)

This setting controls when the percentage of free disk space on the disk used by the MAS for general information (such as MUP logs) falls below this threshold.

The FreeSpaceRelWarn value is the percentage of total disk space, represented in hexadecimal. The default <value> is 0.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\MirrorsGeneral\DiskLimits\FreeSpaceRelWarn=dword:0
```

Allowed Values

Key value	Description
FreeSpaceRelWarn=dword:0	Sets threshold for low disk space percentage warning to 0. (Default)
FreeSpaceRelWarn=dword:<value>	Sets threshold for low disk space percentage warning to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

DesignSync Data Manager Administrator's Guide

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

This setting controls e-mail notifications for users on the mirror's notification list when the amount of free disk space for a mirror directory's disk falls below this threshold.

The FreeSpaceAbsWarn value is the number of Mb, represented in hexadecimal. The default value is 20, equating to 32Mb.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's `$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\Mirrors\DiskLimits\FreeSpaceAbsWarn=dword:20

Allowed Values

Key value	Description
FreeSpaceAbsWarn=dword:20	Sets threshold for low disk space warning to 32Mb. (Default)
FreeSpaceAbsWarn=dword:<value>	Sets threshold for low disk space warning to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

This value controls sending e-mail to users on the mirror's notification list when the percentage of free disk space for a mirror directory's disk falls below this threshold.

The FreeSpaceRelWarn value is the percentage of total disk space, represented in hexadecimal. The default value is 0.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Mirrors\DiskLimits\FreeSpaceRelWarn=dword:0
```

Allowed Values

Key value	Description
FreeSpaceRelWarn=dword:0	Sets threshold for low disk space percentage warning to 0. (Default)
FreeSpaceRelWarn=dword:<value>	Sets threshold for low disk space percentage warning to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Log file size for mirrors (LogFileMaxSize)

This setting controls the maximum size of the MAS's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/share/content/mirrors/logs/mpdlog.txt` file, in kilobytes, before rotating the existing log file. The `madlog.txt` file is displayed when you click on the `View All Mirrors Log` button, from the DesignSync web interface `Show MAS Status` panel. The default value is 1000 (1 megabyte), represented in hexadecimal as `dword:3E8`.

Note: The `LogFileMaxSize` value must be at least 100, represented in hexadecimal as `dword:64`.

This registry setting pertains to a `SyncServer` that is configured to be a `Mirror Administration Server (MAS)`. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\LogFileMaxSize=dword:3E8
```

Allowed Values

Key value	Description
<code>LogFileMaxSize=dword:64</code>	Sets the maximum size of the MAS's log file to 100 kilobytes. (Minimum Value)
<code>LogFileMaxSize=dword:3E8</code>	Sets the maximum size of the MAS's log file to 1000 (1 megabyte). (Default)
<code>LogFileMaxSize=dword:<value></code>	Sets the maximum size of the MAS's log file to specified value in kilobytes.

Related Topics

Failure notify time (`MADUpdateFailureNotifyTime`)

Failure retries (`MADUpdateFailureRetries`)

Maximum MUPs running in Parallel (`MUPsMaxPerMAD`)

Ignore Locked Objects in the Mirror (`PopulateIgnoreLockedObjects`)

Daily update (NumberHoursToPopulateAllMirrors)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Control Mcache Mode option (MADAddMcacheModeServer)

This setting controls whether the `'-mcachemode server'` option is passed through the Populate options for a specific mirror to the MUP when the mirror is defined with

Mcache Mode set to server. By default, this is set to allow the `-mcacheMode server` option (dword:1) to pass-through. To disable the option pass-through for ' `-mcacheMode server`', set the value of the dword to 0 (dword:0). This can be useful if the mcache mode and the mcache paths are defined in the registry.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\MADAddMcacheModeServer=dword:1
```

Allowed Values

Key value	Description
<code>MADAddMcacheModeServer=dword:0</code>	Disallows <code>-mcacheMode server</code> option to pass-through.
<code>MADAddMcacheModeServer=dword:1</code>	Allows <code>-mcacheMode server</code> option to pass-through. (Default)

Related Topics

Failure notify time (`MADUpdateFailureNotifyTime`)

Failure retries (`MADUpdateFailureRetries`)

Maximum MUPs running in Parallel (`MUPsMaxPerMAD`)

Ignore Locked Objects in the Mirror (`PopulateIgnoreLockedObjects`)

Daily update (`NumberHoursToPopulateAllMirrors`)

Log file size for mirrors (`LogFileMaxSize`)

Absolute value for general disk space check (`FreeSpaceAbsWarn`)

Relative value for general disk space check (`FreeSpaceRelWarn`)

Enable general disk space check (`Active`)

Absolute value for mirror directory disk space check (`FreeSpaceAbsWarn`)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Failure notify interval for mirrors (MADNotifyInterval)

This setting specifies how often the MAS administrator is notified, in hours, about a mirror on the MAS failing to be updated. The default value is 24, represented in hexadecimal as `dword:18`.

Note: The value must be at least 0.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

DesignSync Data Manager Administrator's Guide

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\MADNotifyInterval=dword:18

Allowed Values

Key value	Description
MADNotifyInterval=dword:0	Sets the notification interval of a failed MAS update to 0 hours. (Minimum Value)
MADNotifyInterval=dword:18	Sets the notification interval of a failed MAS update to 24 hours. (Default)
MADNotifyInterval=dword:<value>	Sets the notification interval of a failed MAS update to the specified value.

Failure notify time for mirrors (MADUpdateFailureNotifyTime)

This setting controls the amount of time (in minutes) before sending email to the MAS administrator, when a mirror failed to be updated. The default value is 60, represented in hexadecimal as `dword:3C`.

Notes: The value must be at least 0. The `MADFailureRetries` value below must also be met, for failure notification to be sent.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\MADUpdateFailureNotifyTime=dword:F

Allowed Values

Key value	Description
MADUpdateFailureNotifyTime=dword:0	Sets time before failure notification to be immediate. (Minimum Value)
MADUpdateFailureNotifyTime=dword:3C	Sets time before failure notification to 60 minutes. (Default)
MADUpdateFailureNotifyTime=dword:<value>	Sets time before failure

	notification to specified value.
--	----------------------------------

Related Topics

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Failure retries for mirrors (MADUpdateFailureRetries)

This setting controls the number of times an attempt is made to update a mirror on the MAS, before designating the update a failure. The default value is 2 tries, represented in hexadecimal as `dword:2`.

Notes: The value must be at least 0. The `MADUpdateFailureNotifyTime` value above must also be met, for failure notification to be sent.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\MADUpdateFailureRetries=dword:2
```

Allowed Values

Key value	Description
<code>MADUpdateFailureRetries=dword:0</code>	Sets number of attempts to update to 0. (Minimum value)
<code>MADUpdateFailureRetries=dword:2</code>	Sets number of attempts to update to 2. (Default)
<code>MADUpdateFailureRetries=dword:<value></code>	Sets number of attempts to update to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

This setting controls the maximum number of MUP processes that will run in parallel. After the maximum value is reached, any additional transactions will queue up to await a free process. As the processes terminate, the queued processes are executed in the order they were received.

The default value is 20 concurrent processes (hexadecimal value 14).

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\MUPsMaxPerMAD=dword:14
```

Allowed Values

Key value	Description
MUPsMaxPerMAD=dword:14	Sets maximum number of MUPs that will run in parallel to 20. (Default)
MUPsMaxPerMAD=dword:<value>	Sets maximum number of MUPs that will run in parallel to specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (`FreeSpaceRelWarn`)

Enable mirror directory disk space check (`Active`)

Secondary mirror fetch (`SMAAllowFetchFromRepository`)

Update command (`MUPNewVersForPopulate`)

Populate options for all mirrors on the MAS (`MUPNewVersForPopulate`)

Co options for all mirrors on the MAS (`CoOptions`)

Populate options for a specific mirror (`PopulateOptions`)

Co options for a specific mirror (`CoOptions`)

Control Mcache Mode option (`MADAddMcacheModeServer`)

Upgrade mirror properties (`PerformUpgrade`)

Replicate Scrub Time (`ReplicateScrubTime`)

Replicate Scrub Age (`ReplicateScrubAge`)

Auto Deleting of Empty Mirrors (`AutoDeleteEmptyMirrors`)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Update command (`MUPNewVersForPopulate`)

This setting controls whether a mirror is updated via `co` or `populate`.

By default, when the number of new versions/objects to be fetched into a mirror is fewer than 100, `co` is used to fetch the objects. If at least 100 objects are being fetched, `populate` is used. The default value of 100 is represented in hexadecimal as `dword:64`.

Note: The `MUPNewVersForPopulate` value must be at least 1.

This registry setting pertains to a `SyncServer` that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `registry` command set. The settings must be made to the server's

DesignSync Data Manager Administrator's Guide

`$$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MUP\MUPNewVersForPopulate=dword:64`

Allowed Values

Key value	Description
<code>MUPNewVersForPopulate=dword:1</code>	Sets threshold number of objects being fetched above which <code>co</code> is used and below which <code>populate</code> is used to 1. (Minimum Value)
<code>MUPNewVersForPopulate=dword:64</code>	Sets threshold number of objects being fetched above which <code>co</code> is used and below which <code>populate</code> is used to 100. (Default)
<code>MUPNewVersForPopulate=dword:<value></code>	Sets threshold number of objects being fetched above which <code>co</code> is used and below which <code>populate</code> is used to specified value.

Related Topics

Failure notify time (`MADUpdateFailureNotifyTime`)

Failure retries (`MADUpdateFailureRetries`)

Maximum MUPs running in Parallel (`MUPsMaxPerMAD`)

Ignore Locked Objects in the Mirror (`PopulateIgnoreLockedObjects`)

Daily update (`NumberHoursToPopulateAllMirrors`)

Log file size for mirrors (`LogFileMaxSize`)

Absolute value for general disk space check (`FreeSpaceAbsWarn`)

Relative value for general disk space check (`FreeSpaceRelWarn`)

Enable general disk space check (`Active`)

Absolute value for mirror directory disk space check (`FreeSpaceAbsWarn`)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Daily update (NumberHoursToPopulateAllMirrors)

This setting controls the time-span, in hours, during which each mirror on the MAS will issue an incremental populate. The default value is 24, represented in hexadecimal as `dword:18`. The default value of 24 means that a mirror encountering a problem will only be out-of-date for at most 24 hours.

Notes: The value must be at least 1. The mirrors on a MAS are not updated simultaneously; their updates are spaced throughout the `NumberHoursToPopulateAllMirrors` period.

This registry setting pertains to a `SyncServer` that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

DesignSync Data Manager Administrator's Guide

Registry Setting

HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\NumberHoursToPopulateAllMirrors=dword:18

Allowed Values

Key value	Description
NumberHoursToPopulateAllMirrors=dword:1	Sets time-span during which mirrors on the MAS will issue an incremental populate to 1 hour. (Minimum Value)
NumberHoursToPopulateAllMirrors=dword:18	Sets time-span during which mirrors on the MAS will issue an incremental populate to 24 hours. (Default)
NumberHoursToPopulateAllMirrors=dword:<value>	Sets time-span during which mirrors on the MAS will issue an incremental populate to the specified value.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Upgrade mirror properties (PerformUpgrade)

This setting controls whether to automatically upgrade mirror properties used by the `mirror wheremirrored` command.

Mirrors created on DesignSync servers prior to the implementation of "wheremirrored" support have "needs-upgrade" as a value for most of their properties reported by the `mirror wheremirrored` command after the server has been updated to support "wheremirrored". Those properties are automatically upgraded when the MAS's mirror daemon is reset, or the MAS is restarted. To not upgrade mirror properties automatically, set the `PerformUpgrade` value to `dword:0`.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's

DesignSync Data Manager Administrator's Guide

`$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAS\PerformUpgrade=dword:1`

Allowed Values

Key value	Description
<code>PerformUpgrade=dword:0</code>	Disallows mirror properties to upgrade automatically.
<code>PerformUpgrade=dword:1</code>	Allows mirror properties to upgrade automatically. (Default)

Related Topics

Failure notify time (`MADUpdateFailureNotifyTime`)

Failure retries (`MADUpdateFailureRetries`)

Maximum MUPs running in Parallel (`MUPsMaxPerMAD`)

Ignore Locked Objects in the Mirror (`PopulateIgnoreLockedObjects`)

Daily update (`NumberHoursToPopulateAllMirrors`)

Log file size for mirrors (`LogFileMaxSize`)

Absolute value for general disk space check (`FreeSpaceAbsWarn`)

Relative value for general disk space check (`FreeSpaceRelWarn`)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (`FreeSpaceAbsWarn`)

Relative value for mirror directory disk space check (`FreeSpaceRelWarn`)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (`SMAAllowFetchFromRepository`)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

This setting controls whether the mirror stops updating when locked objects are encountered. When a mirror workspace contains locked objects, it can cause the MUP to error when updating the mirror objects. By default, the MUP considers the update to have failed when locked objects are encountered (dword:0). To ignore the errors and continue updating the mirror successfully, enable this setting (dword:1).

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\PopulateIgnoreLockedObjects=dword:0
```

Allowed Values

Key value	Description
-----------	-------------

PopulateIgnoreLockedObjects=dword:0	Disallows MUPs to continue updating after encountering a locked object. (Default)
PopulateIgnoreLockedObjects=dword:1	Allows MUPs to continue updating after encountering a locked object.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Populate options for all mirrors on the MAS (PopulateOptions)

This setting controls the `populate` options used when updating all mirrors on a MAS.

You can change the options used for populating mirrors by setting the `PopulateOptions` registry key. For example, to specify that empty directories remain in the mirror, set `PopulateOptions="<-force> <-get> <-retain> <-emptydirs>"`

Out-of-the-box, `populate` uses the `-incremental` option by default. If you set the MAS installation's default populate behavior to `-full` (via SyncAdmin's General Command Defaults tab), you should specify the `-incremental` option for mirror updates. When a `populate` is run with the `-full` option's behavior, mirror updates traverse the entire vault hierarchy for the mirrored project and may take a long time to complete.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Note: To set the `populate` options for a specific mirror, see `Populate options for a specific mirror (PopulateOptions)`.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MUP\PopulateOptions="<value>"
```

Allowed Values

The allowed values for this registry setting are located in `Populating your work area`.

Related Topics

DesignSync Data Manager Administrator's Guide

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Populate options for a specific mirror (PopulateOptions)

This setting controls how populate operates on a specific mirror.

For example, assuming a mirror named `m1`, to specify that empty directories remain in the `m1` mirror, set `PopulateOptions="<-force> <-retain> <-emptydirs>"`

Note: To set the populate options for all mirrors, see Populate options for all mirrors on the MAS (PopulateOptions).

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `registry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\<m1>\PopulateOptions="<-retain> <-force> <-noemptydirs>"
```

Allowed Values

The allowed values for this registry setting are located in *Populating your work area*.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

DesignSync Data Manager Administrator's Guide

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Replicate Scrub Age (ReplicateScrubAge)

This setting controls the number of days, as an integer, that replicated data in the dynamic folder remains untouched before the data is scrubbed from the repository. The default value is 7 (days).

Note: This must be an integer value.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the

sregistry command set. The settings must be made to the server's \$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\ReplicateScrubAge="7"
```

Allowed Values

Key value	Description
ReplicateScrubAge=7	Replicated data is set to be stored for seven days. (Default)
ReplicateScrubAge=<value>	Replicated data is set to be stored for the specified number of days.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAllowFetchFromRepository)

DesignSync Data Manager Administrator's Guide

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Replicate Scrub Time (ReplicateScrubTime)

This setting sets the time, in the format hh:mm, that the data replicate scrubber runs on the system. The default value is 1:00, (1:00 AM).

Note: Time values should be entered in a 24 hour format.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\ReplicateScrubTime="<hh>:<mm>"
```

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Secondary mirror fetch (SMAAllowFetchFromRepository)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

Secondary mirror fetch (SMAllowFetchFromRepository)

Controls the behavior of secondary mirror fetches from the Repository Server.

If the primary mirror is not available, the default behavior is for a secondary mirror's update to fail. That is represented by a value of `dword:0`. For secondary mirrors to fetch data from the Repository Server if the primary mirror is not available, set the `SMAllowFetchFromRepository` value to `dword:1`.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\SMAllowFetchFromRepository=dword:0
```

Allowed Values

Key value	Description
<code>SMAllowFetchFromRepository=dword:0</code>	Sets secondary mirror updates to fail if primary mirror is unavailable. (Default)
<code>SMAllowFetchFromRepository=dword:1</code>	Sets secondary mirror to fetch data from repository if primary mirror is unavailable.

Related Topics

Failure notify time (MADUpdateFailureNotifyTime)

Failure retries (MADUpdateFailureRetries)

Maximum MUPs running in Parallel (MUPsMaxPerMAD)

Ignore Locked Objects in the Mirror (PopulateIgnoreLockedObjects)

Daily update (NumberHoursToPopulateAllMirrors)

Log file size for mirrors (LogFileMaxSize)

Absolute value for general disk space check (FreeSpaceAbsWarn)

Relative value for general disk space check (FreeSpaceRelWarn)

Enable general disk space check (Active)

Absolute value for mirror directory disk space check (FreeSpaceAbsWarn)

Relative value for mirror directory disk space check (FreeSpaceRelWarn)

Enable mirror directory disk space check (Active)

Update command (MUPNewVersForPopulate)

Populate options for all mirrors on the MAS (MUPNewVersForPopulate)

Co options for all mirrors on the MAS (CoOptions)

Populate options for a specific mirror (PopulateOptions)

Co options for a specific mirror (CoOptions)

Control Mcache Mode option (MADAddMcacheModeServer)

Upgrade mirror properties (PerformUpgrade)

Replicate Scrub Time (ReplicateScrubTime)

Replicate Scrub Age (ReplicateScrubAge)

Auto Deleting of Empty Mirrors (AutoDeleteEmptyMirrors)

Registry Files

Warning Threshold for Low Disk Space

Finding Mirrored Data

DesignSync Client's Mirror Functionality Registry Settings

Check for matching selector (ClientUseMirrorCheck)

This setting controls whether the selector sent by a `ci` command matches the selector defined for the mirror. By default, any `ci` operation in a workspace associated with a mirror directory requires that the workspace's selector exactly matches the mirror's selector, for the "write-through" of data from the workspace to the mirror. Similarly, `co -mirror`, `populate -mirror` and `cancel -mirror` require that the workspace's selector exactly matches the mirror's selector. Otherwise, the operation fails. This

requirement ensures that the mirror files linked to from users' workspaces are the file versions that users expect. The selector check also pertains to the [co]populate - from local [-get|-lock]" optimization, applied when possible by default, to copy data from the mirror.

The selector check can be disabled. However, that may result in users linking to incorrect versions in the mirror. For instance, if the workspace was setmirror to the wrong mirror. Or if their workspace selector and the mirror's selector resolve to a different versions.

To disable the selector check, set ClientUseMirrorCheck to dword:0 .

The default behavior, of requiring the workspace selector to exactly match the mirror's selector, is a ClientUseMirrorCheck value of dword:1 .

Prior to Developer Suite 4.0, mirrors were only supported for the Latest versions on branch "1". To enforce that behavior, of only allowing mirror operations to succeed if it is for the Latest version on the main branch "1" ("Trunk" by default), set a ClientUseMirrorCheck value of dword:2 .

This registry setting pertains to DesignSync client "write-through" behavior. A client "write-through" is a ci (in any mode, not just -mirror) causing the file version to be fetched into the mirror associated with the workspace. This ensures that the newly created data is immediately available to the site from which the new data was created.

To modify registry values programmatically, use the sregistry command set. The settings must be made to the client installation's \$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg file, and DesignSync clients restarted, for the changes to take effect. To restart a DesignSync client, exit and restart your DesSync, dssc, or stlc session. If you are using dss or stcl, use the syncdadmin command to restart syncd. An alternative to restarting a DesignSync client is to use the sregistry reset command in the current DesignSync session, to re-load the registry files.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\Client\General\Optimiz  
ations\ClientUseMirrorCheck=dword:1
```

Allowed Values

Key value	Description
Optimizations\ClientUseMirrorCheck=dword:0	Disables selector check.
Optimizations\ClientUseMirrorCheck=dword:1	Requires workspace selector to match exactly to mirror's

	selector. (Default)
Optimizations\ClientUseMirrorCheck=dword:2	Only allows mirror operations to succeed if it is for the latest version of the main branch.

Related Topics

Whether write through occurs (EnableClientMirrorUpdate)

Symbolic link Handling (ManagedLinkInMirrorMode)

Site Options

Default Fetch State (DefaultFetchType)

Whether write through occurs (EnableClientMirrorUpdate)

This setting controls whether mirror write through is allowed or not. The `EnableClientMirrorUpdate` registry key is set when SyncAdmin's Site Options pane is used, to specify "Mirror Write-through" behavior.

To "Prohibit mirror write-through", set an `EnableClientMirrorUpdate` value of `dword:0`.

The default behavior is to "Allow mirror write-through", represented by an `EnableClientMirrorUpdate` value of `dword:1`.

There is one other behavior available, which out-of-the-box is not shown in SyncAdmin. And that is to "Permit write-through for mirror fetch state only". If `EnableClientMirrorUpdate` is set to a value of `dword:3`, when SyncAdmin is next invoked by the installation owner, the Site Options tab will show the "Permit write-through for mirror fetch state only" option, with that option selected. This enforces pre-DesignSync 4.0 default behavior, of `ci -mirror` being the only `ci` operation to write-through to a mirror directory.

This registry setting pertains to DesignSync client "write-through" behavior. A client "write-through" is a `ci` (in any mode, not just `-mirror`) causing the file version to be fetched into the mirror associated with the workspace. This ensures that the newly created data is immediately available to the site from which the new data was created.

To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$_SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file, and DesignSync clients restarted, for the changes to take effect. To restart a DesignSync client, exit and restart your DesSync, `dssc`, or `stcl` session. If you are using `dss` or `stcl`, use the

`syncdadmin` command to restart `syncd`. An alternative to restarting a DesignSync client is to use the `sregistry reset` command in the current DesignSync session, to reload the registry files.

Registry Setting

```
HKEY_CURRENT_USER\Software\Synchronicity\Client\General\Optimizations\EnableClientMirrorUpdate=dword:1
```

Allowed Values

Key value	Description
<code>EnableClientMirrorUpdate=dword:0</code>	Disallows mirror write-through.
<code>EnableClientMirrorUpdate=dword:1</code>	Allows mirror write-through.
<code>EnableClientMirrorUpdate=dword:3</code>	Allows mirror write-through for mirror fetch state only.

Related Topics

Check for matching selector (`ClientUseMirrorCheck`)

Symbolic link Handling (`ManagedLinkInMirrorMode`)

Site Options

Default Fetch State (`DefaultFetchType`)

Symbolic link Handling (`ManagedLinkInMirrorMode`)

The `ManagedLinkInMirrorMode` registry key is used to determine how to handle symbolic links to mirrored objects during checkout and populate operations.

By default, Relative symbolic links point directly to the referenced object resulting in those objects being fetched in the Copy fetched state. Absolute links point to the mirror, resulting in those objects being fetched in the Mirror fetch state. This is represented by a `ManagedLinkInMirrorMode` value of `dword:0`.

To set all managed symlinks to point directly to the referenced object, resulting in those objects being fetched in the Copy fetch state, set a `ManagedLinkInMirrorMode` property value of `dword:1`.

To set all managed symlinks to point to the mirror, resulting in those objects being fetched in the Mirror fetch state, set a `ManagedLinkInMirrorMode` property value of `dword:2`.

To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the client installation's `$SYNC_CUSTOM_DIR/site/config/SiteRegistry.reg` file, and DesignSync clients restarted, for the changes to take effect. To restart a DesignSync client, exit and restart your DesSync, `dssc`, or `stcl` session. If you are using `dss` or `stcl`, use the `syncdadmin` command to restart `syncd`. An alternative to restarting a DesignSync client is to use the `sregistry reset` command in the current DesignSync session, to reload the registry files.

Registry Setting

`HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Options\ManagedLinkInMirrorMode=dword:0`

Allowed Values

Key value	Description
<code>ManagedLinkInMirrorMode=dword:0</code>	Sets relative symbolic links to point directly to the referenced object. (Default)
<code>ManagedLinkInMirrorMode=dword:1</code>	Sets all managed symlinks to point directly to the referenced object.
<code>ManagedLinkInMirrorMode=dword:2</code>	Sets all managed symlinks to point to the mirror.

Related Topics

Check for matching selector (`ClientUseMirrorCheck`)

Whether write through occurs (`EnableClientMirrorUpdate`)

Site Options

Default Fetch State (`DefaultFetchType`)

Troubleshooting the Mirror System Registry Settings

Detect "Absent Files" (`FindAbsentFiles`)

This setting detects Absent Files in the mirror directories. "Absent Files" are DesignSync references that do not appear on disk. If there are Absent Files in a mirror directory, that indicates a problem.

If `FindAbsentFiles` is set to a value of `dword:1`, the MAS will detect Absent Files in mirrors. An e-mail message will be sent to users on the Notify List for the problem

mirror, listing the Absent Files. A message will be sent once every 24 hours, while Absent Files remain in the mirror

To correct the problem mirrors, force a full `populate` (via `-unifystate`) update of the mirror, by issuing a `mirror reset` in DesignSync, or `Mirror Reset` in DesignSync Web UI. That will replace DesignSync references with local file copies.

Note: Turning off absent file detection provides increased performance, however, it may have an impact on usability.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Options\FindAbsentFiles=dword:1
```

Allowed Values

Key value	Description
<code>FindAbsentFiles=dword:0</code>	Disallows checking for "Absent files" in mirrors.
<code>FindAbsentFiles=dword:1</code>	Allows checking for "Absent files" in mirrors. (Default)

Related Topic

Save all mup log files (`SaveLogFiles`)

Verifying that the Mirror System is Running (`HeartBeatInterval`)

This setting checks on the running status of the mirror. By increasing the periodicity the MAD is relieved of the burden of a more frequent check of the mirror run status.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\HeartBeatInterval=dword:120
```

Allowed Values

Key value	Description
HeartBeatInterval=dword:0	Disables heart beat checking for the mirror system.
HeartBeatInterval=dword:#	Amount of time, in minutes, between mirror system checks.
HeartBeatInterval=dword:120	Default interval of time for checking the mirror system. (Default)

Related Topic

Enable Automatic Requeueing (AutoRequeue)

Save all mup log files (SaveLogFiles)

Save all mup log files (SaveLogFiles)

This setting determines whether to save or remove logs for the mirrors system. Because the mirror system does not recognize the presence of Absent Files as a problem, the MAS's

`$_SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/muplogs/dss*.log` files are automatically cleaned up according to DesignSync's internal log file cleanup algorithm.

If `SaveLogFiles` is set to a value of `dword:1`, all of the `muplogs/dss*.log` files will be saved aside, for debugging.

If `SaveLogFiles` is set to a value of `dword:2`, the log from the mirror update is stored in the `<mirror_directory>/_SYNC/saved.dss.mirror.log`. Each mirror has its own log which is overwritten by the new mirror log, when the mirror is updated.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Options\SaveLogFiles=dword:2
```

Allowed Values

Key value	Description
SaveLogFiles=dword:0	Disallows saving of mup log files, files will be cleaned automatically.
SaveLogFiles=dword:1	Allows saving of mup log files in a centralized location.
SaveLogFiles=dword:2	Allows saving up mup logs on a per mirror basis in the mirror directory.

Related Topic

Detect "Absent Files" (FindAbsentFiles)

Enable Automatic Requeueing (AutoRequeue)

Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry)

Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry)

This setting determines whether the mirror system restarts transactions from the first failure or operates on the new transactions.

By default this will continue with the retry. Once set to 'dword:1' it will not do the retry.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\MAD\TurnOffAutomaticRetry=dword:0
```

Allowed Values

Key value	Description
TurnOffAutomaticRetry=dword:0	Starts retrying transactions beginning with the first transaction queued after the failed transaction, or, if the MAS has crashed, the first transaction queued after the restart.

	(Default)
TurnOffAutomaticRetry=dword:1	Does not attempt to retry transactions automatically. Any mirror system failure must be processed manually..

Related Topic

Detect "Absent Files" (FindAbsentFiles)

Enable Automatic Requeueing (AutoRequeue)

Turn Off Automatic Retry on Transaction Failure (TurnOffAutomaticRetry)

Scripted Mirrors Registry Settings**Script Name (Script)**

The setting specifies the name of the script to use when creating a scripted mirror. This can be overridden by specifying a different script when creating the mirror.

The default value is "generate_mirror.tcl" located in \$SYNC_DIR/share/tcl.

The registry setting below uses <m1> as an example mirror.

This registry setting pertains to the Mirror registry keys specific to scripted mirrors. To modify registry values programmatically, use the sregistry command set. The settings must be made to the server's

\$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\<m1>\Script="generate_mirror.tcl"
```

Allowed Values

Any specified value will be accepted if the format "<specified_value>.tcl" is entered.

Related Topics

Scrub Generated Mirrors (ScrubGeneratedMirrors)

Scrub Generated Mirrors (ScrubGeneratedMirrors)

This setting sets the recurring period of time, in hours, starting from the time that the MAS server is started when all auto-generated mirror definitions are deleted. The default is 24 hours (represented in hexadecimal as `dword:18`).

Note: The value must be at least 1.

This means an auto-generated mirror definition may be deleted soon after it was created, if its creation happened to be close to the upcoming `ScrubGeneratedMirrors` time event.

This registry setting pertains to the Mirror registry keys specific to scripted mirrors. To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's

`$SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Definitions\MAS\ScrubGeneratedMirrors=dword:18
```

Allowed Values

Key value	Description
<code>ScrubGeneratedMirrors=dword:1</code>	Sets time period for deletion of auto-generated mirror definitions to 1 hour. (Minimum Value)
<code>ScrubGeneratedMirrors=dword:18</code>	Sets time period for deletion of auto-generated mirror definitions to 24 hours. (Default)
<code>ScrubGeneratedMirrors=dword:<value></code>	Sets time period for deletion of auto-generated mirror definitions to specified value.

Related Topics

Script Name (Script)

Diagnostics and Troubleshooting

List of Error Messages

For ease of reference, this topic contains a list of all the published DesignSync SOM-E error messages. The list is contained in a table which contains:

- the SOM-E error.
- a brief description of the error.

The list is organized in numerical order, by SOM-E error number.

Som Errors			
SOM-E Error Number	Short Description	Explanation	Next Steps
01	Already Exists.	The directory already exists or the object already has been committed.	Verify that you have performed the operation on the correct object.
02	No Container.	The container folder for the object does not exist so the object cannot be committed.	
03	Cannot Be Committed.	The object cannot be programmatically committed.	Use an interactive creation method, such as Edit to modify the object.
04	Communication Failure.	The operation was unable to use a needed network communication. This indicates a TCP/IP error.	Verify that the user is connected to the VPN, if needed. Verify that DNS lookup is functioning and that all expected servers are visible on the network.
05	Communication Read Failure.	The operation requires network communication, which failed while reading the server's response.	Verify that the user is connected to the VPN, if needed. Verify that DNS lookup is functioning and that all expected servers are visible on the network.

06	Communication Write Failure.	The operation requires network communication, which failed while writing the request to the server. Failure occurred at the TCP/IP level.	This can be caused by an invalid "ProxyNamePort" environment variable setting or registry key. For more information, see Proxy Errors.
07	Communication Channel Busy.	The operation requires network communication, but the connection to the server is already in use.	
08	Communication Read Canceled.	The network response read portion of this function was cancelled by the user.	
09	Communication Write Canceled.	The network write (send request) portion of this function was cancelled by the user.	
10	Communication Dns Failure.	DNS Failed to locate the SyncServer.	
11	Communication Connect Failure.	Network communication failed attempting to connect to the SyncServer.	
12	Communication Wrong Content.	Network communication failed attempting to connect to the SyncServer.	
13	Communication Zero Content.	Network communication failed. The SyncServer returned a an empty reply.	This can happen as a result of checking in large files through a proxy server; for more information see Proxy Errors.
14	Sync Communication Failure.	Network communication failed to communicate with the SyncServer.	
15	Server Out Of Memory.	The server could not complete the operation because it ran out of memory	

DesignSync Data Manager Administrator's Guide

16	Server Version Not Supported.	This version of the client not supported by the server.	
17	No Such Protocol.	The protocol specified with URL of this object is not a known protocol.	
18	Failed On Metadata.	The operation failed because of errors in the metadata in the workspace. Possibly the object in the workspace is locked or opened in another application that is locking the file.	Close the application that has the object open, or delete the file lock and retry the operation.
19	Server Failed On Metadata.	The operation failed because of errors in the metadata on the SyncServer.	
20	Corrupt Metadata Log.	The metadata log in the workspace has become corrupt.	
21	Corrupt Metadata Log On Server.	The metadata log on the server has become corrupt.	
22	Required Dir Not Found.	One of the directories required to run SOM on the client system has not been created.	
23	Required Dir Not Found On Server.	One of the directories required to run SOM has not been created on the file system hosting the SyncServer.	
24	Server Not Licensed.	The network server is not licensed.	
25	No Permission.	The user does not have the permissions on the Operating System to perform the operation.	
26	Can't Write Output.	The user does not have the ability to write the output of	

		the operation to the specified location.	
27	Som Not Initialized.	The Initialize method of SOM has not been called successfully.	
28	Ambiguous Url.	Multiple objects match this URL. This can happen when the URL contains wildcards and the operation only supports a single match.	Refine the URL so that it matches a single object and perform the operation again.
30	Failed On Access Control.	The operation failed in trying to determine access rights.	
31	Can't Copy Across Server Boundary.	This object doesn't support a copy from one domain to another.	
32	Illegal Url.	The URL for this object is not in a legal format.	Adjust URL to the correct formatting, and attempt the operation again.
33	No disk space	The client system is out disk space and cannot complete the operation.	Clear enough disk space to complete the operation. For large data archives, such as Upload Archive, in addition to the space required for the archive, the temp space available must be approximately 2.5 times the size of the uploaded archive.
34	Server Out Of Disk.	The file system hosting the SyncServer is out of disk space.	If you are out of disk space on the server, you can add additional disk storage space or clear disk space from the server by removing temp files, checking the system for runaway processes that are consuming disk space, or purging obsolete data.

35	Cannot send a file to the client	Sending a file from the server has failed on the server.	
36	Cannot send a file to the server. Check that the file exists and has the correct Unix file permissions.	Sending file to the server has failed on the client.	Check that the file exists in the client workspace and that the file has the correct permissions for the user to perform the specified operation.
37	Cannot receive a file from the client. Check the disk space on the server, especially \$TMPDIR (/var/tmp by default)."	Receiving a file from the client has failed on the server.	If you are out of disk space on the server, you can add additional disk storage space or clear disk space from the server by removing temp files, checking the system for runaway processes that are consuming disk space, or purging obsolete data.
38	Cannot receive a file from the server	Receive a file from the server has failed on the client.	
39	Cannot send a response to the client	The attempt to send a response from the server has failed on the server.	
40	Cannot send a request to the server	The attempt to send a request to the server has failed on the client.	
41	Cannot receive a request from the client	The server's attempt to receive a request from the client has failed on the server.	
42	Cannot receive a response from the server	The attempt to receive a response from the server has failed on the client.	
43	Can't Move Across Domain Boundary.	The object doesn't support a move from one domain to another.	

44	Failed On Content.	The operation failed on the content.	
45	Illegal Delete Type.	The DeleteType parameter is not a legal delete type	
46	Not Allowed.	The operation is not allowed for the specified object.	
47	Not An Ancestor.	An attempt was made to remove an ancestor which was never added.	
48	No Such Property.	The Name parameter identifies the name of a property which has never been added to this object.	
49	Marking Not Supported.	This implementation does not support marking for deletion, but the DeleteType parameter was MarkedForDeletion.	
50	Cannot delete a branch-point version	This version cannot be deleted because it is the a branch point.	
51	Cannot delete a tagged version	This version cannot be deleted because it is tagged.	
52	Cannot delete first version on a branch	This version cannot be deleted because it is the first version on a branch.	
53	Cannot delete an upcoming version	This version cannot be deleted because it is a template version.	
54	Folder Not Empty.	Folder is not empty and cannot be deleted.	To delete the folder, first delete the contents of the folder.
55	Cannot delete vault as it has tagged revision(s) and/or locked branch(es)	If a files-based vault has tagged version or locked branches, it cannot be deleted.	

56	Cannot delete the Latest version on a locked branch.	This version cannot be deleted because its the Latest version on a locked branch	Unlock the object to delete the Latest version on the branch. Note you can never delete the initial branch point version on a branch, even if it is the Latest version and the branch is unlocked.
57	Unsupported Unit Type.	The specified type of unit is not support by this implementation.	
58	Option not supported by this vault type.	An option specified for a command isn't valid for the specified vault type.	
59	Invalid name for the vault.		
60	Cannot lock vault's file/directory.	The file or directory cannot be locked.	Check the file to see if the the file is already locked by another user or in a different workspace.
61	Vault already exists.	The stored object already exists.	
62	Not under revision control	The object is not under revision control.	Add the object to a module using the Add command or use the checkin command with the -new option to add files to source control.
63	No rights to access vault's file/directory	The operation does not have appropriate permissions to access the specified file or directory.	Verify the access controls for the specified object.
64	No rights to access the vault	The operation does not have appropriate permissions to access the specified vault.	Verify the access controls for the vault.
65	Cannot lock the vault	Vault cannot be locked.	Verify that the access controls are correct and that the vault is not already locked.

66	Vault is not locked	The operation requires the vault to be locked, but DesignSync is unable to lock the vault.	Verify that the access control permissions are correct and that the vault can be locked.
67	Internal error in the vault processing		
68	Vault is corrupted	Vault is corrupted.	Contact your system administrator or support.
69	Vault is not open	DesignSync was unable to open the vault to perform the operation.	
70	Vault is open read-only	DesignSync was unable to perform an edit operation on the read-only vault.	Verify that the vault is not locked to another user or in another workspace.
71	Cannot create the vault	DesignSync was unable to create the vault.	Verify that the user has write permissions for vault location.
72	Cannot open the vault	DesignSync was unable to open the vault to perform the operation.	
73	Cannot close the vault	DesignSync was unable to close the vault to complete the operation.	
74	Cannot delete the vault	DesignSync cannot delete this vault.	Verify the access control permissions are correct, or check to see if the vault is locked by another user.
75	Bad format for keys expansion	The keys in a file are not formatted correctly.	For more information on keyword expansion, see <i>ENOVIA Synchronicity DesignSync User's Guide Using Revision Control Keywords</i> .
76	Invalid name of the working file	The file name has changed from the expected source controlled file name.	
77	Working file already exists		

78	Working file doesn't exist		
79	No rights to access working file	The server does not believe you have access to the file being processed.	Verify that you are correctly logged in or refresh the client before attempting the operation again.
80	Invalid version name		
81	Version already exists	The version already exists on the server.	If you have made local modifications, perform a populate with the merge option in your workspace to review and resolve any merge conflicts, then retry the checkin operation or perform a populate with the force option to overwrite local modifications. If you have not made changes, populate to refresh the data in your workspace.
82	Cannot delete a branch-point version	The first version on a branch can never be deleted.	For files-based vaults branches can be retired to prevent further use. For modules, if you do not wish further checkins on a branch, you can disable access to the branch, but you cannot remove module branches.
83	Version is already locked.	The lock operation cannot be performed because the object is already locked.	If you are not the lock owner, you can see who owns the lock by viewing the properties of the object, or using the showlocks command.
84	Version is not locked	The cancel checkout or unlock operation has failed because the object was not locked in the workspace.	

85	Cannot create a template version		
86	Version doesn't exist		
87	Version is not a template		
88	Version doesn't exist		
89	Invalid tag name	The specified tag name does not exist or is not a valid tag name.	Select a different name for the tag. For a list of legal characters, see ENOVIA DesignSync User's Guide: URL Syntax.
90	Tag already exists	The specified tag already exists.	Use a different tag name or, if you are applying the tag with the tag command, specify the Replace option. For more information, see Tagging Versions and Branches.
91	Tag doesn't exist	The specified tag does not exist.	Verify that you specified the correct tag name.
92	Selector resolves to version (branch expected)	The selector specified resolves to a version ID, when a branch ID is expected for the operation.	Specify the branch to operate on.
93	Bad argument to the vault member function		
94	Cannot move one of the vault's files	The server is unable to move the vault object.	Verify that you have the correct access permissions for this operation.
95	Cannot copy one of the vault's files	The server cannot copy the vault object.	Verify that you have the correct access permissions for this operation.
96	Error initializing vaults engine		

97	Object Not Ready.	The versionable object has another pending request. The operation in process must either complete or be instructed to clean up, based upon its current state.	
101	Already Locked.	Either the version is already locked or, for storage managers which do not support multi-locks, the only lock on the stored object is already taken.	
102	Locked By Other User.	The object you are attempting to check into is already locked by another user.	
103	Attempted to checkin either an unknown version, or a version which does not match the version currently checked out.	Since the last check out to this versionable object, the object was checked in from a different versionable object. At the moment, another lock has been placed on the current latest version on the branch.	
104	Merge Conflicts.	The object being checked in conflicts with the latest version on the server.	Merge the object to your workspace to review and resolve the merge conflicts, then retry the checkin operation.
105	Cannot skip versions on a branch without the '-skip' option	The version being checked in is not the latest version on the branch.	The user must specify the -skip option (or Allow Version Skipping) for checkin or merge the latest version into the workspace, resolve any conflicts, and retry the checkin operation.
107	Vault structure doesn't correspond to the working object.	When checking in an object, the object parameter points to a object that is incompatible with the vault.	

108	Bad Versionable Path.	The path to the target object could not be found.	
109	Cannot open the working file.	The target object could not be opened by the storage manager.	
110	Bad Revision Name.	The storage manager could not work with the revision number supplied by the versionable object.	
111	Not Checked Out.	The versionable object did not arrive in its current folder from a DesignSync checkout operation. The VersionRetrieved is unknown, and therefore there is no basis for determining where its history belongs in the vault.	
112	Failed On Dependency.	One or more of the objects which need to be checked in before this object failed to check in.	
113	Unable to find mirror directory. Please check that it is set correctly.	The required mirror directory was not found to perform the operation.	
114	Has Conflicts.	The file has conflicts which prevent it from being checked in.	
115	Setvault Not Done.	No vault has been specified using setvault.	
116	Failure While Opening Contents Metadata For Write	Failed writing the metadata.	
117	Failed To Open Contents Metadata For Read	Failed reading the metadata.	

118	Failed To Obtain Write Lock On Contents Metadata	Failed to obtain a write lock for the metadata.	
119	Failed To Obtain Read Lock On Contents Metadata	Failed to obtain read lock for the metadata	
120	Not Metadata Owner	Failed to operate on metadata because the user is not the owner of the object and does not have the appropriate permission.	
121	Metadata No Permission.	Failed to operate on metadata because the user does not have the appropriate permission.	For more information, see Troubleshooting Metadata Errors.
122	Cannot Create Metadata for Root of a File System	The system attempted to create metadata at the file system root. This is an illegal operation. The metadata must be in a .sync folder that is not set at the file system root.	
123	Timed out waiting to acquire metadata read lock.	The attempt to obtain a read lock failed because the metadata was busy and timed out.	
124	Timed out waiting to acquire metadata write lock.	The attempt to obtain a write lock failed because the metadata was busy and timed out.	
125	Timed out waiting to acquire metadata lock.	The attempt to obtain a lock failed because the metadata was busy and timed out.	
126	Interrupted while waiting to acquire metadata lock.	The attempt to obtain a lock was interrupted.	

129	Locally Modified.	This object has local modifications that would be erased if the operation continued.	To preserve the data, perform a populate with the merge option or perform a populate with the force option to overwrite local modifications.
130	Can't Write To Versionable.	The object in the workspace could not be updated during checkout.	
131	Revision Corrupt.	The version being checked out has an internal issue.	
132	New Rev Exists.	When using the "New Rev on Checkout" revision control mode, the new version identified as the next checkout version already exists. This occurs when someone else has checked out the same object with the same mode, but has not yet checked in the object.	
133	Storage Manager Failure.	The storage manager failed to complete the check in.	
134	Merge is not Supported for Binary Files and Collection Objects.	DesignSync recognizes the file as a binary file or collection object. Merge is unavailable for these object types.	
135	Can't overwrite local changes. Use force switch.	This object has local modifications that would be erased if the operation continued.	To preserve the data, perform a populate with the merge option or perform a populate with the force option to overwrite local modifications.
137	Merge Not Required.	The merge option was specified, but no merge is needed.	

139	Merge Conflicts No Lock.	The merge operation was successful, but there are merge conflicts in the object and DesignSync was unable to lock the object.	Review the object to accept/reject the merge conflicts before checking in the object. If the object is checked out by another user, you may need to merge changes again, or coordinate with the other user to release the file.
141	Merge No Lock.	The merge was successful, but DesignSync was unable to lock the object.	
142	Req Mirror File Not Found	The specified mirror was not found. Verify that the mirror host is operational and that there are no network issues.	
143	Null Web Object.	The Item parameter is NULL.	
144	Illegal Name for a item in this folder	The supplied name parameter is not a legal name for a item in this folder.	Specify a valid name and retry the operation. For a list of legal characters, see ENOVIA DesignSync User's Guide: URL Syntax.
145	Cannot Place a Temporary Copy of a File in the Cache.	DesignSync is unable to write a temporary file into the file cache.	Verify that there is enough temp space to write to the file cache and that permissions on the file cache are set correctly.
146	Failed At Source.	The object which currently controls the file failed to produce it.	
147	Cannot Unensure.	The DesignSync file caching system was unable to remove a temporary file from the cache.	
148	Cannot Make Symlink.	DesignSync was unable to create a symbolic link to an object in the file cache.	

149	Cannot Unlink Symlink.	DesignSync was unable to remove a symbolic link to an object in the file cache.	
150	Item Already Exists.	There is already an item in source control with the same relative path.	
151	There is no object with that name	There is no item with the specified relative path.	
152	No Such Version.	No version matches the specified fetch preferences.	
156	Not Locked.	The version is not currently checked out (locked).	
158	Name Taken.	The specified name is already a revision name or symbolic name.	
159	No Such Name.	The specified name is not an existing symbolic name.	
160	Insufficient Property Info.	The Props parameter did not provide a set of properties sufficient to define an object.	
161	Wrong Protocol.	The container object was not using an expected protocol.	
162	No License.	The specified license is missing, incorrect, or all licenses are in use.	Specify a different license using the SyncServer setup, close unused clients, or contact support other options.
163	Bad License.	The user has an illegal license assigned.	Specify a different license using the SyncServer setup or contact support for other options.
164	Can't Access Storage Manager.	The storage manager subsystem failed to initialize.	Restart the SyncServer. If this does not fix the problem, contact support.
165	Cannot Access Event Context.	The event context failed to initialize, making event	

		triggers inoperable.	
166	The AccessControl system failed to initialize	The Access Control system did not start up correctly.	Restart the SyncServer. If this does not fix the problem, contact support.
167	Channel Not In Use.	The channel was not in use and therefore didn't need to be cancelled.	
168	Cannot Create Parent Directory.	DesignSync was unable to create the parent directory.	Verify access permissions for the directory path and that there is sufficient disk space to perform the operation.
169	Cannot Create Directory.	DesignSync was unable to create the target directory.	Verify access permissions for the directory path and that there is sufficient disk space to perform the operation.
170	No Parent Directory.	The path to the specified object does not exist.	
171	Path contains illegal character(s) like ;	The specified dir path has illegal/reserved characters in it.	Rename the files or folders on the path that have illegal characters. For a list of legal characters, see ENOVIA DesignSync User's Guide: URL Syntax.
172	Null Parameter.	The object parameter is NULL.	
173	Can't Open Cache Directory.	The file cache directory cannot be opened or created	
174	Can't Make Cache Lock.	The file cache lock cannot be opened or created.	
175	Cache Lock Status.	The DesignSync file cache status cannot be accessed or modified.	

176	Can't Lock Cache.	The DesignSync file cache cannot lock the cache directory.	
177	Can't Unlock Cache.	The DesignSync file cache cannot unlock cache directory.	
178	Cannot Create Link to Cache. Check the Unix permissions on the cache directory and its files.	The DesignSync file cache cannot create a new hardlink to the cached file.	
179	Can't Unlink Tmpfile.	The temp file is a symbolic link, which cannot be unlinked.	
180	Can't Unlink Hardlink.	DesignSync could not unlink a hard link to the cached object file	
181	Can't Touch Hardlink.	DesignSync could not unlink a hard link to the cached object file.	
182	Can't Make Tmpfile.	The temp file could not be created.	Verify that you have access permissions to write to the temp space and that there is enough disk space to create the temporary object.
184	Object Not Compatable.		
185	Vault URL assignment would cause recursive working dir loop.	The Valut URL property was set to a URL above the versionable object folder.	
186	No Specified Vault.	The DefaultVaultURL property is not set.	
187	No Such Vault.	The specified vault URL is not a valid vault folder.	
188	Can't Create Sub Folder.	The populate was called recursively, but could not	

		create a sub folder to match a sub folder of the vault.	
189	Can't Open/Create Mirror Directory.	The mirror directory cannot be opened or created.	Verify that the mirror owner has write access to the server and that the server is accessible on the network.
190	Not Allowed. An ancestor directory already has mirror set.	A mirror directory is already defined in an ancestor.	
191	Not Allowed. A descendant directory already has mirror set.	A mirror directory is already defined in a descendant.	
192	Didnt Help.	This attempt to recover did not succeed.	
193	Object is a malformed collection.	The collection was not created correctly, and cannot be checked in.	
195	Could not write to the mirror directory. Please check read/write access on the mirror directory.	The required mirror directory was not writable or created.	Verify that the mirror owner has write access to the server and that the server is accessible on the network.
196	Invalid branch name	The specified branch does not exist.	
197	Branch doesn't exist	The specified branch does not exist.	
198	A fromVaultFilter trigger returned an error status.	An operation that called the fromVaultFilter trigger returned an error.	For more information on events that use the fromVaultFilter trigger, see Events Overview.
199	Permission denied by the AccessControl system.	The operation was denied by the Access Control system.	Verify that you have correct access permissions and have complied with minimum comment length

			requirements.
200	Server Access Denied.		
201	Server Authentication Req'd.	The server doesn't recognize your client. It may be because server had previously been communicating with this user from another IP address, or with other client computer.	
202	No Server Script Specified.	A server side script to execute was not provided.	
203	Server Script Does Not Exist.	Could not find such a script on the server.	
204	Error In Executing Server Script.	The Server side script execution resulted in an error.	
205	Unable to open output file for write.	Unable to open the designated output file.	
207	Branch Is Retired.	The branch is retired so you cannot perform this operation.	
210	Mirror is only supported on Branch 1.	Attempted to get a mirror link to a file other than on first branch..	
211	Move Not Supported On This Object	The move operation is not supported for this object	
212	Parent Folder Of Destination Does Not Exist	The destination parent folder does not exist and cannot be created.	
213	Destination File Already Exists, Cannot Move	The file cannot move to the destination, because it already exists at that destination.	

214	Destination Folder Already Exists, Cannot Move	The folder cannot move to the destination, because it already exists at that destination.	
215	Error While Moving File To Destination	The file cannot be moved to the destination, because of a unidentified error during a move file operation.	
216	Error While Moving Folder To Destination	The folder could not be moved to the destination, because the system was unable to create the destination directory. Verify the user has write access.	
217	Destination Vault Already Exists	The vault cannot move to the destination, because it already exists at that destination.	
218	Destination Vault Folder Already Exists	The vault folder cannot move to the destination, because it already exists at that destination.	
219	License Manager, Unknown Agent.	The license manager did not recognize the agent performing the operation.	
220	License Manager, failure in attempting to obtain license.	The license manager was unable to acquire a valid license. Try the operation again. If it continues to fail, make sure that you have available licenses or contact your system administrator.	
221	License Manager, license not available.	There are no available licenses for this feature.	Verify that you are licensed for this feature and try the operation again. If it continues to fail, make sure that you have available licenses or contact your system administrator.

222	License Manager, Cadence Object Processing license not available.	License unavailable for Cadence Object Processing.	Verify that you are licensed for this feature and try the operation again. If it continues to fail, make sure that you have available licenses or contact your system administrator.
223	Null Parameters.	The value passed in web object pointer is NULL.	
224	Duplicate Init.		

DesignSync Client Troubleshooting

Diagnostics Overview

The DesignSync graphical client (DesSync/DesignSync GUI) provides a set of diagnostic tools for troubleshooting. Within the interface, **Tools=>Diagnostics** brings up the **DesignSync Diagnostics** pane. Three tabs are available:

- **Environment Variables** — Displays DesignSync-related environment variables and their current settings. The SyncAdmin tool also has an Environment Variables tab that displays the same list of environment variables.
- **Installation Settings** — Displays settings that were selected during the DesignSync installation or were later overridden using the SyncAdmin tool.
- **Java Properties** — Displays the current Java classes, paths, and versions.

The **DesignSync Diagnostics** dialog box is useful for debugging your DesignSync environment. For example, a DesignSync representative may ask you for information that is available from this dialog box when assisting you with a problem.

All of the information displayed is read-only -- you cannot edit the values from the **DesignSync Diagnostics** dialog box.

Note:The Java Properties and Environment Variables diagnostic information is also available from the Diagnostics menu in SyncAdmin.

Related Topics

syncinfo Command

Using Environment Variables

About DesignSync Client Log Files

When you run DesignSync, `dssc`, `stcl`, `dss`, or `stcl`, these log files are created:

- `dss_*.log` file
- `sync_client_trace_*.log`

Note: Separately, you may also log the output of a `populate` command. For more information, see [Populate Log](#).

`dss_*.log` file

By default, for every DesignSync session you run, a log file is placed in your home directory. The default log filename is `dss_<date>_<time>.log`. For example, the log file `dss_01102002_182640.log` was created on January 10, 2002 at 18:26:40 hours. Because the log file is time-stamped, it is not overwritten and log files are retained from every session of DesignSync that you run.

The log file has a default maximum size of 10 MB. If the maximum size is reached, the logged information is moved to `dss_<date>_<time>.bak.log`, the existing log file is zeroed out, and client session logging continues.

This process repeats every time the session log limit is reached. DesignSync only maintains one backup file, which essentially has the same maximum size as the log file. This means that if the value is set too low, or the logging level is too high (for example `synctrace` is on) you may not see a complete history. For more information on customizing maximum log file size, see [DesignSync client log size \(MaxClientLogSize\)](#).

Logging options can be set for an installation or a project, or for an individual user, using SyncAdmin's Logging pane. The logging options let you specify:

- Whether logging should begin at startup
- Whether to log both commands and their results
- The default location of log files
- Whether log file names should be unique, or whether a specific log file name should be reused

When using a unique log file name for each client session, all `dss_<date>_<time>.log` files older than two days are automatically removed when a DesignSync client starts. If there are more than 20 log files less than two days old, all log files older than one hour are removed.

Trace messages are also logged to this file when tracing is enabled. You can enable tracing by setting the `SYNC_TRACE` environment variable to 0 for debugging purposes.

You can also enable tracing using the **synctrace set** command, which takes effect immediately and is in effect until you stop tracing with **synctrace unset** or when the client session exits. The trace messages that are logged to this file are for the most part not present in the `sync_client_trace_*.log` file.

Note: For logging changes to take effect, you must restart any DesignSync clients that are running, including `syncd`, which manages `dss` and `stcl` communication. You can restart `syncd` using the `syncdadmin` tool. See ENOVIA Synchronicity Command Reference: `syncdadmin` for more information.

You can also use the **log command** to control the logging of DesignSync commands and command output.

When you restart a DesignSync client, and you have set `SYNC_TRACE`, you will see messages output from the initialization process. The `dss_*.log` file begins to capture information when the initialization process is complete.

Note: When sending client trace log files to ENOVIA support, provide both the `dss_*.log` files and the `sync_client_trace_*.log` files for the debug session.

sync_client_trace_*.log

If you set the `SYNC_TRACE` environment variable to 0 for debugging purposes, certain trace messages are logged to `sync_client_trace_<date>_<time>.log`. These trace messages are not always present in the `dss_*.log` file. You can also enable tracing using the **synctrace set** command. This log file is stored in `$$SYNC_USER_CFGDIR/logs`.

Note: The default value of the `$$SYNC_USER_CFGDIR` is `$HOME/.synchronicity` on UNIX. On Windows, the default value is `%AppData%\Synchronicity`. This value is implicitly defined. To change it, create the `SYNC_USER_CFGDIR` environment variable and set the value as desired.

All `sync_client_trace_*.log` files older than two days are automatically removed when a DesignSync client starts. If there are more than 20 log files less than two days old, all log files older than two hours are removed.

`sync_client_trace_*.log` files are created for each DesignSync client. When you set the `SYNC_TRACE` environment variable, you must restart any application you are debugging for the tracing to take effect. If you use the **synctrace set** command, tracing is started and enabled only for this client session or until tracing is stopped with the **synctrace unset** command.

No trace log files are created for `dss` or `stcl` because these applications communicate to SyncServers through the client `syncd` session, and the `syncd` creates its own log file.

Note: You can restart syncd using the syncdadmin tool. See ENOVIA Synchronicity Command Reference: syncdadmin for more information.

When you restart a DesignSync client, and you have set SYNC_TRACE, you will see messages output from the initialization process. The sync_client_trace*.log file begins to capture information when the initialization is complete.

Note: When sending client trace log files to ENOVIA support, provide both the dss_*.log files and the sync_client_trace_*.log files for the debug session.

Related Topics

Running a DesignSync Client in Debug Mode

ENOVIA Synchronicity Command Reference: log command

ENOVIA Synchronicity Command Reference: synctrace set

ENOVIA Synchronicity Command Reference: synctrace unset

Using Environment Variables

Command Line Clients Fail to Start

Problem: DesignSync command line clients (stcl, stclc, dss, dssc) fail to start.

Possible Cause: The \$SYNC_HOSTNAME and/or \$SYNC_DOMAINNAME environment variables were unable to be automatically set by the DesignSync .syncinc script.

Solution: On complicated systems, it may be difficult for DesignSync to automatically parse and create the \$SYNC_HOSTNAME and \$SYNC_DOMAINNAME environment variables. In these cases, you can manually set these environment variables in the .cshrc, .profile, .alias file, or any other sourced file used by your users when they log in to UNIX systems and use DesignSync.

The \$SYNC_HOSTNAME should be set to the IP address or hostname of the server.

The \$SYNC_DOMAINNAME should be set to the domain name of the server.

Example: This example user UNIX commands in .csh to set the values for the current session.

```
> setenv SYNC_HOSTNAME `hostname`
```

```
> setenv SYNC_DOMAINNAME `domainname`
```

```
>setenv
...
SYNC_HOSTNAME=serv1
SYNC_DOMAINNAME=ABCo.com
```

Could Not Locate Module

Problem: During a revision control operation, DesignSync issues the following error messages:

```
Error: Could not locate module <ModuleInstance>
```

Possible Cause: The workspace root directory is set in the wrong location. This can happen when a module workspace is in place and a setvault operation is run on an upper level folder.

Solution: Clear the workspace root directory and repopulate the module, or perform a setroot at the correct location.

1. Determine the workspace root location. This involves using two different url root commands, the first to determine the proper module root directory location and the second to determine where the incorrect module root is defined.

```
dss> url root <module_base_directory>
<Returned_directory>
dss> url root <Returned_Directory>
<Incorrect_Root>
```

For example:

```
dss> url root c:\workspaces\chip_design\Nz17
c:\workspaces\chip_design
dss> url root c:\workspaces\chip_design
c:\workspaces
```

2. Unset the workspace root.

```
dss> setroot -unset <rIncorrect_Root>
```

For example:

```
dss> setroot -unset c:\workspaces
```

3. Verify that the following registry keys exist and are set appropriately in the \$SYNC_CUSTOM_DIR/site/config registry file:

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Commands\General\MetadataCheckRelocation=dword:0
```

```
HKEY_CURRENT_USER\Software\Synchronicity\General\Commands\General\EnsureRoot=dword:0
```

4. Reset the module root by repopulating the workspace:

```
dss> populate <moduleInstance>
```

For example:

```
dss> populate Nz17%0
```

See Also

Enable the setvault command to set the module root directory (EnsureRoot)

[ENOVIA Synchronicity Command Reference: setroot](#)

[ENOVIA Synchronicity Command Reference: url root](#)

[Auto-creation of workspace root directory \(AllowAutoRootCreation\)](#)

[Workspace root path \(DefaultAutoRootPath\)](#)

Error Accessing or Storing to Database

Problem: During a revision control operation, DesignSync issues one of the following error messages:

```
Error accessing database
```

```
Error storing to database
```

Possible Cause: The tags database might have become corrupted.

Solution: DesignSync performs regular checkpointing of the tags database, so that if the database appears to be corrupted, the database can be restored. Upon restarting, the SyncServer automatically recovers the database by restoring from the last checkpoint, then replaying the journal. To restore the database:

1. Shut down the SyncServer by running `stop_sync_server` on UNIX.
2. Restart the SyncServer by running `start_sync_server` on UNIX.

DesignSync automatically recovers the database.

By default, checkpointing of the tags database occurs daily at 1:00 AM local time. DesignSync tool administrators can update the `DBCheckpointTime` registry setting to fine-tune when checkpointing is to occur. See [Advanced Registry Settings](#) for details.

Files Missing from the Cache

Problem: A cache file that is still being referenced was unintentionally removed after running `cachescrubber`.

Possible Cause: You may have removed the file when:

- `cachescrubber` was run with the `'- noref'` or `'-preds32'` options, or
- all workspaces were not supplied when `cachetouchlinks` was run.

Solution: When a cache file that is being referenced is removed, you can use the `refreshcache` command to recreate links to the cache from any workspace. The recreation of the links will also re-fetch the version into the cache if it is not already present.

Examples:

```
stcl> refreshcache -workspace /home/Projects/ ASIC
```

or

```
stcl> refreshcache -file /home/Projects/ workspace_list_file
```

where `workspace_list_file` contains a list of workspaces.

Related Topics

[refreshcache command](#)

[cachetouchlinks command](#)

[cachescrubber command](#)

[How DesignSync Manages Caches](#)

[Moving a Cache](#)

Files are Missing from My Module Workspace

Problem: You have applied or reapplied a module view, or performed a full populate on your module workspace and now files are missing.

Possible Cause: The files were created and added to the module from your workspace, but they do not match the module view definition.

When files are created in your workspace, they persist, even if they do not match the workspace view definition. When the view is applied (or reapplied) to the workspace, or if a full populate is done (Unify Workspace Space), any files that do not match the definition in the view are removed automatically.

Solution: Update the view to include the files you have created in the workspace.

Related Topics

Overview of Module Views

Creating Module View Definitions

Running a DesignSync Client in Debug Mode

There are several approaches to debugging a client:

- Restarting the client with debug environment variables set
- Enabling debug output within a DesignSync client session
- Using `syncdiag trace` for extensive OS-level debugging

Restarting the client with debug environment variables set

The simplest debug approach is to restart the client, with a debug environment variable set:

```
% setenv SYNC_TRACE 0
```

Then start a DesignSync client.

`setenv SYNC_TRACE 0` starts the client with DesignSync tracing. DesignSync tracing captures debug information pertaining to DesignSync software. The output is logged to the user's `$HOME/dss*.log` file and to their `$HOME/.synchronicity/logs/sync_client_trace*.log` file.

To disable DesignSync tracing, restart the client without the debug environment variable set:

```
% unsetenv SYNC_TRACE
```

Then start a DesignSync client.

Enabling debug output within a DesignSync client session

Use the **synctrace set** command to enable DesignSync-level tracing within a DesignSync session, so that you do not have to restart the client. Then, use the **synctrace unset** command later to disable DesignSync tracing.

Using syncdiag trace for extensive OS-level debugging

Use **syncdiag trace** to run a DesignSync command with DesignSync tracing, with Operating System (OS) level tracing, or with both levels of tracing. DesignSync tracing captures debug information pertaining to ENOVIA Synchronicity DesignSync Data Manager software, as if the command **synctrace set** has been run. OS-level tracing captures debug information pertaining to the Operating System. DesignSync tracing, and OS-level tracing, are only enabled for the duration of the DesignSync command run by `syncdiag trace`.

For information on the current `syncdiag trace` options, in your OS shell, run:

```
% syncdiag -help trace
```

Available options include:

<code>-sync</code>	Run with DesignSync tracing (SYNC_TRACE)
<code>-os</code>	Run with OS level tracing
<code>-all</code>	Run with both OS and DesignSync tracing

The default behavior is **-all**.

For example:

```
% syncdiag trace -sync dssc ci -share -comment "Test" testfile
```

The above command creates a `syncdiag.tgz` package containing the debug log files.

Notes:

- If you are using an SUID installation, you need to run the `syncdiag trace` command as the installation owner.
- The OS-level tracing program used is platform dependent. For HP, your system administrator might need to download "tusc".

Related Topics

Overview of syncdiag Utility

About DesignSync Client Log Files

ENOVIA Synchronicity Command Reference: synctrace set

ENOVIA Synchronicity Command Reference: synctrace unset

Interrupt Button Does Not Seem to Work

Problem: During a DesignSync operation (for example, a checkin) clicking **Stop** or choosing **Tools => Interrupt** doesn't seem to have any effect.

Possible Cause: When a command is run on multiple files or directories, the command operates on multiple files or directories at once, as opposed to processing one file at a time, one directory at a time. When a command is interrupted, before the command stops it will complete its processing of the current files or directories being operated on.

Solution: Because the operation does not stop immediately, it may seem as if the interrupt is being ignored. However, DesignSync is working correctly and should complete the operation or display the message that the command was interrupted.

Performance Issues with DesignSync GUI Client

Problem

The DesignSync GUI client (DesSync), runs slowly on Windows systems.

Possible Cause

DesignSync is attempting to access drives that have been removed or are not accessible across the network.

Solution

Modify the `AllowableDriveTypes` registry key to restrict the client to navigating locations that are always available. For example, fixed media only drives or RAM storage only.

For information on modifying the registry key, see Available drive types (`AllowableDriveTypes`).

Starting a New Browser Process Each Time Help Is Invoked

Note: This topic applies to web browsers on the UNIX platform only.

Problem: Each time help is invoked (by clicking Help, pressing the F1 key, or selecting **Help => Contents and Index**), your web browser starts a new process, which displays a new browser window.

Possible Cause: Each help invocation sends a browser request that causes the browser to launch a new process.

Action: Specify the `nsremote` script as your HTML browser. This script is located in the `$SYNC_DIR/bin` directory. The `nsremote` script calls the specified web browser with the `-remote` option so that browser requests connect to an active browser process if one exists instead of always starting a new process.

To specify `nsremote` as your HTML browser, use the SyncAdmin tool. See General Tab for more information.

Tagging Does Not Warn of Removed Objects

Problem: The tagging operation apparently succeeds even though some of the objects have been removed from the SyncServer vault.

Possible Cause: Either of these operations might cause this problem:

- A UNIX `rm` command was used. (**Note:** `rm` is not recommended; use `rmvault` instead.)
- The `rmvault -nokeepvid` command was used, then the object was checked in again with `ci -new`. (**Note:** The `-nokeepvid` option is not recommended; use the default option, `-keepvid`.)

Solution: Use `tag -warn` exists to ensure that you are notified if objects you are attempting to tag do not exist in the vault.

Troubleshooting Metadata Errors

Client operation stalls, "Waiting to obtain metadata lock"

Problem: You kill a DesignSync client, then restart the DesignSync client in the same workspace. Attempting a DesignSync operation in that workspace stalls, outputting:

```
WARNING - Waiting to obtain metadata lock for
/home/tbarbg7/SD31447testData/
```

```
Already locked by: user (tbarbg7) on host (poulenc), process id
(28730)
```

Possible Cause:

In the above example, the user "tbarbg7" started the initial stclc session, which was killed. "tbarbg7" then restarted stclc from the same workspace in which it was killed.

Because stclc was killed, its lock on the workspace metadata was not released. This was indicated by the lock still being held by the user. When stclc restarts, if the lock file (.SYNC/ Contents.sync.lck) is more than 5 minutes old the lock file is presumed to be orphaned. At that time, the lock file is removed, allowing operations to proceed.

Solutions:

- An alternative to waiting for the 30 second time period to elapse is to remove the .SYNC/ Contents.sync.lck lock file, if you are certain the process that obtained the lock file (shown in the output above) has exited.
- Change the orphan timeout value from 5 minutes to a lower value, so that DesignSync clients wait less time before automatically removing the lock file. See Workspace Metadata Registry Settings for how to modify the orphan timeout value.

Setvault Command Returns Error: Metadata No Permission

Problem: You enter the **setvault** command and DesignSync returns the error "Metadata No Permission." For example:

```
[somlib-16] Failed on metadata in /home (Open failed; check permissions)
```

```
file:///u1/userA: som-E-121: Metadata No Permission.
```

Possible Causes:

- Your umask permissions are not set to allow write access. DesignSync needs to be able to write into the parent directory of the work area directory where you run the **setvault** command. For an explanation of this concept, see *DesignSync Data Manager User's Guide: Local Metadata* and the .SYNC Directories and *DesignSync Data Manager User's Guide: The Role of setvault in Moving a Workspace*.
- There is insufficient disk space to write to the local metadata file.

Solutions:

- Check UNIX permissions and, if necessary, set them so that you have write access to the parent directory of the work area directory.
- Make there is enough space on the disk in which the local metadata file resides.

RevisionControl Operation Returns Error: Not Metadata Owner

Problem: You perform a revision control operation and DesignSync returns the error:

"Not metadata owner." For example:

```
Checking out: samp.asm : [somlib-16] Failed on metadata in
/u1/userA/Sportster/code (Not Owner)
```

```
Failed:som: Error 120: Not Metadata Owner
```

Possible Cause:

Your umask permissions are not set to allow write access. DesignSync needs to be able to write into the parent directory of the work area directory where the setvault is being formed.

Solution: Change the umask permissions to allow write access.

For example, if user1 has a umask set to 022 (which turns off write permission for group and world), then DesignSync creates the `.SYNC/Contents` files with permissions 644. If user2 attempts to operate on files in user1's managed work area, user2 does not have UNIX write access to user1's `.SYNC/Contents` file.

Related Topics

DesignSync Data Manager User's Guide: Metadata Overview

DesignSync Data Manager User's Guide:Setting Up a Shared Workspace

DesignSync Data Manager User's Guide:Relocating a Workspace

Attempts to acquire metadata lock

Amount of time to wait for metadata lock

Orphaned lock timeout

Troubleshooting nsremote Issues

This topic discusses troubleshooting trips for the default browser definition for DesignSync.

Problem: `nsremote` fails when the browser is not running.

Possible Cause: You are using `nsremote` with a wrapper script, and the wrapper script fails to return the status from the underlying call to the browser.

Solution: `nsremote` uses the status value returned from the wrapper script to determine if a connection to an existing browser session is possible.

An example of a script you might use to return the status of the call to the browser is:

```
#!/bin/sh
FIREFOX_HOME=/usr/local/firefox1508
export FIREFOX_HOME
exec $FIREFOX_HOME/firefox ${1+"$@"}
```

SyncServer Troubleshooting

About SyncServer Log Files

Running a SyncServer creates these log files:

- `sync_server_trace.log`
- `SyncServer.log`
- `error_log`
- `access_log`

`sync_server_trace.log`

Server initialization messages are captured in the log file:

```
<syncdata>/<host>/<port>/server_vault/sync_server_trace.log
```

Trace messages are also logged to this file when tracing is enabled. You can enable tracing by setting the `SYNC_TRACE` environment variable to 0 for debugging purposes, before starting the server. You can also enable tracing using the **`synctrace set`** command, which takes effect immediately and is in effect until tracing is stopped with **`synctrace unset`**.

This file is overwritten whenever the server is started.

Additional trace messages are captured in the SyncServer's `error_log` file.

`SyncServer.log`

Brief error messages are logged in the `SyncServer.log` file:

```
$SYNC_CUSTOM_DIR/servers/<host>/<port>/server_metadata/SyncServer.log
```

Note: `SYNC_CUSTOM_DIR` defaults to `$SYNC_DIR/custom`.

The complete failure context is written to the `error_log` file.

error_log

The server's `error_log` file captures all server-related processes. The `error_log` is located in the directory:

```
$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
```

Any errors generated by the server are recorded in the `error_log` file.

Note: `SYNC_CUSTOM_DIR` defaults to `$SYNC_DIR/custom`.

Trace messages are also logged to this file when tracing is enabled. You can enable tracing by setting the `SYNC_TRACE` environment variable to 0 for debugging purposes, before starting the server. You can also enable tracing using the **`synctrace set`** command, which takes effect immediately and is in effect until tracing is stopped with **`synctrace unset`**.

access_log

The server's `access_log` file contains information about client requests to the server. The `access_log` is located in the directory:

```
$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
```

Note: `SYNC_CUSTOM_DIR` defaults to `$SYNC_DIR/custom`.

The `access_log` file increases in size depending on the amount of client interaction you have at your site. If you find the `access_log` file grows too large, you can disable its creation.

To disable logging to the `access_log` file:

- In the server's `$SYNC_CUSTOM_DIR/servers/<host>/<port>/conf/httpd.conf` file, comment out the following line:

```
TransferLog logs/access_log
```

You can limit the `access_log` file to a certain size by using a nightly cron job. The following script moves both the `access_log` and `error_log` files to `*.old` files when the `access_log` file reaches 100 blocks.

```
#!/bin/sh
cd $SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
size=100;
size1=`du -a access_log | cut -f1`;
if [ $size1 -gt $size ]; then
mv access_log access_log.old;
mv error_log error_log.old;
kill -USR1 `cat httpd.pid`;
fi
```

Note: If the `SYNC_CUSTOM_DIR` environment variable not defined, specify `$SYNC_DIR/custom` in the script, instead of `$SYNC_CUSTOM_DIR`.

The contents of the `access_log` file are defined in the "Custom Logging Format Aliases" section of the `$SYNC_CUSTOM_DIR/servers/<host>/<port>/conf/httpd.conf` file. If you want to modify the information that is tracked in the `access_log` file, edit the "Custom Logging Format Aliases" section of the `httpd.conf` file. See <http://www.apache.org/docs> for further information.

Note: `SYNC_CUSTOM_DIR` defaults to `$SYNC_DIR/custom`.

Related Topics

Running a SyncServer in Debug Mode

Mirror Log Files

ENOVIA Synchronicity Command Reference: synctrace set

ENOVIA Synchronicity Command Reference: synctrace unset

Using Environment Variables

Running a SyncServer in Debug Mode

There are several approaches to debugging a server:

- Restarting the server with debug environment variables set
- Using `syncdiag servercheck` for extensive OS-level debugging

Restarting the server with debug environment variables set

The simplest debug approach is to restart the server, with debug environment variables set:

```
% stop_sync_server
% setenv Syncdata trace
% setenv SYNC_TRACE 0
% start_sync_server
```

`setenv SYNC_DBG trace` starts the server with Operating System (OS) level tracing. OS-level tracing captures debug information pertaining to the Operating System. The output is logged to `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/trace_log`. The trace program used is determined by the server's platform.

`setenv SYNC_TRACE 0` starts the server with DesignSync tracing. DesignSync tracing captures debug information pertaining to ENOVIA Synchronicity DesignSync software. The output is logged to `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/error_log`.

To disable DesignSync tracing and OS-level tracing, restart the server without the debug environment variables set:

```
% stop_sync_server
% unsetenv SYNC_DBG
% unsetenv SYNC_TRACE
% start_sync_server
```

Use the **synctrace set** command to enable DesignSync-level tracing on a running server, so that you do not have to restart the server. Then, use the **synctrace unset** command later to disable DesignSync tracing.

Culling the Server's error_log

Use `syncdiag error_log` to excerpt information from the server's `error_log`, based on start and end dates and times. In addition to the specified time range, certain `error_log` events are always included in the output, such as server restarts and crashes.

For information on the `syncdiag error_log` options, in your OS shell, run:

```
% syncdiag -help error_log
```

Using `syncdiag servercheck` for extensive OS-level debugging

Use `syncdiag servercheck` for more extensive OS-level debugging. The `syncdiag servercheck` utility

- Does not require restarting the server
- Can attach either to the server's child processes or only the parent process (issuing a server reset to spawn new child processes that are then traced)
- Has an option to trim, compress, or rotate `trace_log` files

For information on the `syncdiag servercheck` options, in your OS shell, run:

```
% syncdiag -help servercheck
```

Available options include:

Option	Description
<code>-keeplogs <n></code>	With <code>-trimtrace</code> , keep <code><n></code> old trace files, plus current one; default is 4
<code>-logdir <dir></code>	With <code>-starttrace</code> , put trace logs in directory <code><dir></code> ; default is <code>\$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs</code>
<code>-port <p></code>	Server port
<code>-pstack</code>	Run <code>pstack</code> on all server processes
<code>-reset</code>	Reset server after <code>-starttrace</code> / <code>-stoptrace</code> ; default is to attach to running server processes
<code>-starttrace</code>	Start <code>truss</code> / <code>strace</code>
<code>-stoptrace</code>	Stop running <code>truss</code> / <code>strace</code>
<code>-trimtrace</code>	Trim log files while <code>truss</code> / <code>strace</code> is running; keeps trace logs from growing unbounded

Example 1: Start tracing, wait, then stop tracing

```
% syncdiag servercheck -starttrace
```

```
% syncdiag servercheck -stoptrace
```

Example 2: Start tracing, reset the server, and send the logs to /tmp.

Use `-trimtrace` to limit the tracing to 8 log files (2 hours worth).

```
% syncdiag servercheck -starttrace -reset -logdir /tmp
```

```
% syncdiag servercheck -trimtrace -keeplogs 8
```

Related Topics

Overview of syncdiag Utility

About SyncServer Log Files

ENOVIA Synchronicity Command Reference: synctrace set

ENOVIA Synchronicity Command Reference: synctrace unset

SyncServer Error Log File Messages

The `error_log` file for a server records server activities and the actions of server-side triggers. This file is located at:

```
<SYNC_CUSTOM_DIR>/servers/<host>/<port>/logs/error_log
```

Trigger Errors

The `error_log` file contains all errors generated by triggers. For example, if the trigger script is not found, you see a message like the following:

```
stg: Error 1: trigger: Couldn't find file "set_state.tcl" in the
following directories: ...
```

The error message lists the search path used to try to find the trigger file.

Email Reply Failure Errors

Your log file may contain a message such as:

```
Connect failure. [server] Server may have reset the connection.
Tue Sep 16 09:11:45 AM EDT 2003 :16671 read_email: Unexpected
failure...
```

This means that an email reply to a SyncNote was not appended as intended. This situation may arise if a user sends an email reply when the server is suspended. The text of the email reply appears at the end of the error message.

Unhandled Error Condition Reports

When DesignSync encounters an error that it cannot resolve, it generates an "unhandled error condition" report (a Tcl stack trace) that gives information on the cause of the error.

By default, the error information is added to the `error_log` file unless you have server administration access rights.

Troubleshooting Email Notification of RevisionControl Notes

Sometimes, after you set up automatic email notification of RevisionControl notes, users may report problems with receiving or not receiving notification.

Problem: Users report they are receiving email notification when they did not subscribe for it.

By default, DesignSync does not send email notification to the user who performed the revision control operation when generating a RevisionControl note. Users have to subscribe to receive notification.

To enable notification for a RevisionControl note, use the **Fields Used to Build Distribution List** in the Email Administrator.

Note: Removing the Author field disables email notification to the author for all users. If you disable the field and a user still wants to receive these email notifications, he or she can set up an email subscription. See *ProjectSync User's Guide: Adding Email Subscriptions*.

Related Topics

How Email Notification Works

Communication Errors Troubleshooting

Controlling Network Communication Timeouts

As a system administrator, you can use the **Limit Transfer Timeout** setting on Performance Tab to help prevent network communication timeouts when using DesignSync.

DesignSync's default behavior is never to time out. This behavior ensures that, if communication between the client and server is interrupted, DesignSync continues processing immediately the communication is re-established.

When you turn on the **Limit Transfer Timeout** setting, the number of seconds you set in the **Transfer Timeout** field represents the maximum time of inactivity between the client and server, that is, between sending and receiving TCP/IP packets. This setting does not represent the total time of a read/write operation on a file. The default timeout is 500 seconds. If you set the **Transfer Timeout** value too low, for example 300 seconds, DesignSync might exit unnecessarily.

Note:

If, at your site, you typically move large files of data between clients and servers you should increase the time out value accordingly. For example, a 30 GB file could take 30 minutes (1800 seconds) to check in on the server side.

To turn on the **Limit Transfer Timeout** setting:

1. Start SyncAdmin.
2. Navigate to the **Performance** Tab.
3. Turn on **Limit Transfer Timeout**.

The **Transfer Timeout** field becomes active, with 500 seconds set as the default timeout value.

4. Change the **Transfer Timeout** value if necessary.
5. Click **Apply**.

The connection between the DesignSync client and server will time out after the time period you indicated in the **Transfer Timeout** field has elapsed without activity and you will see one of the following messages:

"Communication Read Failure"

"Communication Write Failure"

You might see a "Communication Connect Failure" message if there has been more than one read or write attempt. At this point, the read or write operation is aborted.

Proxy Errors

Problem: Checking in a large file fails with the following error message:

```
som-E-13: Communication Zero Content.
```

Possible Cause: If you are connecting to a DesignSync server through a proxy server, the request could be timing out waiting for data to be returned from the DesignSync server.

Solution: Increase the proxy server's default timeout for services.

Problem: Attempting to access a SyncServer returns the following error message:

```
som-E-6: Communication Write Failure.
```

Solution: This is caused by an invalid "ProxyNamePort" environment variable setting or registry key. The process is probably being killed by the Windows kernel due to proxy errors that cannot be trapped. For more information on setting up HTTP Proxies, see About HTTP Proxy.

Mirror System Troubleshooting

Detect "Absent Files" (FindAbsentFiles)

This setting detects Absent Files in the mirror directories. "Absent Files" are DesignSync references that do not appear on disk. If there are Absent Files in a mirror directory, that indicates a problem.

If `FindAbsentFiles` is set to a value of `dword:1`, the MAS will detect Absent Files in mirrors. An e-mail message will be sent to users on the Notify List for the problem mirror, listing the Absent Files. A message will be sent once every 24 hours, while Absent Files remain in the mirror

To correct the problem mirrors, force a full `populate` (via `-unifystate`) update of the mirror, by issuing a `mirror reset` in DesignSync, or `Mirror Reset` in DesignSync Web UI. That will replace DesignSync references with local file copies.

Note: Turning off absent file detection provides increased performance, however, it may have an impact on usability.

This registry setting pertains to a SyncServer that is configured to be a Mirror Administration Server (MAS). To modify registry values programmatically, use the `sregistry` command set. The settings must be made to the server's `$_SYNC_CUSTOM_DIR/servers/<host>/<port>/PortRegistry.reg` file, and the server restarted, for the changes to take effect.

Registry Setting

```
HKEY_LOCAL_MACHINE\Software\Synchronicity\General\Mirrors\Options\FindAbsentFiles=dword:1
```

Allowed Values

Key value	Description
<code>FindAbsentFiles=dword:0</code>	Disallows checking for "Absent files" in mirrors.
<code>FindAbsentFiles=dword:1</code>	Allows checking for "Absent files" in mirrors. (Default)

Related Topic

Save all mup log files (SaveLogFiles)

First Steps for Troubleshooting the Mirror System

When you encounter a problem with the mirror system, capture the current log files, by using the `syncdiag` utility. See [Mirror Log Files](#) for details.

If there are heartbeat failures, reset the mirror daemons.

If `Absent Files` are appearing in the mirror, see [Detect "Absent Files" \(FindAbsentFiles\)](#).

If a mirror is not being updated, but its mirror status shows as enabled and up-to-date, as the mirror owner, in `DesignSync`, `cd` into the mirror directory, and run `populate -recursive -force -get -report verbose`. If that, or the same command with the addition of `-full -unifystate`, do not fetch the expected data, then the mirror definition is probably incorrect (either its vault URL or selector).

The transaction log database file shown in the [Architecture of the Mirror System](#) is located in `syncdata/<host>/<port>/server_vault/tlog.db`. As shown in the architecture diagram, the `tlog.db` file exists on both RS's and MAS's. If Customer Support needs a copy of the `tlog.db` file, if the server is currently running the `tlog.db` file will not be flushed to disk, so will be incomplete.

If the server is stopped, to capture the content of the `tlog.db` file, in UNIX:

```
% cd syncdata/<host>/<port>/server_vault/Partitions/Default
% db_dump -p tlog.db > ~/tlog_listing
```

("-p" is to format the binary data into ascii)

If the server is running, to capture the content of the `tlog.db` file, in UNIX:

```
% db_dump -p -h $SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
~/path/to/syncdata/<host>/<port>/server_vault/Partitions/Default/tlog.db > ~/tlog_listing
```

Related topics

[Mirror Log Files](#)

[Resetting the Mirror Daemons](#)

Architecture of the Mirror System

Mirror Log Files

When you use the mirror system several log files are created logging the operations you perform.

- Log files for Repository Server
- Log files for Mirror Administration Server
- dss_*.log from the Mirror Update Process
- Log Files from Failed Mirror Update
- Collecting log files for debugging

For both the Repository Server and the Mirror Administration Server, mirror tracing is enabled through the General Settings panel in the Mirrors section of the DesignSync menu in the DesignSync Web UI. When enabled, the mirror system trace messages are sent to the relevant log file for the server that has tracing enabled.

Log files for Repository Server

If your SyncServer is setup as a Repository Server (RS), it will create the following log file:

```
$SYNC_CUSTOM_DIR/servers/RS-host/RS-port/share/content/mirrors/logs/mpdlog.txt
```

To view the log file using a browser, go to the **Mirrors** section of the **DesignSync** menu of the RS. Then select the "View All Mirrors Log" button in the Show RS Status panel. This file shows when mirrors are added, modified or removed, but not in an easily readable format.

Note: These log files are automatically rotated if the file size exceeds one megabyte. The file size limit is configurable. See Registry Settings for a Repository Server for details. The mpdlog.txt file is rotated to mpdlog.txt.last. If tracing is enabled, the log files are not rotated.

Log files for Mirror Administration Server

If your SyncServer is setup as a Mirror Administration Server (MAS), it will create the following log file:

```
$SYNC_CUSTOM_DIR/servers/MAS-host/MAS-port/share/content/mirrors/logs/madlog.txt
```

To view the log file using a browser, go to the **Mirrors** section of the DesignSync menu of the MAS. Then select the "**View All Mirrors Log**" button in the Show MAS Status

panel. This file contains information such as new mirrors added, mirrors removed, resets applied.

If tracing was set (using the "Activate mirror tracing" option), much more information is output, such as the commands to spawn the mups, the failures and successful updates, and when failed emails are sent or failures are cleared up.

Note: These log files are automatically rotated if the file size exceeds one megabyte. The file size limit is configurable. See Registry Settings for a Mirror Administration Server for details. The madlog.txt file is rotated to mapdlog.txt.last. If tracing is enabled, the log files are not rotated.

Detailed information for debugging

dss*.log files from failed mirror updates are moved to the `$_SYNC_CUSTOM_DIR/servers/MAS-host/MAS-port/share/content/mirrors/logs` directory, so they could be viewed on the server via a browser. Every failure of a specific mirror overwrites the previous failed dss*log file, since the log file is moved to a hashed name.txt.

By looking in the log file you can determine the mirror name, one of the arguments that the mup was spawned with. Also, each transaction that is being applied by this MUP is also output to the file, along with comments when submirrors are being created, removed, etc. This information in the log file pertains to either a failed dss* log file or a successful one (.../logs/muplogs/).

Also in the `$_SYNC_CUSTOM_DIR/servers/MAS-host/MAS-port/share/content/mirrors/logs` directory is the mad.log.dss, the initial MAD startup dss*log file. If dss logging is disabled, this will show nothing, because upon starting a MAD we disable dss log file logging and then do our own logging, to the madlog.txt file.

A MAS will also have a `$_SYNC_CUSTOM_DIR/servers/MAS-host/MAS-port/share/mirrors` directory, containing:

- a madmirrors.txt file, with all the status information that the MAS uses to report status
- individual MUP change set files that are written by the MAD and read by the MUP
- a MirrorRegistry.reg file, written by the MAD on startup, and used by the MUP

dss_*.log files from the Mirror Update Process

The dss_*.log files from the mirror update process are written to:
`$_SYNC_CUSTOM_DIR/servers/MAS-host/MAS-port/logs/muplogs`

This directory uses the normal `dss_*.log` cleanup procedures (see `dss_*.log` file). However, the muplogs can be saved aside, for debugging. See Registry Settings for Troubleshooting the Mirror System for details.

If mirror tracing is enabled for the MAS, then these log files will contain additional tracing information, similar to the tracing information captured in the `dss_*.log` file for a DesignSync client.

The filenames use the `dss_<date>_<time>_<mirror_update_process_id>.log` format. You need to grep on the mirror filename to find the corresponding log file for a specific mirror.

Log Files from Failed Mirror Update

The `dss_*.log` files from failed mirror update go to:

```
$SYNC_CUSTOM_DIR/servers/MAS-host/MAS-  
port/share/content/mirrors/logs
```

To view the log file using a browser after a mirror update failure has occurred, select the "First Failure" column on the Show MAS Status panel under the **Mirrors** section of the **DesignSync** menu. Every instance of failure of a specific mirror overwrites the previous failed log file since the log file is moved to a unique file name. The name of the text file can be seen by selecting the link.

Looking in the directory, you can also find the log file corresponding to a specific mirror since it contains the arguments that the mirror update process was spawned with and one of these arguments is the mirror name.

Collecting log files for debugging

Use the `syncdiag` utility to collect log files pertaining to the mirror system. For command syntax and usage details, run this command in a UNIX shell:

```
% syncdiag -help mirror
```

The resulting `mirror_logs.tgz` package should be provided to Customer Support.

Related topics

[About DesignSync Log Files](#)

[Overview of syncdiag Utility](#)

[Architecture of the Mirror System](#)

Log file size (LogFileMaxSize)
Detect "Absent Files" (FindAbsentFiles)
Save all mup log files (SaveLogFiles)
Failure notify time
Failure retries
Failure notify interval
Log file size
Check tags for mirror update

Resetting The Mirror Daemons

The **Reset MAS Daemon** (mad) and **Reset RS Daemon** (mpd) hyperlinks in the **DesignSync** menu in the DesignSync web UI, let you reset your mirror daemons for the server. Resetting the mirror daemons terminates the existing daemon process and starts a new one. These hyperlinks appear only when you have AdministrateServer access permissions. (See the ENOVIA Synchronicity Access Control Guide for details on access rules.)

Resetting the RS daemon clears heartbeat failures. The most common reason for heartbeat failures is an IP address change, typically on a system hosting a Mirror Administration Server (MAS). The mpd (mirror push daemon) on the Repository Server (RS) cached the original IP address, so can no longer communicate with its MAS's. Restarting the RS will clear this problem. However, you will not have to restart the entire server, only its mpd.

Note: In most cases, when a problem is detected with a mirror daemon, DesignSync automatically resets the daemon.

To reset the Mirror Daemon:

1. Click **Reset MAS Daemon** or **Reset RS Daemon** in the **Mirrors** section of the DesignSync menu.
2. Click **OK** on the confirmation pop-up window.

When the daemon has been reset, you see an Operation Successful panel that confirms the operation.

Note: To reset mirror services for all servers at your site or all servers at all sites, use the `resetmirrordaemons` command. For command syntax and usage details, enter the following command in a DesignSync client:

```
stcl> resetmirrordaemons -help
```

Related Topics

Architecture of the Mirror System

Diagnosing Performance

Overview of syncdiag Utility

The diagnostic utility `syncdiag` can be used to debug a DesignSync client command, to debug a SyncServer, and to diagnose performance. Also you can use the `syncdiag` diagnostic for debugging the mirror system by collecting mirror related log files. Note that on Windows 7, you must run the DesignSync shell with Administrator privileges, in order to run `syncdiag` within the DesignSync shell.

For a list of the main options and diagnostics available to `syncdiag`, in an OS shell run:

```
% syncdiag -help
```

General diagnostics include:

<code>syncinfo</code>	Print information about the DesignSync installation
-----------------------	---

Main options to `syncdiag` include:

<code>-help</code>	With no arguments, the <code>-help</code> option reports a one line summary for each option available. When a diagnostic argument is specified, the <code>-help</code> option reports usage information for the argument.
<code>-version</code>	Prints the version number of <code>syncdiag</code>

Specific diagnostics include:

<code>dsperf</code>	Collect performance stats for DesignSync commands
<code>error_log</code>	Cull the server's <code>error_log</code>

link-in	Test link-in with specific workspace and server
mirror	Diagnostics for the mirror system
modperf	Collect performance statistics for modules commands
oscheck	Check your system for recommended patches
perf	Display file system performance stats
servercheck	Check or trace a running server
trace	Run a DesignSync client command with debug enabled

The command syntax for `syncdiag` is:

```
% syncdiag -[main options] <diagnostic> -[diagnostic options]
```

For specific help on a diagnostic, in an OS shell, run:

```
% syncdiag -help <diagnostic>
```

or

```
%syncdiag <diagnostic> -help
```

For example:

```
% syncdiag -help syncinfo
```

syncdiag syncinfo

The most common use of `syncdiag syncinfo` is to report the software that is installed:

```
% syncdiag syncinfo -version
```

The above command lists, for each platform installed:

- the base release
- any service packs
- any hot fixes
- any product overlays

Related Topics

[Running a DesignSync Command in Debug Mode](#)

[Running a SyncServer in Debug Mode](#)

Diagnosing Performance

Mirror Log Files

Diagnosing Performance

To diagnose performance, you can:

- Check whether required Operating System (OS) patches are installed
- Display performance statistics pertaining to the system environment
- Collect performance statistics for DesignSync commands
- Test whether the environment supports the link-in performance optimization

Check whether required OS patches are installed.

Use `syncdiag oscheck` to check your system for recommended patches. For information on the current `syncdiag oscheck` options, in your OS shell, run:

```
% syncdiag -help oscheck
```

Display performance statistics pertaining to the system environment.

To analyze your system and network environment, you should run `syncdiag perf`

- From a client workspace, when logged into a client machine
- From the server's `syncdata/<host>/<port>/server_vault` directory, when logged into the server machine

For information on the current `syncdiag perf` options, in your OS shell, run:

```
% syncdiag -help perf
```

Note: The `-ssh` option to `syncdiag perf 'exec's ssh`. If your `.cshrc` is interactive (sourcing the `.cshrc` file prompts you to respond), then 'exec' ing `ssh` will hang.

Collect performance statistics for DesignSync commands.

The `syncdiag dsperf` utility measures the performance of DesignSync commands. OS commands are also run, for comparison. The `dsperf` tests create a data set, which is used for client-server tests. Run the `dsperf` tests in a test environment comparable to the production environment. Use a separate vault area, so that users do not inadvertently populate the test data.

Collect performance statistics for modules commands

The `syncdiag modperf` utility measures the performance of Modules commands. OS commands are also run, for comparison. The `modperf` tests create a module with a data set to use for client-server tests. Run the `modperf` tests in a test environment comparable to the production environment. Specify a module that does not exist. No other users should be using that module during the test. You may control access to the module by using access controls or other mechanism to restrict access. After the test has completed, the created module is removed.

UNIX Set-Up

Remove interactive commands from your `.cshrc` file.

Edit your `.cshrc` file, commenting out any lines that run interactive programs. Sourcing the `.cshrc` file cannot prompt for input, or the `-scp` tests will hang. Similarly, comment out any `stty` lines, as these lines will cause the `-scp` tests to fail.

Configure `ssh` to let you `ssh` to another host without requiring a password.

You must configure `ssh` to allow you to `ssh` to another host, without requiring a password. Any prompting or output from `ssh` will cause the `-scp` tests to fail. Follow these steps:

1. Generate a public key:

```
% ssh-keygen -t ssh-rsa
```

(Enter a carriage return at the password prompts.)

2. Add the public key to the authorized users list:

```
% cat .ssh/id_rsa.pub >> .ssh/authorized_keys2
```

3. Create a `.ssh/config` file containing the following:

```
Host *
# two things:
# 1) do not ask me to accept new host key; just do it
# 2) do not fail if the host key changes
StrictHostKeyChecking no
# only use proto 2
Protocol 2
```

4. Restrict the UNIX permissions on the `.ssh` files:

```
% chmod 600 .ssh/*
```

Test that `scp` is in your `$PATH`, and works:

DesignSync Data Manager Administrator's Guide

```
[tbarbg6@sting] echo "testing" > testfile
[tbarbg6@sting] scp testfile lotti:testfile

testfile                               100%
|*****|                               8      00:00
```

In the above example, tbarbg6 is the user running the tests, on the client machine "sting", to the server hosted on "lotti".

DesignSync Set-Up

1. If the server requires user authentication, run:

```
% dssc password -save <host>:<port>
```

This saves the username/password for the server. Otherwise, the `dsperf` tests will fail immediately. The tests do not yet prompt for user authentication, so the workaround is to save a username/password for the server to use automatically.

2. Configure for RevisionControl notes and email.

If RevisionControl note generation is set, DesignSync generates RevisionControl notes, and subsequently sends email notification to subscribers. RevisionControl note generation itself impacts performance.

Also, if **sendmail** is used, and **sendmail** is run in the foreground, that will slow performance. Use the Email Administrator to specify that **sendmail** be run in the background. The **SMTP** mailer always runs in the foreground, and does not have an option to background it.

3. Ensure that the user running the tests has "Delete when TYPE Vault" access, so that the tests can remove the data that is created on the server.

To ensure effective performance, avoid the following:

- Do not set OS-level tracing (for example, `SYNC_DBG`), as that will slow down performance substantially. Also, on Linux RedHat 8.0, tracing a process causes the process being traced to hang.
- Do not set `SYNC_TRACE` to 0, as that also will slow performance. If Sync-level tracing is requested by Support, you will be requested to set `SYNC_TRACE` to 26, to only capture performance-related debug information.
- Do not interrupt the tests. Interrupting tests leaves data in the workspace and in the vault. On Windows, after interrupting `dsperf`, attempts to exit the

DOS shell hang. You must kill the leftover **stcl.exe** process, via the Task Manager.

Usage

For information on the current `syncdiag dsperf` options, in your OS shell, run:

```
% syncdiag -help dsperf
```

Besides the options shown by `syncdiag -help dsperf`, there is a hidden **-nfiles** option, to specify the number of test files to create and use. The current **-nfiles** default value of 3168 takes hours to run. Instead, specify a lower **-nfiles** value, such as 100.

You might want to run several tests, with increasing **-nfiles** values. Note that the **-nfiles** value specified does not map exactly to the number of files used, because of how the data set is constructed. But the same **-nfiles** value will always result in the same data set.

The **-dataset** option takes as its value a directory path to an existing workspace. The `syncdiag` utility directory and files in the workspace are copied into `syncdiag`'s **-dir** working directory, and used as the data set for the `dsperf` tests. The **-dataset** option is mutually exclusive with the **-nfiles** option.

Running the Tests

In the example below, the tests are run from the client machine `sting` to a server hosted on the machine `lotti`. The server's host machine is also the host specified for the **-scp** tests.

The "SCP" test is skipped unless you specify the **-scp** option. The reason you need the option (and it is not the default) is because you have to specify **-scp <hostname>** or **-scp <user>@<hostname>**, and `scp` must be in your PATH.

For example:

```
[tbarbg6@sting] syncdiag dsperf -vault sync://lotti:30158/Perf
-nfiles 100 -scp lotti
```

This command creates a `syncdiag.log` file, and also outputs the `syncdiag.log` content to the screen. A `dss*.log` file is created as usual.

The ending summary looks like:

RESULTS

125 files in 9 directories

Test	Avg Time (seconds)
-----	-----
Initial import of entire design to the vault	19.01
Populate a new GET workspace	15.08
cp -r workspace with metadata	12.11
scp -r workspace with metadata	14.74
Tag entire design to make a configuration v1	1.09
List the contents of a configuration	0.36
Checkout (lock) and change all files	3.80
Status check on all locked files	0.21
Compare workspace to vault (all files changed)	0.61
Cancel lock/revert all files	21.12
Checkin all files after re-checkout	20.17
Re-populate all files (in a different workspace)	14.44
Tag entire design to make a configuration v2	0.97
Compare two configurations	0.52
Checkout lock a handful of changes	0.80
Status check on handful of locked files	0.24
Status check on handful of local mods	0.15
Compare workspace to vault (handful of files changed)	0.55
Cancel lock/revert handful of files	3.13
Checkout lock a handful of changes 2	0.68
Checkin handful of changes	3.12
Re-populate handful of files (in a different workspace) 2	3.54
Checkout without lock all files	13.58
Checkout without lock a handful of changes	1.85
Status check on all local modifications	0.30
Switch configurations	14.44

This summary is what a Support Analyst will request.

Note: The message, "0: No Such Object", is output towards the beginning of the test run. This message is harmless.

Test whether the environment supports the link-in performance optimization.

Use `syncdiag link-in` to test whether the link-in performance optimization is available for a specific client workspace and server combination.

For information on the current `syncdiag link-in` options, in your OS shell, run:

```
% syncdiag -help link-in
```

Related Topics

About DesignSync Client Log Files

Overview of syncdiag Utility

ENOVIA Synchronicity Access Control Guide: User Authentication Access Controls

Configuring Email Notifications

Linking Large Files

Running a DesignSync Client in Debug Mode

There are several approaches to debugging a client:

- Restarting the client with debug environment variables set
- Enabling debug output within a DesignSync client session
- Using `syncdiag trace` for extensive OS-level debugging

Restarting the client with debug environment variables set

The simplest debug approach is to restart the client, with a debug environment variable set:

```
% setenv SYNC_TRACE 0
```

Then start a DesignSync client.

`setenv SYNC_TRACE 0` starts the client with DesignSync tracing. DesignSync tracing captures debug information pertaining to DesignSync software. The output is logged to the user's `$HOME/dss*.log` file and to their `$HOME/.synchronicity/logs/sync_client_trace*.log` file.

To disable DesignSync tracing, restart the client without the debug environment variable set:

```
% unsetenv SYNC_TRACE
```

Then start a DesignSync client.

Enabling debug output within a DesignSync client session

Use the **synctrace set** command to enable DesignSync-level tracing within a DesignSync session, so that you do not have to restart the client. Then, use the **synctrace unset** command later to disable DesignSync tracing.

Using syncdiag trace for extensive OS-level debugging

Use **syncdiag trace** to run a DesignSync command with DesignSync tracing, with Operating System (OS) level tracing, or with both levels of tracing. DesignSync tracing captures debug information pertaining to ENOVIA Synchronicity DesignSync Data Manager software, as if the command **synctrace set** has been run. OS-level tracing captures debug information pertaining to the Operating System. DesignSync tracing, and OS-level tracing, are only enabled for the duration of the DesignSync command run by `syncdiag trace`.

For information on the current `syncdiag trace` options, in your OS shell, run:

```
% syncdiag -help trace
```

Available options include:

<code>-sync</code>	Run with DesignSync tracing (SYNC_TRACE)
<code>-os</code>	Run with OS level tracing
<code>-all</code>	Run with both OS and DesignSync tracing

The default behavior is **-all**.

For example:

```
% syncdiag trace -sync dssc ci -share -comment "Test" testfile
```

The above command creates a `syncdiag.tgz` package containing the debug log files.

Notes:

- If you are using an SUID installation, you need to run the `syncdiag trace` command as the installation owner.
- The OS-level tracing program used is platform dependent. For HP, your system administrator might need to download "tusc".

Related Topics

[Overview of syncdiag Utility](#)

[About DesignSync Client Log Files](#)

ENOVIA Synchronicity Command Reference: `synctrace set`

ENOVIA Synchronicity Command Reference: `synctrace unset`

Running a SyncServer in Debug Mode

There are several approaches to debugging a server:

- Restarting the server with debug environment variables set
- Using `syncdiag servercheck` for extensive OS-level debugging

Restarting the server with debug environment variables set

The simplest debug approach is to restart the server, with debug environment variables set:

```
% stop_sync_server
% setenv Syncdata trace
% setenv SYNC_TRACE 0
% start_sync_server
```

`setenv SYNC_DBG trace` starts the server with Operating System (OS) level tracing. OS-level tracing captures debug information pertaining to the Operating System. The output is logged to

`$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/trace_log`. The trace program used is determined by the server's platform.

`setenv SYNC_TRACE 0` starts the server with DesignSync tracing. DesignSync tracing captures debug information pertaining to ENOVIA Synchronicity DesignSync software. The output is logged to

`$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs/error_log`.

To disable DesignSync tracing and OS-level tracing, restart the server without the debug environment variables set:

```
% stop_sync_server
% unsetenv SYNC_DBG
% unsetenv SYNC_TRACE
% start_sync_server
```

Use the **synctrace set** command to enable DesignSync-level tracing on a running server, so that you do not have to restart the server. Then, use the **synctrace unset** command later to disable DesignSync tracing.

Culling the Server's error_log

Use `syncdiag error_log` to excerpt information from the server's `error_log`, based on start and end dates and times. In addition to the specified time range, certain `error_log` events are always included in the output, such as server restarts and crashes.

For information on the `syncdiag error_log` options, in your OS shell, run:

```
% syncdiag -help error_log
```

Using syncdiag servercheck for extensive OS-level debugging

Use `syncdiag servercheck` for more extensive OS-level debugging. The `syncdiag servercheck` utility

- Does not require restarting the server
- Can attach either to the server's child processes or only the parent process (issuing a server reset to spawn new child processes that are then traced)
- Has an option to trim, compress, or rotate `trace_log` files

For information on the `syncdiag servercheck` options, in your OS shell, run:

```
% syncdiag -help servercheck
```

Available options include:

Option	Description
-keeplogs <n>	With -trimtrace, keep <n> old trace files, plus current one; default is 4
-logdir <dir>	With -starttrace, put trace logs in directory <dir>; default is \$SYNC_CUSTOM_DIR/servers/<host>/<port>/logs
-port <p>	Server port
-pstack	Run pstack on all server processes
-reset	Reset server after -starttrace/-stoptrace; default is to attach to running server processes
-starttrace	Start truss/strace
-stoptrace	Stop running truss/strace

<code>-trimtrace</code>	Trim log files while <code>truss/strace</code> is running; keeps trace logs from growing unbounded
-------------------------	--

Example 1: Start tracing, wait, then stop tracing

```
% syncdiag servercheck -starttrace
% syncdiag servercheck -stoptrace
```

Example 2: Start tracing, reset the server, and send the logs to /tmp.

Use `-trimtrace` to limit the tracing to 8 log files (2 hours worth).

```
% syncdiag servercheck -starttrace -reset -logdir /tmp
% syncdiag servercheck -trimtrace -keeplogs 8
```

Related Topics

[Overview of syncdiag Utility](#)

[About SyncServer Log Files](#)

[ENOVIA Synchronicity Command Reference: synctrace set](#)

[ENOVIA Synchronicity Command Reference: synctrace unset](#)

External Modules

Unable to locate external module interface command Error

If you perform a defined external module action and receive the following error message:

```
Unable to locate external module interface command
'::ExternalModule::InterfaceName::DesignSyncCommand.
```

There was probably a `tclIndex` file already present in the `$SYNC_CUSTOM/custom/site/share/client/tcl` directory when you created your external module interface files. You must delete `tclIndex` file. It will be re-created automatically when the next `stcl` process is run. The newly created file includes the new external module interface file.

Additional Information

Add/Edit Tool Pane

Use the **Add/Edit Tool** pane to add new custom tools and to edit existing tools. You can specify the following properties:

- The **name** of the tool as it appears in the menu. You must enter a unique name for your custom tool. The name cannot be longer than 40 characters.
- The **icon** is optional, but a tool that does not have an icon cannot be placed in the tool bar. The dropdown list provides some suggested icons, but you can also specify a full path, using environment variables, if desired, to any gif or jpg file. For example, if you had a `$SYNC_PROJECT_CFGDIR` variable defined to `/home/rsmith/PrjCfg` (or on Windows `C:/home/rsmith/PrjCfg`), any of the following icon definitions would be valid, and equivalent:
 - `/home/rsmith/PrjCfg/icons/appicon.ico`
 - `$SYNC_PROJECT_CFGDIR/icons/blue.gif`
 - `${SYNC_PROJECT_CFGDIR}/icons/blue.gif`
 - `%SYNC_PROJECT_CFGDIR%/icons/blue.gif`
- The **description** of the tool as it appears in the status bar when you drag the mouse over the menu entry. The description cannot be longer than 80 characters.
- The **Type** indicates whether the tool executes **tcl** or **Tk** code, or a **shell** command.
- The **command** is either the tcl or shell command to execute.
- If you check **Append selected objects to command**, the URLs of the selected objects will be appended to the command before it is executed.
- **Output** determines where the display output of the tool:
 - **Send output to output region** displays output in the DesignSync output window.
 - **Create text report** displays the output as a new view in the View Pane.
 - **Create HTML report** displays the output as a new view in the View Pane. The output may contain html formatting tags.

Note: When a tool is executed, information about the tool is displayed in the output window. Custom tcl tools displays the name of the tool in comments, and then the highlighted executed command. Custom shell tools displays the name and executed shell command in comments.

Customizing the List View

The DesignSync list view can be customized to suit your working style. For example, you can remove columns that are not useful to you, and move the most important columns towards the left part of the view.

One of the columns in the list view is the sorted column. This is indicated by an arrow in the column header. You may sort the display by a different column by clicking on that column's header; clicking on the header of a column that is already sorted toggles between descending and ascending sort.

You may rearrange the order columns in the list view by clicking and dragging the column header. For example, if the **Locker** column is important to you, click on the header of the **Locker** column and drag it to the immediate right of the **Name** column.

Clicking the right mouse button on a column header displays a context menu with the following options:

- **Sort Descending/Sort Ascending** sorts the display by that column.
- **Hide Column** removes the column from the display. For example, if you do not want the **Size** column displayed in the list view, click the right mouse button on the header of the **Size** column and select **Hide Column**.
- **Show/Hide Columns** displays the pane with which columns may be hidden or restored to the list view.
- **Reset Columns** restores the display and order of columns to the default.

Color Selection dialog

The color selection pane is used to select colors for highlighting. The pane has three pages: **Swatches**, **HSB**, and **RGB**. Each page provides a different way to select a color.

General Exclusions

DesignSync users cannot override the following settings that were specified during installation of DesignSync software or by a LAN administrator using SyncAdmin:

- Keep vault information on delete by default
- Default cache directory

Tcl Script for Importing Users

```
#####
# Copyright © 1997-2010, Dassault Systemes. All rights reserved.      #
# Use of this source code is restricted to the terms of your          #
# license agreement with Dassault Systemes. Any use, reproduction   #
# distribution, copying or re-distribution of this code outside      #
```


DesignSync Data Manager Administrator's Guide

```
# the scope of that agreement is a violation of U.S. and International      #
# Copyright laws.                                                            #
#####
#
# This script imports a user whose login name is passed as the "user"
argument.
# The password can be generated through 3 modes (defined by the mode
argument,
#   which defaults to "system"):
# - "system" : the password is found in the /etc/passwd or /etc/shadow file
(depend on availability)
# - "fixed"  : the password is ${id}1234
# - "random" : the password is generated "randomly (equal to [ clock seconds
]) and
#   mailed to ${id}@${domain}
# if any method fails, a secondary mode can be tried (argument mode2, which
defaults to
# "random". The secondary mode is ignored if equal to the primary one.
#
# Failure to read the password file to find the user entry is fatal and does
not
# lead to a second try.
#
#
# MUST BE CHANGED to your own domain
# Default domain to build the email address
#
set domain "acme.com"
#
# Change this to the correct location (this one is most likely OK)
#
set syncSendMail "/usr/lib/sendmail"
# Needs various utilities
source [locate share/tcl/sutil.tcl]
# Retrieve various arguments
cgi_arg user
# Mode can be one of system/fixed/random
cgi_arg mode "system"
# This is a secondary default mode, for the case when the primary ($mode)
fails
cgi_arg mode2 "random"
#####
# Creates the user
#
proc createUser { user id name domain keyMode key } {
    if { 1 == [ catch { user create $id [ list Name $name EmailAddr \
        "$id@$domain" $keyMode $key ] } msg ] } {
        puts "Error: Failed to create user '$user' ($msg)<BR>"
        set success 0
    } else {
        puts "Success: Added user '$user'<BR>"
        set success 1
    }
}
return $success
}
#
# End of createUser
```

```
#####
#####
# Returns the next mode to try with
#
proc secondaryMode { success mode } {
    if { ( $success == 0 ) && ( $mode != $::mode2 ) } {
        set mode $::mode2
    } else {
        set mode ""
    }
    return $mode
}
#
# End of secondaryMode
#####
# Exit if argument not passed
if { $user == "" } {
    puts "Error: no user specified<BR>"
    quit
}
# Exit if unrecognized mode
if { ( $mode != "system" ) && ( $mode != "fixed" ) && ( $mode != "random" ) }
{
    puts "Error: mode must be one of system / fixed / random.<BR>"
    quit
}
# Get all details but password
if { 1 == [ catch { exec grep "^$user:" /etc/passwd } details ] } {
    #or use {exec ypmatch $user passwd} for NIS
    puts "Error: failed to locate entry for user '$user'<BR>"
    quit
}
set detailList [ split $details : ]
# Should be same as $user, but as we're doing a grep...
set id [ lindex $detailList 0 ]
set name [lindex $detailList 4]
set key [lindex $detailList 1]
set keyMode "Key"
while { $mode != "" } {
    puts "Info: Trying in \"$mode\" mode<BR>"
    if { $mode == "system" } {
        if { $key == "x" } {
            # Get password from /etc/shadow
            # Not needed if NIS used; in that case, get it from the previous
call
            if { 1 == [ catch { exec grep "^$user:" /etc/shadow } passwd ] }
{
                puts "Info: Could not get the password from the shadow file
or could not read the shadow file itself<BR>"
                puts "Info: Defaulting to $mode2 password generation
mode.<BR>"
                set mode [ secondaryMode 0 $mode ]
            } else {
                set passwdList [ split $passwd : ]
                set key [lindex $passwdList 1]
                set keyMode "Key"
            }
        }
    }
}
```

DesignSync Data Manager Administrator's Guide

```
        set mode [ secondaryMode [ createUser $user $id $name
$domain $keyMode $key ] $mode ]
    }
    } else {
        set mode [ secondaryMode [ createUser $user $id $name $domain
$keyMode $key ] $mode ]
    }
    } elseif { $mode == "fixed" } {
        set key "${id}1234"
        set keyMode "ClearKey"
        set mode [ secondaryMode [ createUser $user $id $name $domain
$keyMode $key ] $mode ]
    } elseif { $mode == "random" } {
        set key [ clock seconds ]
        set keyMode "ClearKey"
        set mailTo ${id}@${domain}
        set email "To: ${mailTo}\nSubject: Account created on [syncinfo
serverName]:[syncinfo serverPort]\nAn a
ccount has been created for you on [syncinfo serverName]:[syncinfo
serverPort] with the following details\n
username=${id}\n    password=$key\nPlease login (http://[syncinfo
serverName]:[syncinfo serverPort]) and change y
our password ASAP."
        set success [ createUser $user $id $name $domain $keyMode $key ]
        set mode [ secondaryMode $success $mode ]
        if { $success == 1 } {
            # last but not least, mail the user his account and password
            if { 1 == [ catch { exec echo $email | $syncSendMail $mailTo }
message ] } {
                puts "Error: Could not send mail to user.\n$message<BR>"
                quit
            } else {
                puts "Success: Sent mail to ${id}@${domain} with login
details.<BR>"
            }
        }
    }
}
}
```

The CPU Team Subscribes to Email on a Hierarchy

The CPU design is nearing its tape out date; any problem with the design at this point must be scrutinized. Design work is focused on the CPU@C21 module configuration, which contains the CPU module hierarchy. As CPU team leader, Robert wants to know when a defect is filed against any part of the CPU@C21 configuration, including its submodules. To be notified of these defects, Robert uses **Advanced Subscription** panel to subscribe for email about defects logged against the CPU module hierarchy. This type of subscription is called a **hierarchical subscription**.

Note: This scenario assumes that:

- Robert has valid login accounts on all SyncServers with which the hierarchical subscription will communicate. For example, if Robert subscribes for notes on a module and all its submodules, he must have a login account on each submodule server.
- Robert or the DesignSync administrator has mapped note types to ensure that hierarchical subscriptions to notes used on the cpu server include notes on submodule servers using different note types. This note type mapping must be done before performing hierarchical subscriptions or hierarchical queries. See Mapping Note Types for more information.

To subscribe for email notification:

1. In his browser, Robert enters the URL the server for the CPU module (the same as the CPU development server).
2. From the User Profiles menu, he selects **Email Subscriptions** and then clicks **Advanced Subscriptions**. DesignSync displays the **Add New Subscriptions for ...** panel.

Note: Hierarchical subscriptions (subscriptions to email about a module hierarchy) can be done only through the **Advanced Subscriptions** panel.

3. In the Advanced Subscriptions part of the panel, Robert selects **SyncDefect** from the Note Type pull-down menu.
4. Robert wants to receive email about all defects logged against the CPU@C21 module configuration, so he does not enter any value in the **Property Filter** field.

DesignSync filters out, or excludes, items from email notification based on values specified in this field. For example, if Robert wanted to receive email notification only for CPU module defects with stopper priority, he could specify that property value in this field. (For information on specifying values in this field, you can view help information by clicking the **Help** button next to the **Property Filter** field.)

5. From the **Object Filter** pull-down menu, Robert selects CPU. The page displays the URL `sync:///Projects/CPU` in the **Object Filter** field.

Note: The CPU module is displayed below `/Projects` because it is a project object. (All projects display below `/Projects`.)

6. To the `sync:///Projects/CPU` URL, Robert appends the configuration name as `@C21`.

Robert can also browse to find the configuration he wants by selecting **Browse...** from the **Object Filter** pull-down menu. This action displays the Browse Server window, from which he can navigate, selecting projects or

configurations. If Robert has already selected a project (as he did in step 5), the browser is positioned at that project, displaying its configurations for selection.

7. Since Robert wants to be notified of defects not only on the upper-level CPU module but also all of its submodules, he selects **This object and all levels below** from the **Scope** field.
8. When he has completed his subscriptions, Robert clicks **Submit**.

DesignSync sets up the email subscription and displays the Success panel. The panel lists the email subscription for the SyncDefect note type on CPU@C21 and subscriptions added for referenced submodules.

Note: Email subscriptions do not apply to referenced IP Gear deliverables. However, IP Gear users can subscribe to information related to deliverables on the IP Gear server.

Robert Deletes an Email Subscription

Robert is taking an extended vacation and does not want to receive email about the CPU@C21 configuration while he is away.

To delete the email subscriptions to the CPU module and all its submodules, Robert again selects **Email Subscriptions**. The panel lists all of his subscriptions, including the CPU@C21 configuration. Robert clicks **Delete** for CPU@C21. Then he clicks **Submit** to have the change take effect. DesignSync deletes his subscription to not only CPU@C21 but also its submodules, their submodules, and so on down the module hierarchy.

Robert Subscribes to RevisionControl Notes on Module Operations

During design development, module hierarchies can change as submodules are added and removed or aliases are changed. Module users including owners and project leaders, need to know of such changes so they can update their module configurations or determine the effects of the changes on their work areas. To receive notification of such changes, they can use DesignSync to subscribe to email notification of RevisionControl notes for module commands.

For example, suppose the CPU@C21 module configuration has a hierarchical reference to the submodule ALU@CTO19. Robert, the CPU team leader, wants to know of configuration changes at all levels in the CPU@C21 hierarchy. He subscribes to email notification of RevisionControl notes for **rmconf** and **mkconf** operations on the CPU@C21 configuration and all submodules below it in the hierarchy.

Note: These steps assume that Robert or the DesignSync administrator has set up email notification of module Revision Control notes on the alu server. For information, see Setting Up Notification of Module RevisionControl Notes.

1. Robert follows the steps for subscribing for email notification, clicking **Advanced Subscriptions**.
2. At the **Add New Subscriptions for...** panel, for **Note Type**, he chooses **RevisionControl**.
3. For **Property Filter**, he types the string `Tag=` and the module command for which he wants notification. For example, `Tag=rmconf`.

Using this string sets DesignSync to send email only for RevisionControl notes for the **rmconf** operation and no others.

4. From the **Object Filter** pull-down menu, Robert selects CPU. The page displays the URL `sync:///Projects/CPU` in the **Object Filter** field.
5. To the `sync:///Projects/CPU` URL, Robert appends the configuration name as `@C21`.

Robert can also browse to find the configuration he wants by selecting **Browse...** from the **Object Filter** pull-down menu. This action displays the Browse Server window, from which he can navigate, selecting projects or configurations. If Robert has already selected a project (as he did in step 4), the browser is positioned at that project, displaying its configurations for selection.

6. In the **Scope** field, he selects **This object and all levels below**.
7. He clicks **Submit** to have the subscription take effect.

This displays the Success panel after the email subscription is completed. The panel lists the email subscription for the RevisionControl note type on `CPU@C21` and subscriptions added for referenced submodules.

8. Robert repeats the steps, this time subscribing to email notification of RevisionControl notes for **mkconf** operation on the `CPU@C21` configuration.

A few weeks later, when the ALU team removes the `ALU@CTO19` configuration from the alu server, Robert receives email notification of the operation. He can then remove the CPU module's reference to the `ALU@CTO19` module and, if appropriate, replace it with another reference.

Robert Updates Email Subscriptions

Module hierarchies can change after a user has subscribed for email notifications. For example, suppose that the CPU module has a hierarchical reference to the submodule `ALU@GOLDEN` and that `ALU@GOLDEN` is an alias pointing to the `ALU@R1` release. As development progresses, the ALU team creates a new release and then changes the `ALU@GOLDEN` alias to point to a different release, `ALU@R2`.

Or suppose that Robert (the CPU team leader) subscribes for email notification of defects on the CPU@C21 configuration and all of its submodules (a hierarchical subscription). Then some time later, one of the ALU submodule's hierarchical references changes from FPU@R1 to FPU@R2.

In each case, Robert must manually update his email subscriptions to reflect the change.

Manual Update of Email Subscriptions When an Alias Changes

To manage subscriptions when aliases change, Robert takes the following steps.

1. If he has not done so, Robert subscribes to RevisionControl notes for the **mkalias** operation. Taking this action ensures that he will always be notified when an alias changes. For information, see Robert Subscribes to RevisionControl Notes on Module Operations.
2. When he receives email that an **mkalias** operation has taken place on the ALU@GOLDEN object, Robert deletes his subscription to the release to which ALU@GOLDEN previously pointed (in this case ALU@R1). (The email generated from an **mkalias** command includes both the old and the new alias.)
3. Robert then subscribes for email on the ALU@GOLDEN alias again. The subscription now includes notes for the ALU@R2 submodule, to which the ALU@GOLDEN alias currently points.

Manual Update of Email Subscriptions When a Hierarchical Reference Changes

To manage subscriptions when hierarchical references change, Robert takes the following steps.

1. If he has not done so, Robert subscribes to RevisionControl notes for the **addhref** and **rmhref** operations. Taking this action ensures that he will always be notified when a hierarchical reference changes. For information, see Setting Up Email Notification of Module RevisionControl Notes.
2. Robert receives email notifications that an **rmhref** removed ALU@C1's reference to FPU@R1 and a subsequent **addhref** added a reference to FPU@R2. Robert resubscribes for notes on the CPU@C21 module hierarchy (a hierarchical subscription). After Robert resubscribes for notes on the CPU@C21, the subscription includes notes for the FPU@R2 submodule.

Notes:

- The email notification generated from the **rmhref** and **addhref** operations includes the previous and current hierarchical references.

- Although resubscribing to the CPU@C21 module hierarchy includes a subscription to notes on the new FPU@R2 submodule, it does not delete subscriptions to notes on submodules that the hierarchy no longer references (for example, FPU@R1). Robert must manually delete this subscription.

Using the -h Option to Specify a Unique Identifier for a Remote Mirror

The **-h (hostid)** option is an optional argument for the RemoteMirrorRegister, RemoteMirrorUnRegister, RemoteMirrorUpdate, and RemoteMirrorList scripts used in the remote mirror assurance operation. When used in addition to the **-m** (mirror path), the **-h** option uniquely identifies the remote mirror directory. It is recommended to use this option.

You do not have to specify the **-h** option for the remote mirror to have a unique identification. By default, both the RemoteMirrorUpdate script and the RemoteMirrorRegister script use the hostid of the machine where the script runs. **All you have to do is run both scripts on the same machine.** When you run each script, the value for the **-h** option defaults to the machine's hostid, a unique hexadecimal number.

In certain situations however, you may need or want to use the **-h** option:

- When RemoteMirrorRegister and RemoteMirrorUpdate are run on different machines.
- Sometimes the path to the remote mirror directory is not sufficient to uniquely identify it.

For example, there could be two different remote mirror sites, each mirroring the same vault, each having the same path to the mirror directory.

- You want to make it easier to distinguish among mirrors when you display a remote mirror status listing.

For example, suppose you have set up a vault reflected by remote mirrors at three sites: San Jose, Austin, and Tucson. Each mirror has the same path. If you register each without specifying the **-h** option, each mirror is distinguished only by the hostid of the machine, a hexadecimal number. To be able to more quickly and easily distinguish among mirrors, register each mirror with an **-h** option value that specifies its location:

```
% RemoteMirrorRegister -h "SanJose-Mirror" -v
sync://mirrorserver.mycompany.com:2888/Projects/ASIC
```

```
-m /users/admin/Projects/mirror/ASIC
```


DesignSync Data Manager Administrator's Guide

```
% RemoteMirrorRegister -h "Austin-Mirror" -v
sync://mirrorserver.mycompany.com:2888/Projects/ASIC

-m /users/admin/Projects/mirror/ASIC

% RemoteMirrorRegister -h "Tucson-Mirror" -v

sync://mirrorserver.mycompany.com:2888/Projects/ASIC

-m /users/admin/Projects/mirror/ASIC
```

Note: For the names to work, when you set up the RemoteMirrorUpdate cron job on each remote mirror (San Jose, Austin, and Tucson), you must specify the same -h option value you specified with RemoteMirrorRegister. For example, on the machine you registered as SanJose-Mirror, you would define the cron job:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * *
/syncinc/bin/RemoteMirrorUpdate -h "SanJose-Mirror"
/users/admin/Projects/mirror/ASIC > /dev/null/ 2>&1
```

Related Topics

General Mirror Topics:

[Mirroring Overview](#)

[Mirrors Versus Caches](#)

[Using a Mirror](#)

Mirror Administration Topics:

[Administering Mirrors](#)

[Upgrading Legacy Mirrors](#)

Legacy Mirror Topics:

[Legacy Mirrors](#)

[RemoteMirrorUnRegister](#)

[RemoteMirrorList](#)

Getting Assistance

Using Help

ENOVIA DesignSync Product Documentation provides information you need to use the products effectively. The Online Help is delivered through WebHelp.

Note:

Use the General Pane to change your default Web browser, as specified during installation.

When Help is open, you can find information in several ways:

- Use the **Contents** tab to see the help topics organized hierarchically.
- Use the **Index** tab to access the keyword index.
- Use the **Search** tab to perform a full-text search.

Within each topic, there are the following navigation buttons:

- **Show** and **Hide**: Clicking these buttons toggles the display of the navigation (left) pane of WebHelp, which contains the Contents, Index, and Search tabs. Hiding the navigation pane gives more screen real estate to the displayed topic. Showing the navigation pane gives you access to the Contents, Index, and Search navigation tools.
- **<<** and **>>**: Clicking these buttons moves you to the previous or next topic in a series within the help system.

You can also use your browser navigation aids, such as the **Back** and **Forward** buttons, to navigate the help system.

Related Topics

Getting a Printable Version of Help

Getting a Printable Version of Help

The *ENOVIA DesignSync CD User's Guide* is available in book format from the ENOVIA Documentation CD or the DSDocumentationPortal_Server installation available on the 3ds support website (<http://media.3ds.com/support/progdir/>). The content of the book is identical to that of the help system. Use the book format when you want to print the documentation, otherwise the help format is recommended so you can take advantage of the extensive hyperlinks available in the DesignSync Help.

You must have Adobe® Acrobat® Reader™ Version 8 or later installed to view the documentation. You can download Acrobat Reader from the Adobe web site.

Contacting ENOVIA

For solutions to technical problems, please use the 3ds web-based support system:

<http://media.3ds.com/support/>

From the 3ds support website, you can access the Knowledge Base, General Issues, Closed Issues, New Product Features and Enhancements, and Q&A's. If you are not able to solve your problem using this information, you can submit a Service Request (SR) that will be answered by an ENOVIA Synchronicity Support Engineer.

If you are not a registered user of the 3ds support site, send email to ENOVIA Customer Support requesting an account for product support:

enovia.matrixone.help@3ds.com

Related Topics

Using Help

Index

A

Access Controls

overview 365

registry keys

Update access control file during
module update
(AddAccessControls) 837

setting up 1

Administration

controlling network communication
timeouts 977

overview 1

registry keys

enable acadmin (ACAdminEnabled)
877

AdvancedFlags 673

Alarm Trigger Maintenance

from site options pane 130

registry keys

alarm trigger frequency
(MaintenanceTimeout) 858

Alias

keyword expansion 130

using DNS aliases 54

Apache

accessing document root 406

configuration files 61

registry keys

apache (httpd) process size
(ProcessSizeLimit) 860

Archive 591

Association

adding 159

Audit Trail Log

registry keys

audit trail log (ACLogDeny) 635

audit trail log maximum file size
(ACDenyLogRotateSize) 635

B

Backups

backing up server 447

defining automatic 458

registry keys

backup frequency
(BackupFrequency) 841

backup logs retention time
(BackupLogRetentionTime) 844

backup retention time
(BackupRetentionTime) 847

backup time, earliest
(BackupEarliestTime) 839

earliest backup time
(BackupEarliestTime) 839

length of time to retain backup logs
(BackupLogRetentionTime) 844

length of time to retain backups
(BackupRetentionTime) 847

number of incremental backups
(BackupNumberOfIncrementals)
846

C

Cache

adding an object 512, 519

cleaning 500

cleaning module cache 516

default cache directory 97

default fetch state 107

determining the correct project cache
507

enabling optimizations 122

excluding IP from cache 510, 517

managing caches 490

moving 500

registry keys

allow non-secure populate
(AllowNonSecureCachePopulate)
868

cache reference counting
(CacheDisableRefCount) 821

cached file uniqueness
(CacheUseHostPost) 703

default cache (WebObjectCache)
714

disable hard links when cache
owner populates the cache
(PBFCAllowUserToOwnFile) 707

project caches
(ProjectCacheTclScript) 709

project caches (Projects) 709

send uncacheable transactions to
the server
(SendUncacheableTransaction)
883

workspace links to cache
(PBFCEnabled) 708

removing an object 512, 519

setting permissions 498

setting up 497

tips for administering 504

versus mirrors 476

Cadence

registry keys

cadence client compatibility mode
setting (ManageViewManifest)
740

cadence design systems'
collections (CadenceView) 734

cadence non-collection members (Cadence View Folder) 736	pre-checkin disk space check (Precheckin) 828
cadence non-view folders in cell directories (Cadence NonView Folder) 733	SCC plug-in auto checkin comment (AutoCheckInComment) 746
cadence view depth (MaxDepth) 741	Checkout
Check In	registry keys
registry keys	checkout options for a specific mirror (CoOptions) 892
cadence auto checkin comment (AutoCheckInComment) 732	checkout options for all mirrors on the MAS (CoOptions) 889
cadence auto-checkin default comment (Cadence\AutoCheckInComment) 732	co options for a specific mirror (CoOptions) 892
check-in comment minimum length (MinCiComment) 692	co options for all mirrors on the MAS (CoOptions) 889
check-in comment require (RequireCiComment) 692	default fetch state (DefaultFetchType) 660
check-in detection of copied workspace (AllowVaultRelocation) 652	fetch read only (FetchReadOnly) 667
checkin error retry attempts (ModuleFailureRetryAttempts) 681	fetching retired data (CoFetchRetired) 656
checkin error retry interval (ModuleFailureRetryInterval) 683	from location default (FetchFromLocal) 665
check-in minimum comment length (RequireCiComment and MinCiCommentLength) 692	Client Triggers
disk space check, pre-checkin (Precheckin) 828	adding 135
	editing 135
	enabling 135
	examples 141
	options 134

- properties 139
- setting up for DesignSync clients 1

Collection

- registry keys
 - collection members (MapMemberOps) 775
 - collection size (Units) 776
 - Synopsys custom designer collections (CDOA) 742

Color

- color selection dialog box 998
- customizing command bar 194
- customizing diff options 198

Columns

- adding new column 217, 218
- customizing 217, 218
 - adding 219
 - customizing column page 196
 - editing 219
- displaying 196

Command

- allowing in GUI 213
- command defaults 102
- customizing 210
 - customizing command bar 194
 - customizing command options 213
 - hiding in GUI 213
 - setting command line defaults 235

Command Triggers

- cancel 258
- ci 259
- co 260
- populate 261
- retire 262
- rmfile 263
- rmfolder 264
- rmvault 265
- rmversion 265
- supported 256, 257
- tag 266
- unlock 269

Compression

- enabling 122
- registry keys
 - data compression (TransferCompressed) 640
- turning off 598

configuring

DesignSync Data Manager Administrator's Guide

- dsdfii 19
- DSMW 19
- Custom Utilities
 - creating 609
 - publishing 615
- D
 - Defaults
 - command defaults 102
 - configuring client default settings 231
 - fetch state 107
 - registry keys
 - save local default (SaveLocal) 697
 - Delta Transfer Compression
 - registry keys
 - delta transfer compression (DeltaTransferCompression) 641
 - delta transfer compression (TransferDelta) 641
- Data Replication
 - introduction 473
 - setting up 478
 - showing hierarchies 487
- Data Replication Root
 - adding data to replicate 483
 - adding root settings 482
 - cleaning data replication 479
 - removing data replication 479
- Data Sheet
 - registry keys
 - data sheet in frame (DataSheetInFrame) 785
- Database Checkpoint
 - registry keys
 - database checkpoint time (DBCheckpointTime) 849
 - database checkpointing occurs (DBCheckpointTime) 849
- Diff
 - annotated format 109
 - choosing color 198
 - customizing output 198
 - format 109
 - registry keys
 - binary compare (Binary) 811
- DesignSync Client
 - running in debug mode 965, 992
- development area
 - enabled shared development areas 835
- Diagnostics Overview 222, 958

- diff format (Diff2Format) 814
- diff options (BlackBg) 812
- diff tool (Command) 813
- diff tool ancestor (Ancestor) 810
- embedded whitespace characters (Embedded) 816
- ignore case (IgnoreCase) 817
- ignore rsc keys (IgnoreKeys) 818
- leading/trailing whitespace characters (IgnoreWhite) 819
- standard format 109
- unified format 109

Disk Space

- registry keys
 - disk quota checking (CheckQuota) 851
 - disk quota checking (DiskLimits) 851
 - disk space check absolute value (FreeSpaceAbsWarn) 894, 897
 - disk space check relative value (FreeSpaceRelWarn) 895, 899
 - disk space required, server (FreeSpaceAbsErr) 852
 - enable general disk space check (Active) 884
 - NFS Disk Quota Check Timeout (NFSTimeout) 856
 - relative value for general disk space check (FreeSpaceRelWarn) 895
 - server disk space required (FreeSpaceAbsErr) 852
 - server disk space required for the postgres database (FreeSpaceRelErr) 854

Display Options 192

Drive Types

- registry keys
 - drive types available (AllowableDriveTypes) 820

DSVS 175

duplicate workspace

- Validate Reference Workspace registry key 830

E

Email

- configuring 383
- how notification works 382
- notifications
 - for RevisionControl notes 403
 - understanding 374
- troubleshooting 977
- upgrading interface 85

Enterprise Design

DesignSync Data Manager Administrator's Guide

automatic synchronization 130

configuring servers 186

synchronization queue 331, 469

Enterprise development

registry keys

 DelegateAC 868

 enabling shared development areas
 835

 EnterpriseUserMapType 869

 UpdateScheme 873

Environment Variables

tab 222

using 231

Errors

accessing or storing to database 963

diagnosing 985

files missing from the cache 964

interrupt button does not seem to
work 967

performance issues with GUI client
967

proxy 978

starting a new browser process each
time help is invoked 967

tagging does not warn of removed
object 968

unable to locate external module
interface command 996

Event Properties

client triggers 134

creating 293

manipulating 294

overview 148, 296

Events

client triggers 145, 289

creating 292

descriptions 145, 289

overview 145, 289

tag operations 145, 289

Exclude

lists 111

objects 111

project-wide 111

registry keys

 custom designer exclude list
 (Exclude) 743

 custom designer exclude proc
 (ExcludeProc) 745

exclude files (UseSyncExclude) 702

exclude lists (SiteFilter and Filter)
699

- exclude warnings (NotifyExcludeByFilter) 687
 - site-wide 111
- exclude files
 - registry keys
 - enable or disable use of exclude files 702
- Expression List 139
- F
 - Fetch State
 - default 107
- File
 - association 159
 - content 157
 - example server-list files 244
 - fetching from the mirror or cache 595
 - fetching to your work area 596
 - linking large files 601
 - missing from the cache 964
 - registry keys
 - enable merged mode (SavePreMergedFile) 712
 - link-in of files to the server (EnableLinkIn) 826
 - mup log files, save (SaveLogFiles) 932
 - retain defaults (FileRetainTime) 669
 - save pre-merged file (SavePreMergedFile) 712
 - revision controlling symbolic links 247
 - used when upgrading legacy mirrors 560
- Filter
 - registry keys
 - CR/LF processing (FilterCrLf) 671
 - fisrvDoesOp optimization
 - enabling 176
 - registry keys
 - fisrvDoseOp optimization (FisrvDoesOp) 738
- Folder
 - initial folder 192
- Font Size 192, 194
- ForbidCmds 673
- FrobidFlags 673
- G
 - GDPR 471
 - General Exclusions 998
- GUI
 - command bar 194
 - displaying commands 210

- look and feel 192
 - options page 190
 - registry keys
 - age of history (Expire) 791
 - always refresh on tree selection (RefreshOnTreeSelect) 801
 - amount of history (RecentMax) 799
 - automatically refresh (AutoRefreshTimer) 777
 - columns (Columns) 781
 - confirm use of 784
 - file editor (FileEditor) 787
 - font size (FontSize) 789
 - GUI and CLI synchronization (SyncGUI) 802
 - GUI display stale icon (ObsoleteTimer) 797
 - GUI font size (FontSize) 789
 - GUI icon size (TreeViewLargelcons) 808
 - GUI web browser definition (Browser) 779
 - icon size (TreeViewLargelcons) 808
 - initial folder (InitialDir, InitialDirSpecify) 793
 - look & feel and metal-style window frames (LookAndFeel, Theme) 795
 - show data sheet in frame (DataSheetInFrame) 785
 - show stale icon (ObsoleteTimer) 797
 - synchronize GUI and CLI (SyncGUI) 802
 - tag values (TagList) 804
 - time format (TimeFormat) 806
 - web browser (Browser) 779
- H
- HCM
- commands 210
 - customizing command options 213
 - options 117
- Help
- contacting ENOVIA 1010
 - hints for using 1009
 - printing 1009
 - registry keys
 - set help mode (HelpMode) 677
- Hide command options 673
- Hide commands in the GUI 673
- Hierarchical References
- changing 572, 721
- Hierarchy

- showing replicated data hierarchy 487
- subscribing to email on a hierarchy 1001
- understanding the custom hierarchy 49

History 195

HTTP

- about 337
- controlling headers 404

I

Icons 190, 192

Installation Settings 224

installing

- DSCD 19
- DSclipse 19
- DSVS 19

installingDesignSync client 19

J

Java Properties 224

K

Keyboard

- shortcuts 207

Keys

- customizing 207

Keystroke 207

Keywords

- expansion 122, 600
- registry keys
 - aliases keyword expansion (EnableAliasesKeyword) 857
 - keyword expansion (EnableKeywordExpansion) 824

L

LDAP

- advanced settings 355
- enabling 353

Legacy Modules

- registry keys
 - enable legacy mode (UseLegacyModules) 730
 - legacy modules without hrefs (FetchModuleNoHrefsAsModule) 719

License

- configuring license server 311
- file 317
- integrating into FlexNet management configuration 314
- management problems and solutions 324
- preparing the license file 310

setting up a license server and default site license 312	metadata lock acquisition (ClientMetadataLockNumTries) 831
setting up a server-specific license 323	orphaned lock timeout (OrphanTimeout) 834
setting up licensing for client-side vault 316	Log File
setting up licensing for products 308	about client log files 959
shutting down the license server 320	about SyncServer log files 971
starting up the license server 320	initial log file 115
understanding product license management 319	registry keys
understanding selection 310	log file size (LogFileMaxSize) 880
using a VendorDaemonName_LICENSE_FILE variable 315	Look and Feel
viewing status of license activity 320	controlling display in GUI 192
Link-In 603	registry keys
List View	look and feel of GUI (LookAndFeel, Theme) 795
customizing 997	Is
Locking	using with external modules 580
registry keys	M
amount of time to wait for metadata lock (ClientMetadataLockTimeout) 832	Maintenance
attempts to acquire metadata lock (ClientMetadataLock) 831	mode 122
	registry keys
	maintenance retry attempts (MaintenanceRetryAttempts) 637
	maintenance retry interval (MaintenanceRetryInterval) 638

PostgreSQL (PGMaintenanceTime) 859	editing 535
postgreSQL maintenance (PGMaintenanceTime) 859	enabling write-through 130
Mapping	finding mirrored data 534
adding 189	legacy 555
editing 189	log files 981
RevisionControl notes 378	overview 524
Metadata	registry keys
registry keys	absolute value for mirror directory disk space check (FreeSpaceAbsWarn) 897
amount of time to wait for metadata lock (ClientMetadataLockTimeout) 832	auto deleting of empty mirrors (AutoDeleteEmptyMirrors) 888
attempts to acquire metadata lock (ClientMetadataLock) 831	check for matching selector (ClientUseMirrorCheck) 926
metadata lock acquisition (ClientMetadataLockNumTries) 831	check tags for mirror update (CheckChangeAffectsMirror) 878
orphaned lock timeout (OrphanTimeout) 834	client write-through (EnableClientMirrorUpdate) 928
Mirrors	control mcache mode option (MADAddMcacheModeServer) 902
adding 548	daily update (NumberHoursToPopulateAllMirrors) 912
administering 541	detect absent files (FindAbsentFiles) 930, 979
architecture of system 527	enable mirror directory disk space check (Active) 886
creating scripted mirrors 532	failure notify interval (MPDNotifyInterval) 881
default fetch state 107	
displaying status 538	

- failure notify interval for mirrors
(MADNotifyInterval) 904
- failure notify time
(MPDPushFailureNotifyTime) 882
- failure notify time for mirrors
(MADUpdateFailureNotifyTime)
905
- failure retries
(MPDPushFailureRetries) 883
- failure retries for mirrors
(MADUpdateFailureRetries) 907
- log file size for mirrors
(LogFileMaxSize) 901
- maximum MUPs running in parallel
(MUPsMaxPerMAD) 908
- mirror administration server update
command
(MUPNewVersForPopulate) 910
- mirror directory free space absolute
value (FreeSpaceAbsWarn) 897
- mirror disk space check enable
(Active) 884
- mirror fetch, secondary mirror
(SMAllowFetchFromRepository)
925
- mirror path resolution
(EnableRealMirrorPaths) 663
- mirror properties upgrade
(PerformUpgrade) 914
- mirror selector match check
(ClientUseMirrorCheck) 926
- populate or checkout a mirror
(MUPNewVersForPopulate) 910
- relative value for mirror directory
disk space check
(FreeSpaceRelWarn) 899
- resolve paths for mirrors
(EnableRealMirrorPaths) 663
- save all mup log files
(SaveLogFiles) 932
- script name (Script) 934
- scrub generated mirrors
(ScrubGeneratedMirrors) 934
- secondary mirror fetch
(SMAllowFetchFromRepository)
925
- set do not cache property on
transaction in the mirror
(IncludeDoNotCacheProperty)
879
- symbolic link handling
(ManagedLinkInMirrorMode) 929
- upgrade mirror properties
(PerformUpgrade) 914
- whether write through occurs
(EnableClientMirrorUpdate) 928
- resetting daemons 555, 984
- scripted 531
- scripts and files used when upgrading
legacy mirrors 560
- setting permissions 554
- setting up a mirror server 544

- troubleshooting the mirror system 980
- upgrading legacy mirrors 556
- using mirrors with modules 533
- using the -h option to specify a unique identifier for a remote mirror 1006
- versus LAN caches 476
- viewing 535

Module

- administration tasks 565
- changing 658
- exporting 568
- external modules 577
 - entobj synchronize 580
 - ls 580
 - populate 580
 - rmmod 580
 - showhrefs 580
 - showstatus 580
 - swap replace 580
 - swap restore 580
 - tag 580
- importing 568
- instance names in the workspace 838
- mirroring 533
 - moving on the server 566

Module Cache

- cleaning 516
- module caches and legacy modules 515
- registry keys
 - do not create filtered links (AllowFilteredLinks) 835
 - ignore invalid module cache paths (IgnoreInvalidPaths) 723
 - link to submodules in the module cache (AllowIntermediateLinks) 717
 - module cache mode (Mode) 725
 - module cache paths (Paths) 727
- setting up 513
- tab 120
- updating 515
- updating legacy module mcache 520

Module Root

- added module roots registry key 716
- adding a module root to the root list 202
- overview 200

Module Views 572

- creating 573

Multithreading Optimization

overview 597

registry keys

multi-threading
(MaxWorkerThreads) 827

O

Objects

registry keys

process locked objects only (IfLock)
679

use checksum (NoChecksumMod)
685

P

Passport

3D 176

Performance

diagnosing 987

optimization overview 595

options 122

tuning 61

Populate

command defaults 102

confirm overwrite option 190

registry keys

allow non-secure cache populate
(AllowNonSecureCachePopulate)
868

direct fetch (EnableDirectFetch)
823

empty directory defaults
(PopulateNoEmptyDirs) 690

ignore locked objects in the mirror
(PopulateIgnoreLockedObjects)
916

mirror daily update
(NumberHoursToPopulateAllMirrors)
912

populate force confirmation
(ConfirmPopForce) 784

populate options for a specific
mirror (PopulateOptions) 920

populate options for all mirrors on
the MAS (PopulateOptions) 918

populate version extended info
(PopulateDoVersionExtendedInfo)
728

using with external modules 580

privacy policy 471

Projects

adding 165

cache 165

defining 169

description 165

determining correct cache 507

- editing 165
- management 9
- options 165
- overview 164
- project-specific configuration 236
- public 169
- registry keys
 - project registry keys (Projects) 709
 - tcl script (ProjectCacheTclScript) 709
- tcl script 507, 709
- vault URL 165

Proxy

- discovery hooks 275, 345
- errors 978
- Header information 337, 338
- HTTP 337
- registry keys
 - HTTP proxies (ProxyNamePort) 639

Q

Queries

- registry keys
 - hierarchical query time-out (TokenLifeTime) 876

R

Refresh

- registry keys
 - tree selection refresh (RefreshOnTreeSelect) 801
- tree selection 190

Registry 95

Registry Files 11, 237, 619

Registry Keys

- AdvancedFlags 673
- alphabetical list 622
- ForbidCmds 673
- ForbidFlags 673
- Simulink 748

Replicate

- general settings 481
- registry keys
 - replicate scrub age (ReplicateScrubAge) 921
 - replicate scrub time (ReplicateScrubTime) 923

Revision Control

- files between Windows and UNIX 241
- symbolic links 160

RevisionControl notes

- attach 169, 367
 - disable 169, 367
 - enable 169, 367
 - enabling note generation 169, 174, 367, 372, 373
 - mapping 378
 - overriding generation 373
 - overview 1, 366
 - registry files 11, 174, 372
 - registry keys
 - RevisionControl note generation (GenerateNote) 874
 - RevisionControl note statistics (CountRevCtlAttachments) 873
 - RevisionControl note statistics (CountRevCtrAttachments) 873
 - setting generation 401
 - setting up triggers 374
 - triggers 169, 367
 - troubleshooting 977
 - using 374
- rmmod
- using with external modules 580
- Root
- running server on ports 71
- show 485
 - show disk usage 489
- S
- SCC plug-in
 - SCC Plug-in Auto-Checkin Default Comment (DSVS\AutoCheckInComment) 746
 - SCC provider 175
 - Scripts
 - samples 586
 - used when upgrading legacy mirrors 560
 - Secure Communications
 - disabling 335
 - enabling 334
 - overview 332
 - setting access controls 333
 - using 336
 - Server
 - backing up 447
 - changing hierarchical references 572
 - configuring to run on root ports 71
 - development 184
 - reducing size of transaction log 467
 - registry files 11

- registry keys
 - send uncachable transactions to the server (SendUncacheableTransaction) 883
- restoring all backup data 452
- restoring vault data 454
- setting up a server-specific license 323
- setting up mirror server 544
- setup scenarios 57
- statistics 169, 367
- using administer panel 329
- using delta transfer hooks to tune client/server communication 272, 598
- showhrefs
 - using with external modules 580
- showstatus
 - using with external modules 580
- Simulink
 - registry keys
 - add 749
 - cancel 761
 - checkin module instance 750
 - Checkin selected 752
 - controlling output display (OutputCmds) 757
 - lock 753
 - mkmod 755
 - overview 748
 - populate 753
 - populate initial 758
 - populate module 759
 - populate module version 765
 - populate revert 761
 - populate selected objects 763
 - remove 767
 - remove using comments 768
 - show status 770
 - tag initial branch 771
 - tag module 773
 - Site Options 130
 - Site-Wide Configuration 235
 - SSL
 - disabling secure communications 335
 - overview of secure communication 332
 - overview of system administration tasks 1
 - using secure communication 336

DesignSync Data Manager Administrator's Guide

Startup	understanding the setting 86
options 127	Swap replace
registry keys	using with external modules 580
automatically log (LogAtStartup) 649	Swap restore
information to log (LogDetailedOutput) 648	using with external modules 580
location of logs (DefaultLogDir) 646	Symbolic Links
name of log file (AutoCreateLogFile and DefaultLogFile) 645	guidelines for creating 246
startup script (RunInitScript and InitScript) 651	handling 245
script 127	registry keys
States	symbolic link handling (SymbolicLinkMode and DirSymbolicLinkMode) 713
controlling cancel 211	symbolic link to mirror handling (AutoUpgradeWorkspaceTo40) 929
controlling check-in 211	revision controlling symbolic links to files 247
controlling check-out 211	revision controlling symbolic links to folders 248
controlling populate 211	
stcl	tab 160
autoloading procedures 240	sync_chown, installing 605
SUID	SyncAdmin
disabling 90	panes
enabling 87	client triggers options 134
enabling for specific mirror or cache directory 89	columns options 196
troubleshooting 91	command bar options 194

- command defaults options 102
- commands 210
- custom tools options 209
- customize diff options 198
- diff format options 109
- display options 192
- environment variables 222
- exclude lists 111
- general options 97
- GUI options 190
- history options 195
- initial folder options 192
- installation settings 224
- java properties 224
- keyboard options 207
- link options 160
- logging options 115
- output pane 194
- performance options 122
- revision control notes options 169, 367
- site options 130
- startup options 127
- tags options 215
- third party integration options 176
- vault types options 180
- tool
 - configuration approaches 231
 - executing
 - from UNIX 14
 - from Windows 16
 - overview 93
 - panes 93
- tool pane
 - adding 997
 - editing 997
- toolbar
 - main 204
- syncdiag Utility 985
- syncmgr 1
- SyncServer
 - administration overview 327
 - aliases 406
 - architecture 7
 - configuring overview 63, 64
 - configuring to run on root port 71
 - considerations before installing 55

DesignSync Data Manager Administrator's Guide

- creating structures for installation 62
 - error log files messages 976
 - list files 242
 - moving 462
 - resetting 328
 - restarting a SyncServer at shutdown or boot time 307
 - running in debug mode 973, 994
 - setting up 328
 - starting 83
 - stopping a SyncServer at shutdown or boot time 307
 - UNIX kernel parameters 60
- T
- Tag 215
 - using with external modules 580
- Tcl
- creating a namespace 220
 - utilities 609
- Temporary Directory (tmp)
- changing 606
 - overview 307
- Third Party Integration
- cadence 176
- DesignSync DFII 176
 - dialog 176
 - fisrvDoesOp 176
 - tab 176
- Time Format 192
- Timeout
- limit transfer timeout 122
 - network communication 977
 - registry keys
 - transfer timeout limit (TransferTimeout) 642
- Timestamp 102
- TkDiff 109
- Tools
- customizing 209
 - installing 19
- Transaction Log
- registry keys
 - transaction log size (TransactionLogRetainRecords) 862
- Triggers
- activating email trigger 395
 - client 134, 141
 - creating 252

- creating note object 407
- customizing email triggers 396
- editing email 394
- editing note object 412
- example client triggers 141, 279
- manipulating 282
- overview 249
- properties 139
- setting note object trigger execution order 416
- setting up Revision Control triggers 374
- transferHook 106
- troubleshooting email triggers 399
- understanding trigger hooks 270
- upgrading email note 85

Troubleshooting

- email notifications of RevisionControl notes 977
- metadata errors 968
- mirror system 980
- nsremote issues 970
- SUID 91

U

- uninstalling
 - DesignSync client 19
 - DSclipse 19
 - Upgrade Considerations 84
 - User Authentication
 - hook 274
 - scripts 351
 - User Profiles
 - changing passwords 348
 - cleaning up references to deleted users 350
 - creating 347
 - deleting 349
 - editing 349
 - environment settings 45
 - overview 346
 - Tcl script for importing users 998

V

- Validate Reference Workspace Registry Key (ValidateReferenceWorkspace) 830
- Vault Browser
 - overview 200
- Vault Types
 - adding 182
 - changing 423

DesignSync Data Manager Administrator's Guide

- editing 182
- overview 417
- pattern 182
- priority 182
- registry keys
 - vault type used for data (VaultType) 864
- setting 419
- tab 180
- Vaults
 - adding association 426
 - client vault fetch state registry key 654
 - converting data 428
 - converting vault repository formats 435
 - exporting 441
 - importing 441
 - moving 442
 - project 164
 - restoring server vault data 454
 - setting up licensing for client-side vault 316
 - using utilities 444
- Versions
 - registry keys
 - keep last-version information (RmVaultKeepVid) 695
 - save local versions (SaveLocal) 697
 - skipping 148, 296
- Visual Studio 175
- W
- Workspace
 - overview 129
 - workspace root path (DefaultAutoRootPath) 704