ENOVIA DesignSync

DesignSync CD User's Reference

3DEXPERIENCE 2022

Table of Contents

Overview

Overview

ENOVIA Synchronicity DesignSync® CD (TM) is the integration of DesignSync® design-management (DM) capabilities into the Synopsys Custom Compiler environment.

ENOVIA Synchronicity DesignSync® CD Capabilities

DSCC installation is described in the Program Directory in the ENOVIA Synchronicity DesignSync Data Manager Installation. DSCC provides all major revision control functionality available in DesignSync including:

- Checking in DesignSync objects
- Checking out DesignSync objects
- Canceling DesignSync checkouts
- Tagging DesignSync objects
- Adding module members to a module
- Branching a DesignSync object
- Viewing the version history of a DesignSync object
- Opening a non-populated version of a View

Using ENOVIA Synchronicity DesignSync Data Manager for Custom Compiler User's Guide Documentation

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

- Integrated help - When you press F1 within the Custom Compiler application or click on the Help button on DesignSync dialog boxes, the appropriate help 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 for DSCC User'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

You might need to refer to the following guides if you are learning to use the ENOVIA Synchroncity DesignSync CD product.

ENOVIA Synchronicity DesignSync Data Manager User's Guide	Describes the concepts and workflow for DesignSync in detail.
ENOVIA Synchronicity DesignSync Administrator's Guide	Describes the customizations available to optimize performance and usability and allows you to enable DFII.

Getting Started with DSCC

Your design data (libraries, cells, cell views) are stored in a DesignSync vault (project data repository), which can be shared with other users. To work on a design, you check out files from the vault into your workspace. When you have completed your changes, you check the files back into the vault.

DesignSync can operate on data stored in a "FileSys" disk structure. It cannot operate on data stored in "Turbo" disk structure.

DSCC manages cell views as single revision-controllable objects called collection objects, which ensure data integrity. When you operate on cell views, DSCC outputs status messages using a cell-view naming convention instead of listing all the files that make up the cell view. This naming convention is **<viewname>.sync.cdoa**, where **<viewname>** is the name of the cell view folder on your file system. DSCC uses a simple viewing scheme so that only the value of **<viewame>** is displayed in the interface.

DSCC manages category files as a single file containing the list of cells/files within that category. Category files are able to be versioned controlled by DSCC.

Notes:

- In order to take advantage of the capabilities of DSCC, you must have a DSCC license. For information about obtaining a DSCC license, speak to your customer service representative. For information about configuring licensing, see the *DesignSync Data Manager System Administor's Guide*: Setting Up Licensing for DesignSync Products.
- DSCC operations can be access controlled. An operation initiated from DSCC will fail if you have been denied access to that operation. See the *Synchronicity Access Control Guide* for more information.
- The *DesignSync Data Manager CD User's Guide* assumes that DSCC has not been customized. You may observe differences in your environment from what is described here if you or your project leader have enabled any customizations.
- The *DesignSync Data Manager CD User's Guide* describes capabilities that are specific to DSCC. Some general DM concepts and DesignSync capabilities are

not documented here. See *DesignSync Data Manager User's Guide* for more information. Also, use Synopsys's documentation for information about the Custom Compiler interface.

Setting up your Environment

Important: If your system is configured to support both Cadence Design System' tools and data and Synopsys' Custom Compiler tools and data, your collection may be incorrectly identified as DSCC information. For more information, see *DesignSync Data Manager DFII User's Guide:* Specifying DesignSync DFII as Your Design Management System.

Installing the system icons

DesignSync CD requires access to a set of predefined icons that visually represent the collections and revision control states in the Custom Compiler interface. The installation procedure for these icons is described in the *ENOVIA Synchronicity DesignSync Data Manager Installation*.

All users must have their .cdesigner.tcl files set up to start the DSCC integration. For more information, see the *ENOVIA Synchronicity DesignSync Data Manager Installation*.

Using DesignSync

When using DesignSync as the revision control system for Custom Compiler, you may need to set DesignSync as the VC for any libraries managed by DesignSync.

Setting up DesignSync as the VC type for a Library.

From the Library manager console, type:
dm::associateVersionControl "DesignSync" -libName <*LibName*>

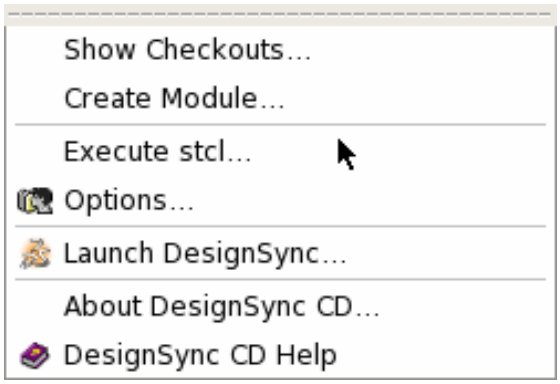
DesignSync Menu Bar

This topic describes the various menus and menu commands that are specific to DesignSync.

Custom Compiler Console

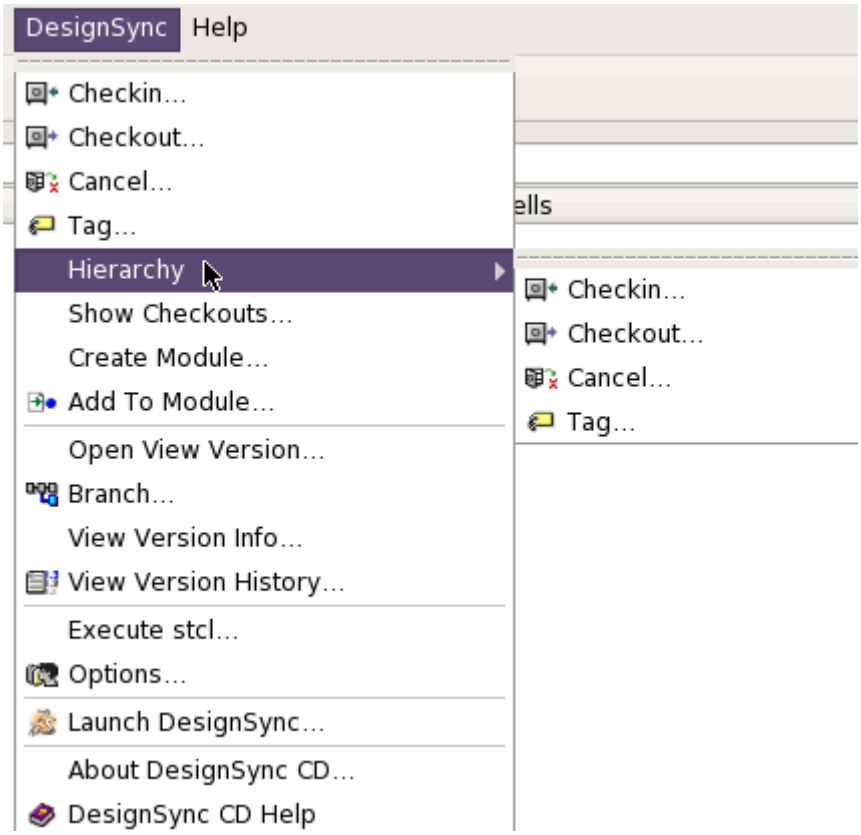
The DesignSync menu on the Custom Compiler Console contains a subset of the available DesignSync commands.

Click on the elements in the DesignSync Menu bar to jump to their description.



Editor Window

The DesignSync menu available from the editor contains the following fields:



Checkin

See Checkin.

Checkout

See Checkout.

Cancel

See Cancel.

Tag

See Tag.

Show Checkouts

See Show Checkouts.

Create module

See Create Module.

Add to module

See Add to Module.

Open View Version

See Open View Version.

Branch

See Branch.

View Version Info

See View Version Info.

View Version History

See View Version History.

Execute stcl

See Execute stcl.

Options

See Options.

Launch DesignSync

Launches the DesignSync GUI client with client focus on the current working directory (cwd).

About DesignSync CD

Shows the About box for DesignSync CD, containing contact and version information for the product.

DesignSync CD Help

Launches the help in your default browser starting on the Overview page.

Using

Checking in Design Data

This task shows you how to check in design data. To preserve edits you've made, you must check your files into source control. DesignSync supports checking in libraries, category files, cells, and cellviews.

Note: Before you check a library into Custom Compiler for the first time, make sure the VC type for the library is set to DesignSync. For more information see Setting up your Environment: Using DesignSync.

Prerequisites

- The DesignSync Server must be running
- You must have defined a connection between the workspace folder and the DesignSync server, for example by populating a module into a workspace or setting a vault association on the workspace directory.

Checking in DesignSync data

1. From the **DesignSync** menu, select **Checkin..**
   or From the **DesignSync** menu, select **Hierarchy** | **Checkin...**
   The Checkin dialog displays.
2. Select the desired checkin options and press **Ok** to begin the checkin.
   The output of the Checkin operation is displayed.

Checking out Design Data

This task shows you how to check out design data, updating the objects in your workspace with the server version. DesignSync supports managing libraries, category files, cells, and cellviews.

The checkout operation is module-aware and can operate on an entire module, recursively within folders, or on selected files.

Prerequisites

- The DesignSync Server must be running
- You must have defined a connection between the workspace folder and the DesignSync server, for example by populating a module into a workspace or setting a vault association on the workspace directory.
- The data must have previously been checked in.

Checking out DesignSync data

1. From the **DesignSync** menu, select **Checkout.**
   or From the **DesignSync** menu, select **Hierarchy** | **Checkout...**
   The Checkout dialog displays.
2. Select the desired options and press **Ok** to begin the checkout.
   The output of the Checkout operation is displayed.

Canceling Checkout of Design Data

This task shows you how to cancel a checkout of design data, removing a checkout lock and optionally replacing the version in your workspace with the latest server version.

Prerequisites

- The DesignSync Server must be running
- You must have defined a connection between the workspace folder and the DesignSync server, for example by populating a module into a workspace or setting a vault association on the workspace directory.
- The server data must have been checked out with a lock.

Cancel the checkout of DesignSync data

1. From the **DesignSync** menu, select **Cancel Checkout..**
   or From the **DesignSync** menu, select **Hierarchy** | **Cancel Checkout...**
   The Cancel Checkout dialog displays.
2. Select the desired options and press **Ok** to begin the checkout cancelation.
   The output of the Cancel Checkout operation is displayed.

Tagging Design Data

This task shows you how to tag design data. Tagging is the application of a symbolic name, called a tag, to a version or a branch. Tags can only be applied to objects that are under revision control. The tag operation tags versions or branches of objects in the vault, not the local copies of objects in your work area. Although tags reside on object versions in the vault, the tag operation uses the object versions in your work area to determine the versions to tag in the vault. For more information on the concept of tagging object versions or branches, naming conventions for tags, and tagging module member versions (snapshots) see the DesignSync Help: Tagging Versions and Branches.

Prerequisites

- The DesignSync Server must be running

Tag DesignSync data

1. Select an object and launch the context menu or from the **DesignSync** menu, select **Tag...**
   The Tag dialog displays.
2. Select the desired options and press **Ok** to begin the tag operation.
   The output of the Tag operation is displayed.

Note: Tag can also be performed as part of the Checkin operation.

Creating a Module

The Create Module action is used to create a new module. You can run the Create Module action to create a new module after selecting a single, unmanaged library.

Prerequisites

- The DesignSync Server must be running
- You must select a single unmanaged library.

Creating a Module

1. From the **DesignSync** menu, select **Create Module..**
   The Create Module dialog displays.
2. Enter the module information or click Browse to launch the Server Browser which allows you a graphical interface to find the correct server and module category for the new module. After you have selected the options (and returned to the Create Module dialog, if applicable), press **Ok** to create the module and perform the initial checkin for the required files within the selected library; the `data.dm` and `oalib` files.

Running DesignSync Commands with the Exec Stcl Interface

To provide complete access to the DesignSync command set, the Custom Compiler features a Tcl command window. Using this command window, you can execute any DesignSync command from within the Custom Compiler.

Notes: Although this is functionally a concurrent stcl (stclc) interface, you must use the commands as they would appear in a script, since this dialog does not provide all the advantages of the stclc command-line interface.

To open the Stcl window:

1. Select **DesignSync => Execute Stcl...** from the menu.
2. Enter the command. The output is displayed in the Custom Compiler Console window.
3. Click **OK**, **Apply** or press the **RETURN** key to execute the command.
4. Click **Cancel** to close the window.

Related Topics

Execute stcl

Options

Moving and Renaming Objects

You can move, rename, or relocate objects using the built-in **Move...** command. When moving or renaming revision controlled objects, the original object is retired from the vault or removed from the module. The history of the object is retained on the original object. Moving and Rename objects uses the underlying Copying Objects functionality to copy the original object.

Note: Moving or renaming all of the objects in a module does not remove the module. Even if all of the contents of the module is moved, the original module is retained. If there are hierarchical references to the module, they remain with the original module versions and need to be manually updated.

If the object is moved or renamed to a managed object in DesignSync, the destination object must be checked out with lock. The resulting object is updated in the local workspace and contains the appropriate object metadata, but requires a checkin to server to commit the changes and remove the lock.

Note: You cannot move, copy, or rename a library to a different managed library in DesignSync.

Because category files can also be managed objects, when you move a cell, you should check out the category file as well.

Note: Before you check a library into Custom Compiler for the first time, make sure the VC type for the library is set to DesignSync. For more information see Setting up your Environment. After a library is moved, the new library will need to be re-associated with the DesignSync Version Control system.

For more information on moving or renaming objects, see the Custom Compiler help.

To reassociate a moved library with DesignSync:

From the Library manager console, type:

```
dm::associateVersionControl DesignSync -libName <new-
library-name>
```

Related Topics

Copying Objects

Understanding and Resolving Cellview Conflicts

Copying Objects

You can copy cells and cellviews using the built-in **Copy...** command. If the destination is managed in DesignSync, the destination object must be locked in DesignSync. The copy functionality is closely related to Moving and Renaming Objects, the primary difference being that in a copy operation, the original object is not removed. If the destination is not managed in DesignSync or does not exist, the new object is unmanaged and must be added to code management as a new file.

For more information on copying objects, see the Custom Compiler help.

Opening a View Version

You can display cell view versions other than the version in your workspace. For example, you might want to compare two or more versions of the same cell view.

To open a cell view version:

1. From the **DesignSync** menu, select **Open View Version**.
2. Modify the fields of the Open Version form as needed.

Viewing Version Information

View Version Info displays the list of versions for the selected cell view and performs revision control operations on a selected version in the list.

To display version status:

1. Select the desired object to view then, from the **DesignSync** menu, select **View Version Info.**

   The View Version Info form appears.

2. Modify the fields of the View Version form as needed.

Deleting Temporary Views

Temporary views are copies of managed views that are created when you use Open View Version command. These versions persist so they can be reused, but over time, they can accumulate and outlive their usefulness. When that happens, you can delete all or specified temporary views. You delete temporary views on a per library basis.

Prerequisites

- The DesignSync Server must be running
- You must have defined a connection between the workspace folder and the DesignSync server, for example by populating a module into a workspace or setting a vault association on the workspace directory.
- The data must have previously been checked in and you must have opened view versions.

To Delete Temporary Views:

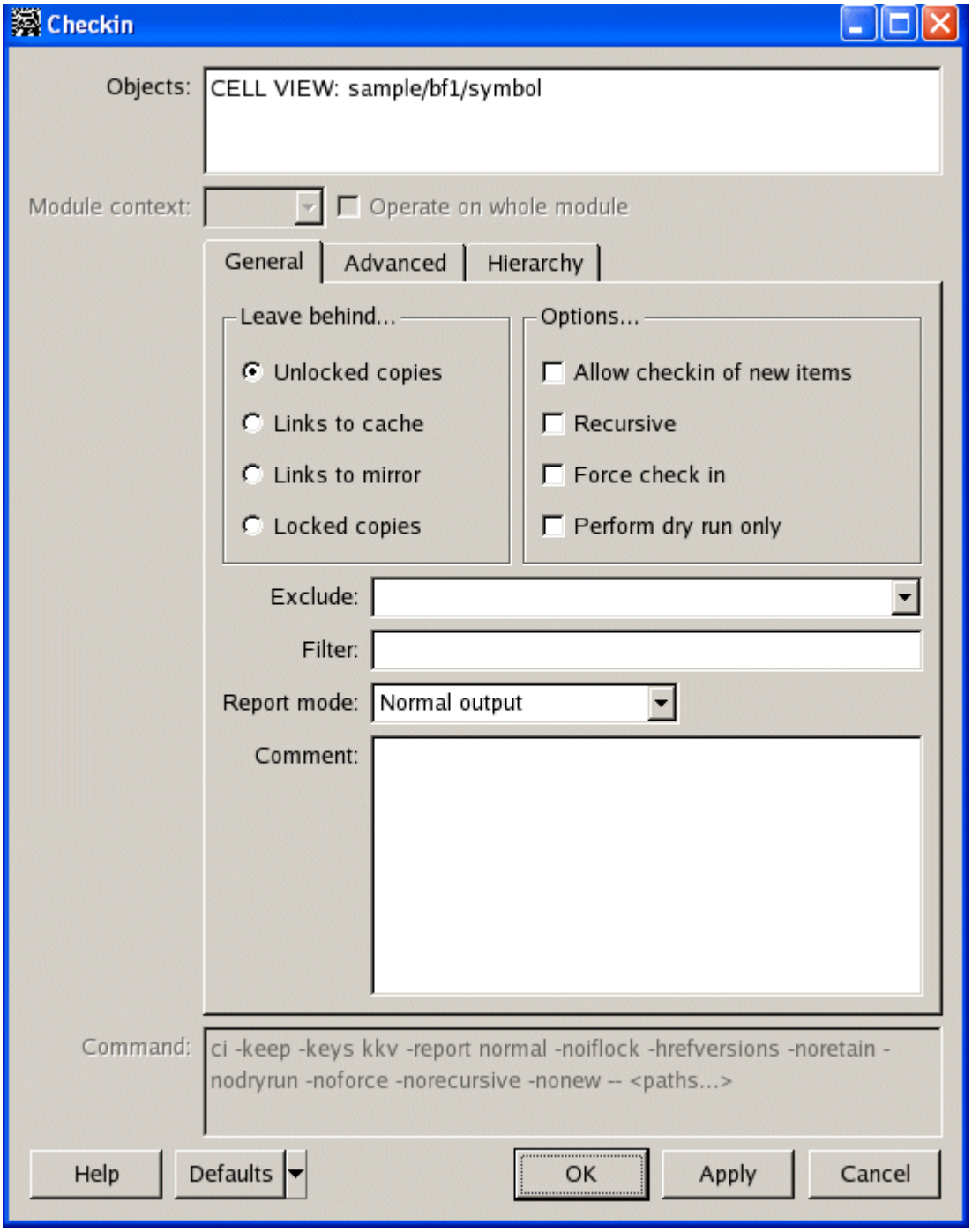
1. Select the desired library to remove the temporary views from.
2. Launch the context menu and select Delete Temporary Views. This opens the Delete Temporary Views Dialog box.
3. Select one or more of the views you want to remove or select **All** to select all views.
4. Click **OK** to remove the temporary views.

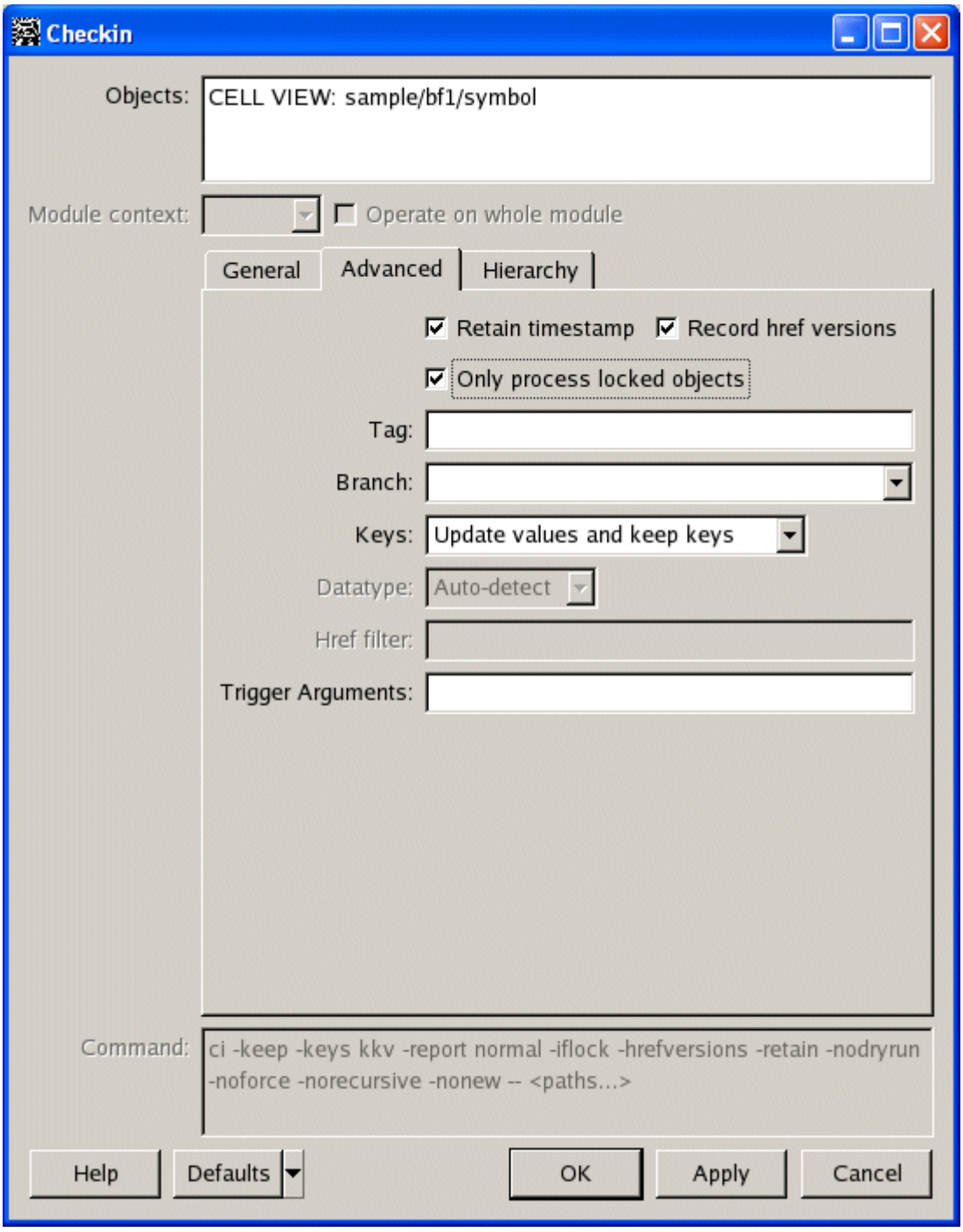
Interface Description

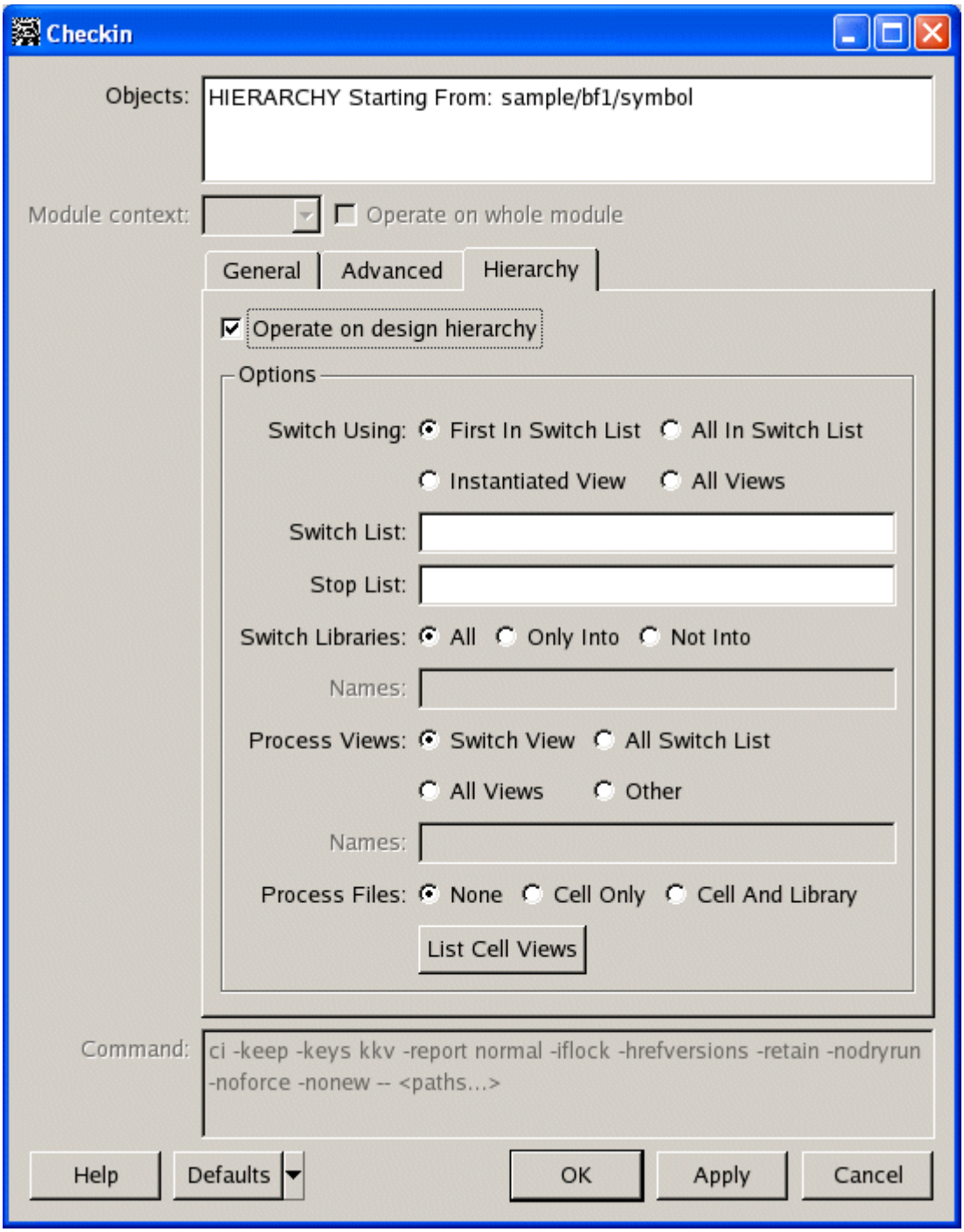
Checkin

This topic describes the Checkin dialog box, which appears when you select Checkin.

Click on the fields in the following illustrations for information.




Checkin
Objects:
CELL VIEW: sample/bf1/symbol
Module context:
Operate on whole module
General
Advanced
Hierarchy
Retain timestamp
Record href versions
Only process locked objects
Tag:
Branch:
Keys:
Update values and keep keys
Datatype:
Auto-detect
Href filter:
Trigger Arguments:
Command:
ci -keep -keys kkv -report normal -iflock -hrefversions -retain -nodryrun -noforce -norecursive -nonew -- <paths...>
Help
Defaults
OK
Apply
Cancel



Objects

Lists the objects selected for checkin. The object field is read-only, and, for a hierarchy, begins at the selected object.

Module context

See Module Context Field.

Operate on whole module

If a module object is selected for the operation, determines whether to operate on the whole module or only the selected objects in the Objects field.

This option is inactive when a hierarchy operation is being performed. This option does not appear when a library object is selected. Smart module detection determines if the entire library is within a module and whether the operation should be run in a module centric or folder centric way.

Leave behind Unlocked copies

After the operation is over, keep an unlocked copy in your work area. This is the default unless your project leader has defined a default fetch state.

Because you have relinquished any lock you may have had on the file, someone else can check the file out from the vault with lock in order to modify it.

Leave behind Links to cache

Use this option to link to a shared copy of the design object in a cache directory.

Leave behind Links to mirror

Use this option to link to a shared copy of the design object in a mirror directory. This option is not available for module data.

Leave behind Locked copies

After the operation is over, keep a locked copy in your work area. You can continue to work on the file. Others cannot check in new versions of the object as long as you have the branch locked. (The copy in your work area is known as an original.)

Allow check in of new items

You must select this option if you are checking in an object that has never been checked in before or, in the case of module data, was neither checked in before, nor added to a module. When you check in the object, DesignSync creates a vault for it. This is the vault from which subsequent versions of the object are checked in and out. Once the vault has been created for the object, you do not need to select this option during checkins.

Note that selecting this option also lets you check an object onto a retired branch. The branch is unretired and a new version of the object is created. The retire information is removed from the object history.

If the unmanaged object is being added to a module, then either a **Module context** must be specified, or there must be only one module in the workspace (so that DesignSync can determine the module context). If a **Module context** is specified with a folder object, DesignSync performs the checkin in a module-centric way, checking in any unmanaged objects into the specified module. If the Recursive option is also specified, it is ignored and the hierarchical references are not followed.

You may check in unmanaged objects to a new branch using the **Branch** option for DesignSync vault objects only. You cannot check in unmanaged objects to a new module branch; you must first Add them to the module. For more information, see the Branch option.

Note: When checking in module data, you cannot specify the **Recursive** or **Branch** options with the **Allow check in of new items** option.

Force check in

If you check out a file, make no changes to it, and then attempt to check it in, DesignSync informs you that it will not check the file in. If you want to check the file in anyway, you must select this check box. The file will be checked in and a new version created that is identical to the version already in the vault.

Note that you must have a local copy of the file in your working directory for a new version to be created. A new version is not created if the object does not exist or is a reference.

In most cases, not being allowed to check in an unchanged file is reasonable. One reason you may wish to use this option is to keep version numbers synchronized.

Perform dry run only

Select this option to indicate that DesignSync is to treat the operation as a trial run without actually checking in design objects. For module data, module hierarchy processing is included in the output.

This option helps detect whether there are problems that might prevent the checkin from succeeding. Because file and vault states are not changed, a successful dry run does not guarantee a successful checkin. Errors that can be detected without state changes, such as a vault or branch not existing, merge conflicts, or a branch being locked by another user, are reported. Errors such as permissions or access rights violations are not reported by a dry run. Note that a dry run checkin is significantly faster than a normal checkin.

Exclude

See Exclude Field.

Filter

See Filter Field.

Report mode

For the **Report mode**, choose the level of information to be reported:

- **Brief output:** Brief output mode reports the following information:
    - Failure messages.
    - Warning messages.
    - Informational messages concerning the level of the hierarchy being processed.
    - Success/failure status.
- **Normal output:** (Default) In addition to the information reported in **Brief mode**:
    - Informational messages for updated objects.
    - Information about all objects processed.
- **Verbose output:** In addition to the information reported in **Normal mode:**
    - Informational message for every object examined but not updated.
    - Information about all filtered objects.
- **Errors and Warnings only:** Errors and Warnings output mode reports the following information:
    - Failure messages.
    - Warning messages.
    - Success/failure status messages

Comment

See Comment Field.

Retain Timestamp

See Retain Timestamp Field.

Record href versions

Specifies whether to update the static hierarchical references associated with the module.

When a module is checked in (either an entire module or any of its contents), DesignSync captures the currently populated versions of the module's hierarchically referenced sub-modules, and records those as part of the next module version, updating the static hierarchical references. (Default).

When this option is not selected and a module is checked in, the module members are checked in, but the hierarchical references are ignored (not updated). This is particularly useful if you have out-of-date submodules, or submodule changes that are not ready to be checked in.

This option is ignored for non-module data.

If the **Recursive option** is selected, and the checkin is operating in a **Module context**, then this option cannot be unselected.

Only process locked objects

Specifies whether to check in all modified objects in the workspace or only targeted files. Changes that are targeted (or locked) are:

- Locked DesignSync vault files or module members.
- Objects that have been added to a module.
- Module members that have been renamed or removed since the last module checkin.

Only process locked objects is mutually exclusive with the **Allow checkin of new items** options.

Tag

Tags the object version or module version being checked in with the specified tag name.

For module objects, all objects are evaluated before the checkin begins. If the objects cannot be tagged, for example if the user does not have access to add a tag or because the tag exists and is immutable, the entire checkin fails.

For other DesignSync objects, if the user does not have access to add a tag, the object is checked in without a tag.

For more information on the access control for the tag command, see ENOVIA Synchronicity Access Control Guide: Access Controls for Tagging. For more information on branch and version tags, see Tagging Versions and Branches.

Notes:

- If both a tag and a comment are specified for a module version or branch checkin, the comment is also used as the tag comment.
- You cannot tag modules stored on DesignSync server versions prior to 5.1.

Branch

Specify a branch in the **Branch** field to check in to a branch other than the one from which you checked the object out. The **Branch** field has a pull-down menu from which you can query for existing branches. See Suggested Branches, Versions, and Tags for details.

Note: The **Branch** field accepts a branch tag, a version tag, a single auto-branch selector tag, or a branch numeric. It does not accept a selector or selector list. To branch a module, you must specify a branch tag that does not already exist.

When branching a module, you must create a new branch. You cannot specify an existing branch. The module branch checkin creates the new module branch version with the following module member objects:

- Added objects that have not been checked in yet.
- Modified objects belonging to the specified module.
- Unmodified objects belonging to the specified module.
- Objects that are part of the module on the server, but have not been populated into the workspace.
- Objects in the workspace that were removed on the server in a later module version.

Note: The module member version in the workspace is always considered the desired version for the ci -branch operation. If you have older member versions in the workspace, those will become the Latest version on the new branch.

When you check a module into the new branch, DesignSync automatically modifies the workspace selector to the Latest version of the new branch tag (<Branch>:Latest).

The **Branch** option, when specified with a module, is mutually exclusive with **Recursive** and **Allow check in of new items**.

Keys

See Keys Field.

Datatype

Specify the data type for any module object or any new vault object being checked in.

- **Auto-detect** uses a built-in algorithm to determine whether the object contains only ASCII text or a binary file. (Default)
- **ASCII** creates the new object with a vault data type of ascii.
- **Binary** creates the new object with a vault data type of binary. Binary objects cannot be merged, they can only be replaced. ZIP vaults are always checked in using binary mode, regardless of whether the vault's data type is designated as ascii.

Note: To change the data type of an existing vault object, use the url setprop command. For more information, see ENOVIA Synchronicity Command Reference: url setprop. For module objects, you can set the data type when you check in a new module version.

Href filter

See Href Filter Field

Trigger Arguments

See Trigger Arguments Field.

Operate on Design Hierarchy

See Operate on Design Hierarchy.

Switch Using

See Switch Using Field.

Switch List

See Switch List Field.

Stop List

See Stop List Field.

Switch Libraries

See Switch Libraries Field.

Names

See Library Name Field.

Process Views

See Process Views Field.

Names

See Process Names Field.

Process Files

See Process Files Field.

List Cell Views

See Hierarchy List.

Command Invocation

See Command Invocation Field.

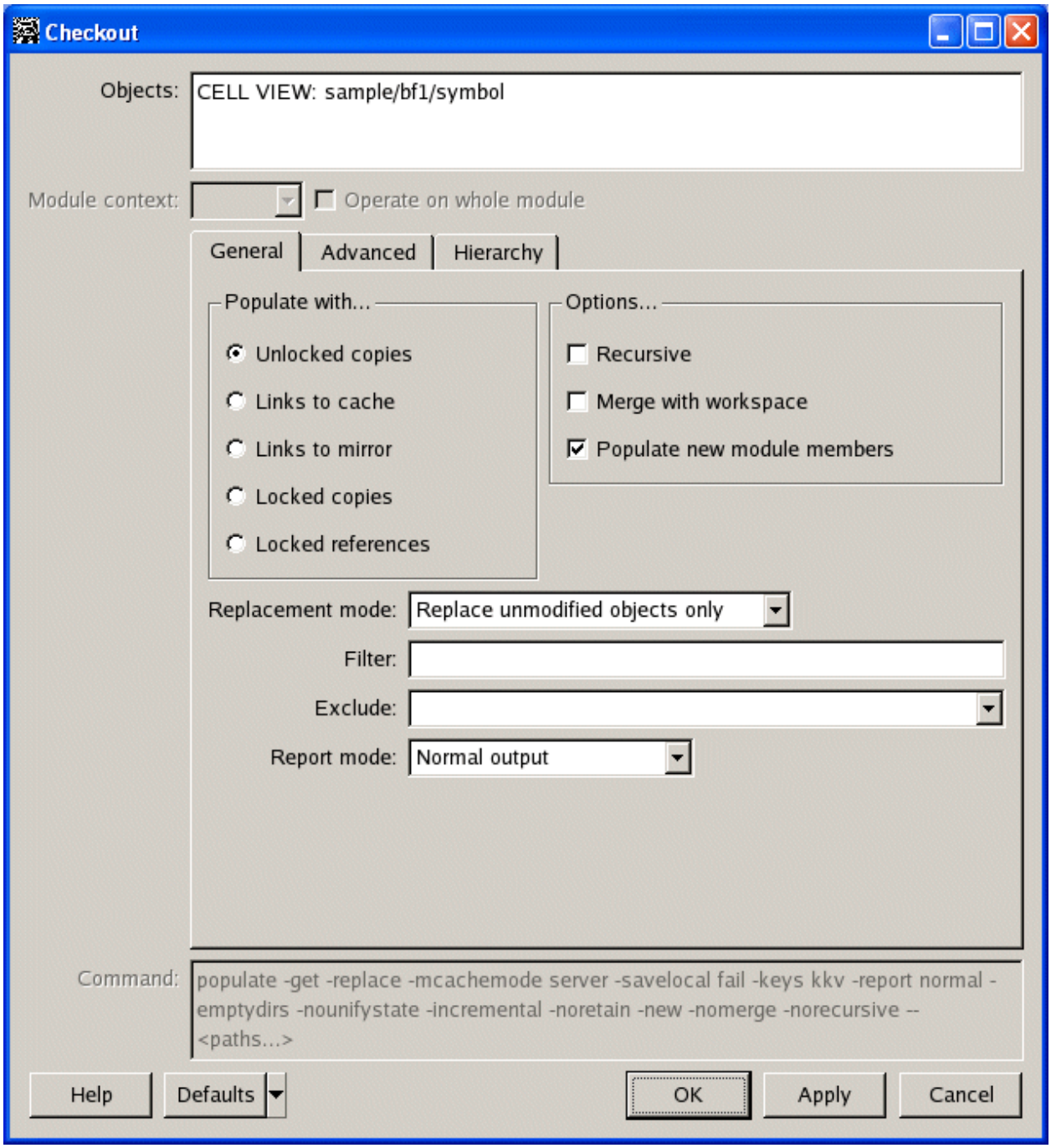
Command Buttons

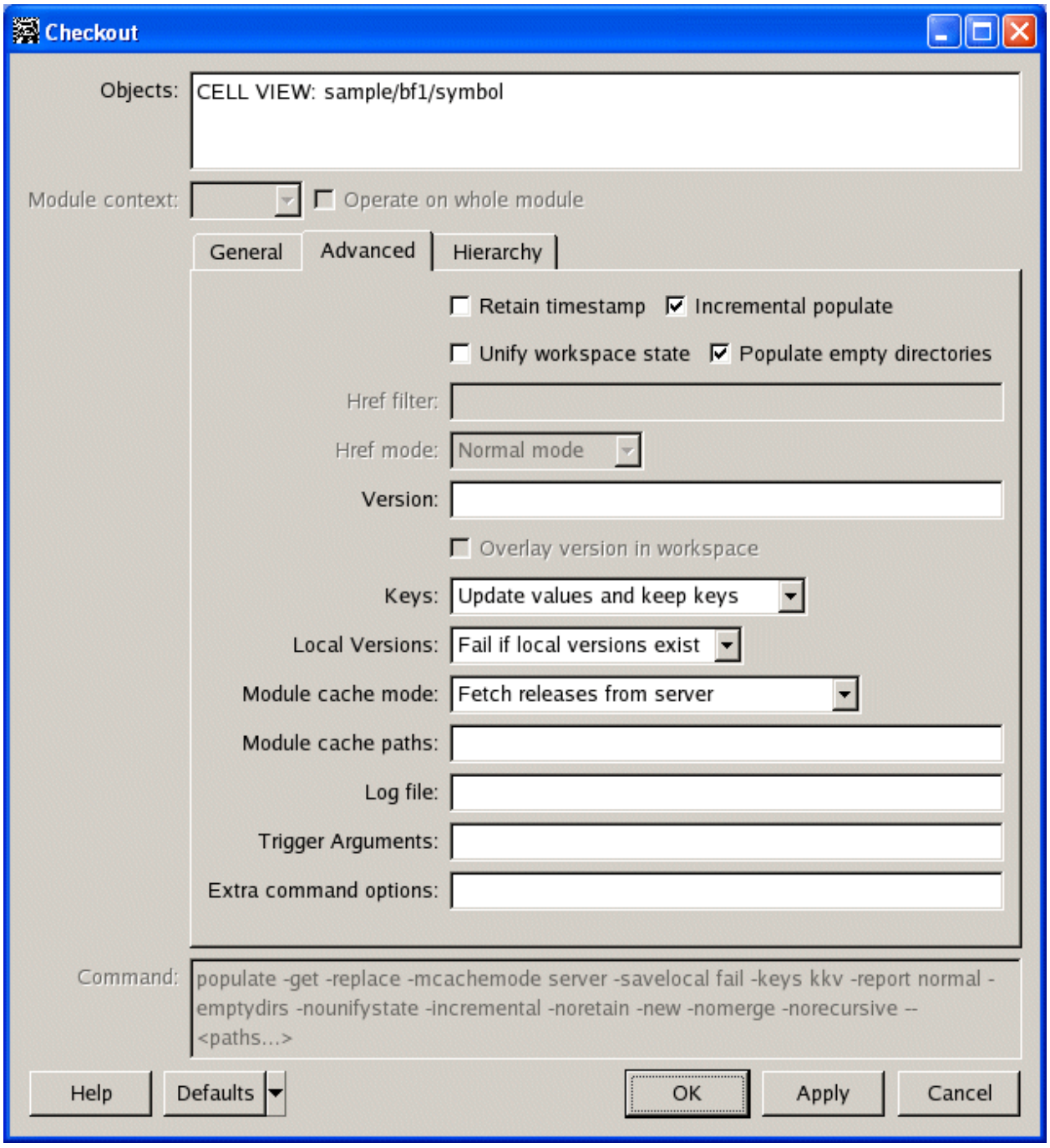
See Command Buttons.

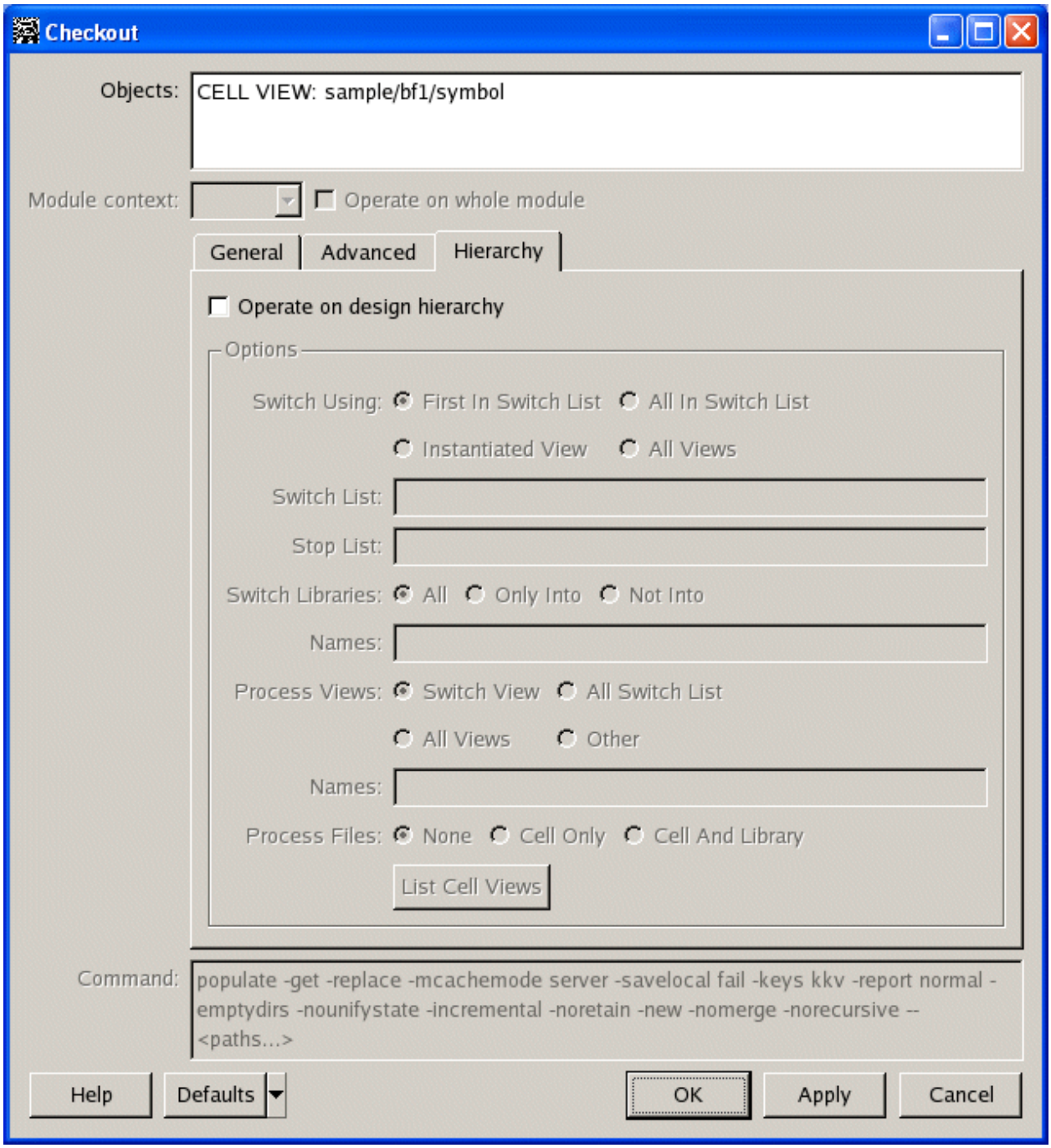
Checkout

The check-out commands let you get the latest version of a design object from the vault for editing or viewing. You also can check out older versions for viewing.

Click on the fields in the following illustrations for information.


Checkout
Objects:
CELL VIEW: sample/bf1/symbol
Module context:
Operate on whole module
General
Advanced
Hierarchy
Populate with...
Unlocked copies
Links to cache
Links to mirror
Locked copies
Locked references
Options...
Recursive
Merge with workspace
Populate new module members
Replacement mode:
Replace unmodified objects only
Filter:
Exclude:
Report mode:
Normal output
Command:
populate -get -replace -mcachemode server -savelocal fail -keys kkv -report normal -emptydirs -nounifystate -incremental -noretain -new -nomerge -norecursive -- <paths...>
Help
Defaults
OK
Apply
Cancel


Checkout
Objects:
CELL VIEW: sample/bf1/symbol
Module context:
Operate on whole module
General
Advanced
Hierarchy
Retain timestamp
Incremental populate
Unify workspace state
Populate empty directories
Href filter:
Href mode:
Normal mode
Version:
Overlay version in workspace
Keys:
Update values and keep keys
Local Versions:
Fail if local versions exist
Module cache mode:
Fetch releases from server
Module cache paths:
Log file:
Trigger Arguments:
Extra command options:
Command:
populate -get -replace -mcachemode server -savelocal fail -keys kkv -report normal -emptydirs -nounifystate -incremental -noretain -new -nomerge -norecursive -- <paths...>
Help
Defaults
OK
Apply
Cancel



Objects

Lists the objects selected for checkout.

Module context

See Module Context Field.

Operate on whole module

If a module object is selected for the operation, determines whether to operate on the whole module or only the selected objects in the Objects field.

This option is inactive when a hierarchy operation is being performed. This option is not displayed when a whole library is selected for checkout. Smart module detection determines if the entire library is within a module and whether the operation should be run in a module centric or folder centric way.

Populate with Unlocked copies

After the operation is over, keep an unlocked copy in your work area. This is the default unless your project leader has defined a default fetch state.

Because you have relinquished any lock you may have had on the object, someone else can check the object out from the vault with a lock in order to modify it.

The **Unlocked files are checked out in Read Only mode** option set on the SyncAdmin: General Tab determines whether these files are read/write or read-only.

Populate with Locked copies

After the operation is over, keep a locked copy in your work area. You can continue to work on the object. Others cannot check in new versions of the object as long as you have the branch locked. (The copy in your work area is known as an original.) This option is not available for legacy modules or legacy module configurations. When operating on 5.0 modules, it is module members that are locked; not whole modules. Thus, this option is mutually exclusive with the **Recursive** option, or with a **Module context** specified.

Populate with References to versions

This option lets you acquire references to the versions you have selected.

Tip: Do not use this option when operating on a collection object. If you use this option, DesignSync creates a reference in the metadata for the collection object, but member files are not processed and are not included in the metadata.

Populate with Links to cache

Use this option to link to a shared copy of the design object in a cache directory. This option is available only on UNIX platforms.

Populate with Links to mirror

Use this option to link to a shared copy of the design object in a mirror directory. This option is available only on UNIX platforms. This option is not available for module data.

Populate with Locked references

This option lets you acquire a locked reference. If you intend to regenerate the object, create a locked reference to avoid fetching a copy of the object from the vault. This option is not available for legacy modules or legacy module configurations. For other module data, this option is mutually exclusive with the **Recursive** option, or with a **Module context** specified.

Merge with workspace

Select this option if you want to merge the Latest version of an object in the vault with a locally modified version. This option supports the merging work style where multiple team members can check out the Latest version of an object for editing. The first team member to check his changes in creates the next version; other team members *merge* their local changes with the new Latest version, and then check the merged version in.

If there are no conflicts, then the merge succeeds, leaving you the merged file in your work area. If there are conflicts, a warning message results. You must edit the merged file to resolve the conflicts before DesignSync allows you to check in the merged version. Conflicts are shown as follows:

```
<<<<<<< local

Lines from locally modified version

=======

Lines from Latest version

>>>>>>> versionID
```

The conflicts are considered resolved when the file no longer contains any of the conflict delimiters (7 less-than, greater-than, or equal signs in the first column).

Populate new module members

This option only applies to module data. If selected, the populate operation fetches objects that are in the vault, but not currently in the workspace (subject to the **Filter** and **Exclude** fields). This is the default behavior. If unselected, the populate operation updates only objects already in the workspace.

Replacement mode

The **Replacement mode** determines how the populate operation updates your work area with the data you are fetching. Specify one of the following update methods:

- **Do not replace any objects**. For DesignSync data, do not overwrite locally modified files. Do not remove files that do match the requested version.

  For module data, preserve local modifications you made to module members, and leave intact any module members that are in your work area that are not in the requested module version.
- This mode causes the least disruption to your work area; however, it may require you to clean up resulting work area data.
- **Replace unmodified objects only.** For DesignSync data update unmodified objects with the current server version and remove any unmodified objects that are no longer present in the vault, for example, if a file was retired.

  Note: Objects that have been retired, but remain in the workspace or were re-created in the workspace are considered locally modified.

  For module data, update module members that have not been locally modified and that are part of the requested module version. Also remove any unmodified module members that are not part of the requested module version.

- This mode, which is the default behavior for module data, leaves intact any module members you have modified in your workspace.
- **Force overwrite of modified objects.** For DesignSync data, overwrite locally modified files. Also remove managed objects that do not match the requested selector.

  For module data, replace or remove module members, regardless of whether you have modified them locally or whether they are part of the requested module version.
- The intent of this mode is to make the workspace match the data being requested (subject to the **Filter** and **Exclude** fields), as closely as possible. Unmanaged data is never removed.

  This mode removes items that have been added to a module but have not yet been checked in; their "added" indicator is removed. The data, which is unmanaged, is not removed. See Adding to Module for information adding members to modules.

  If there is a conflict with data fetched from another module, that other data is not removed.
  If populating a module overlaps with another module already in your workspace, data from that other module is not removed.

  This mode is mutually exclusive with the **Merge with workspace** option.

The **Replacement mode** only applies to items that are not filtered out by the **Href filter**, **Filter** or **Exclude** options.

Filter

See Filter Field.

Exclude

See Exclude Field.

Module context

See Module Context Field.

Report mode

For the **Report mode**, choose the level of information to be reported:

- **Brief output:** Brief output mode reports the following information:
    - Failure messages.
    - Warning messages.
    - Version of each module processed.
    - Creation message for any new hierarchical reference populated as a result of a recursive module populate.
    - Removal message for any hierarchical reference. removed as part of a recursive module populate.
    - Success/failure status.

- **Normal output:** In addition to the information reported in Brief:
    - Informational messages for objects that are successfully updated by the populate operation.
    - Messages for objects excluded from the operation (due to exclusion filters or explicit exclusions).
    - Information about all fetched objects.

- **Verbose output:**In addition to the information reported in **Normal mode:**
    - Informational message for every object even if it is not updated, for example objects that are skipped because the version in the workspace is the current version.
    - For module data, also outputs information about all objects that are filtered.

- **Errors and Warnings only**: Errors and Warnings output mode reports the following information:
    - Failure messages.
    - Warning messages.
    - Success/failure status messages.

Incremental populate

Perform a fast populate operation by updating only those folders whose corresponding vault folders have been modified. DesignSync performs an incremental populate by default whenever possible, although it automatically reverts to a full populate when necessary.

To change the default populate mode, your DesignSync administrator can use the SyncAdmin tool, or you can use the **Save Settings** button to override the default mode.

Note: This option by itself does not cause state changes of objects in your work area (for example, changing from locked to unlocked objects or unlocked objects to links to the cache). DesignSync changes the states of updated objects only. Furthermore, for an incremental populate, DesignSync only processes folders that contain updated objects, so state changes are not guaranteed. Select **Unify workspace state** to change the state of objects in your work area.

For incremental populate operations: If you exclude objects during a populate, a subsequent incremental populate will not necessarily process the folders of the previously excluded objects. DesignSync does not automatically perform a full populate in this case. To guarantee that previously excluded objects are fetched, turn off the **Incremental populate** check box for the subsequent populate operation.

This option usually is not relevant for module data. However, there are two circumstances in which you should deselect this option for a full populate of module data:

- To re-fetch data that was manually removed.

  The **Unify workspace state** option also re-fetches such data, but for missing objects to be considered for the operation, you must deselect the **Incremental populate** option.

- To re-fetch objects that are unchanged from the current module version to the new one, but for which the version in the workspace is incorrect.

  For example, let's say there is a module `Chip`, containing the member `alu.v` . You have version 1.4 of the module `Chip`. The Latest version of the module is 1.5. There is no change to the file `alu.v` between module `Chip` versions 1.4 and 1.5; `alu.v` is version 1.3 in both module versions. However, you actually

have version 1.2 of `alu.v`, having specifically fetched that file from a previous `Chip` module version 1.3. In this case, an incremental populate of the module would not re-fetch the 1.3 version of `alu.v`. A full populate is required to re-fetch the 1.3 version of `alu.v`.

Unify workspace state

Sets the state of all objects processed, even up-to-date objects, to the specified state (**Unlocked copies**, **Locked copies**, **Links to cache**, **Links to mirror**) or to the default fetch state if no state option is specified. If turned off, DesignSync changes the state of only those objects that are not up-to-date. If checked, DesignSync changes the state of the up-to-date objects, as well.

The **Unify workspace state** check box:

- Does not change the state of locally modified objects; select the **Replacement Mode** setting **Force overwrite of modified objects** to force a state change and overwrite the local changes.
- Does not change the state of objects not in the configuration; use the **Replacement Mode** setting **Force overwrite of modified objects** to remove objects not in the configuration.
- Does not cancel locks. To cancel locks, you can select **DesignSync =>Cancel** to cancel locks you have acquired.

Note:

The **Unify workspace state** option is ignored when you lock design objects. If you check out locked copies or locked references, DesignSync leaves all processed objects in the requested state.

Populate empty directories

Determines whether empty directories are removed or retained when populating a directory. Select this option to retain empty directories.

If you do not select this option, the populate operation follows the DesignSync registry setting for "Populate empty directories". This registry setting is by your DesignSync administrator using the SyncAdmin tool. By default, this setting is not enabled; therefore, the populate operation removes empty directories.

This option is not applicable to module data. To prevent specific empty directories from being created by the populate, use the **Filter** field.

Href filter

See Href Filter Field

Href mode

The **Href mode** option lets you specify how hierarchical references should be evaluated in order to identify the versions of submodules to reference when populating a module recursively. This field is only available when operating on module data.

> **Normal mode:** Examines the href's selector. If the href's selector represents a *static* version (a numeric version, a version tag, or a list of such items), the populate fetches the submodule using the selector. In this case, the populate uses **Static mode** for referencing subsequent levels of hierarchy. If the href's selector does *not* represent a static version, the populate still fetches the submodule using the selector, but continues to use **Normal mode** for referencing subsequent levels of hierarchy. This is the default **Href mode**.
>
> **Static mode:** Populates the static version of the submodule that was recorded with the href at the time the parent module's version was created.
>
> **Dynamic mode**: Evaluates the selector associated with the href to identify the version of the submodule to populate.

The **Href mode** option is mutually exclusive with the option to fetch **Locked copies**.

Version

Specify the version number or tag (or any selector or selector list) of the objects on which to operate.

For DesignSync folders, this field is set to the folder's persistent selector list by default. The **Version** field has a pull-down list containing suggested selectors; see Suggested Branches, Versions, and Tags for details.

> **Note:** Modules do not support auto-branching, so for module data, the Version field can not contain the auto() construct.

When populating module data, specifying a **Module Version** value never changes the workspace selector, even for the initial populate of a module. To set or change the workspace selector, use the **setselector** command.

Overlay version in workspace

For DesignSync data, if this option is selected, the version specified in the **Version** field is used to indicate the version to be overlaid on the work area. Overlaying a version does not change the current version status as stored in the workspace metadata. The **Overlay version in workspace** option is often used in conjunction with the **Merge with workspace** option, to merge one branch onto another.

To find out how the **Overlay version in workspace** option operates on module data, see Overlaying Module Data.

Keys

See Keys Field.

Local Version

See Local Version Field.

Module cache mode

This field only applies to module data. A populate operation can link to data in a module cache instead of fetching data from the server, to help decrease fetch time and save disk space.

- **Link to module cache**. (UNIX only) Creates a symbolic link from your workspace to the base directory of a module in the module cache. This is the default mode on UNIX platforms.

  Note: To use this option, the working directory must be empty. In order to guarantee that the workspace is empty, the GUI client requires that the Working directory not already exist for the initial mcache populate. The command line does not enforce this, but overwrites any duplicate files in the workspace.

- **Copy from the module cache**. Copies a module from the module cache to your work area.

  Notes:
  - This mode is the default mode on Windows platforms.
  - This mode only applies to legacy modules.
- **Fetch from the server**. Fetches modules from the server.
- This option overrides the default module cache mode registry setting. If the registry value does not exist, the **Module Cache Mode** selection defaults to **Link to module cache** (UNIX platforms) or to **Copy from the module cache** (Windows platforms).

Note: When a link to an mcache is created, the fetch mode, specified by the "Populate with" option, is ignored and the module is fetched according to Module cache mode settings.

Module Cache Paths

This field only applies to module data. Use the **Module Cache Paths** field to specify paths to the module caches that the populate operation searches, when using a **Module Cache Mode**. You must specify the absolute path to each module cache. To specify multiple paths, separate paths with a comma (,). The paths must exist.

If no **Module Cache Paths** are specified, the populate operation fetches modules from the server.

Log populate messages in populate log file

Use this option to log populate messages to a populate log file. The log file provides easy access to the populate messages to allow for later review. This is particularly useful for complex populate operations such as merging a module version from another branch.

> Note: If the specified log file already exists, DesignSync will append the results of this populate operation to the file. If the file cannot be created for any reason, such as the directory specified does not exist, or you do not have write permissions to the directory, the populate operation fails.

Trigger Arguments

See Trigger Arguments Field.

Extra command options

List of command line options to pass to the external module change management system. Any options specified with the Extra Command options field are sent verbatim, with no processing by the populate command, to the Tcl script that defines the external module change management system. For more information on external modules, see External Modules.

Operate on Design Hierarchy

See Operate on Design Hierarchy.

Switch Using

See Switch Using Field.

Switch List

See Switch List Field.

Stop List

See Stop List Field.

Switch Libraries

See Switch Libraries Field.

Names

See Library Name Field.

Process Views

See Process Views Field.

Names

See Process Names Field.

Process Files

See Process Files Field.

List Cell Views

See Hierarchy List.

Command Invocation

See Command Invocation Field.

Command Buttons

See Command Buttons.

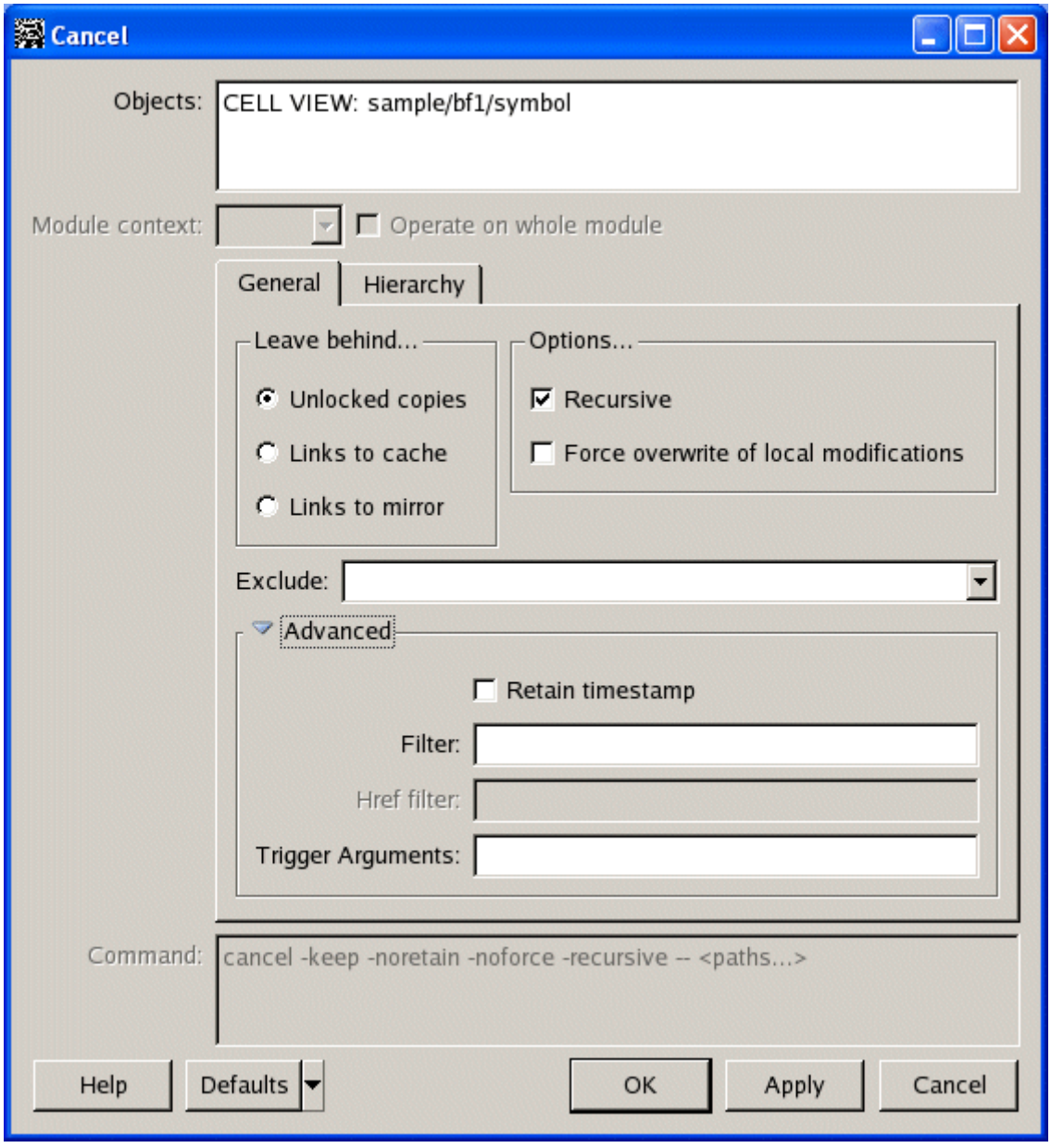
Cancel

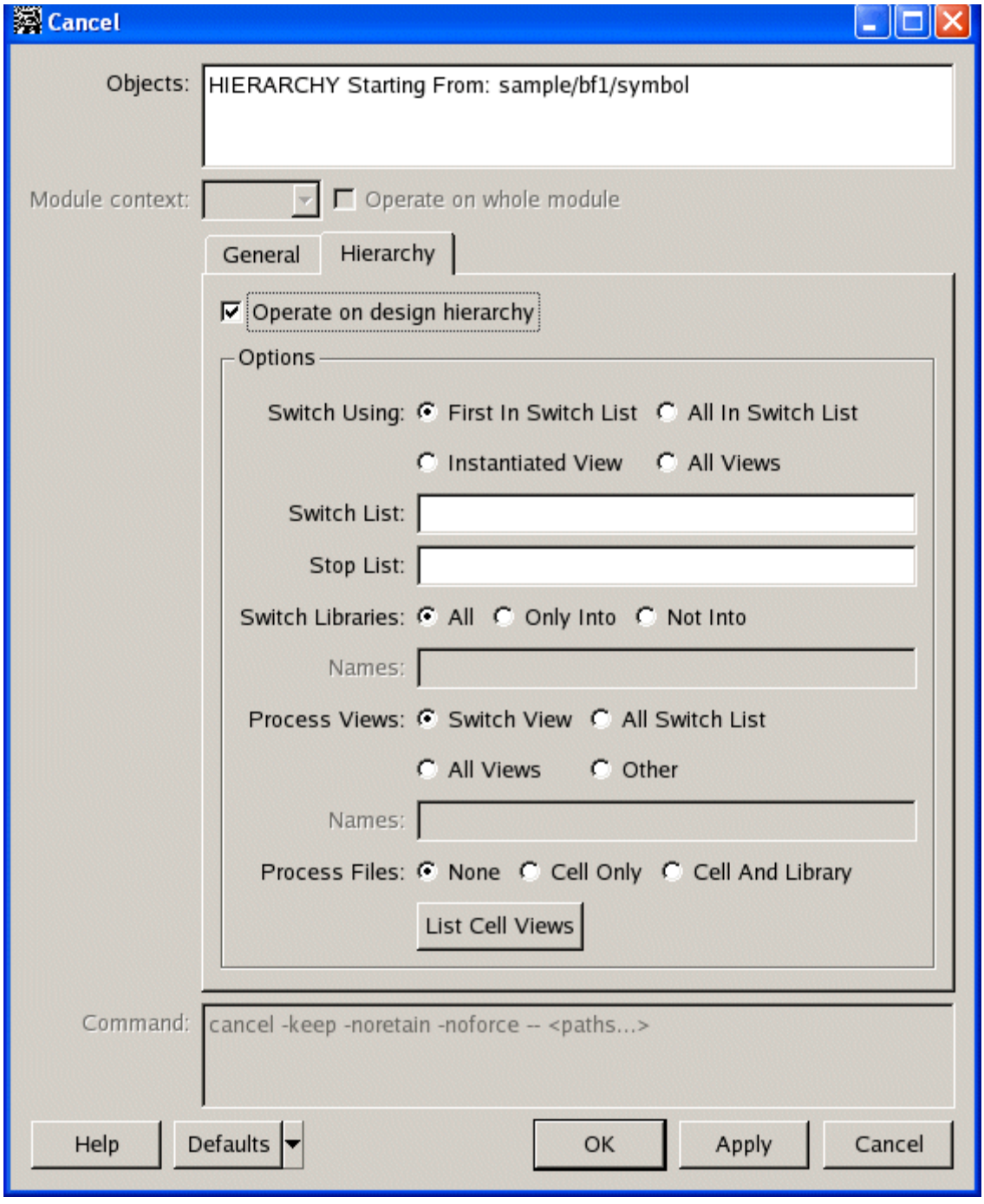
If you check out an object for editing (with a lock) and then change your mind about creating a new version, you can undo (cancel) the checkout with the **Cancel Checkout** dialog box. When you cancel a checkout, an unlocked copy of the object remains in your work area unless your project leader has defined a default fetch state. You can also request a different object state if you use the **cancel** command. If you have made changes to the object prior to canceling the checkout, you cannot change the object state, thereby overwriting your local modifications, unless you specify the `-force` option to the `cancel` command.

This command only cancels a checkout performed by you.

Note: You cannot cancel a checkout on a module member that has been moved or removed in the workspace.

Click on the fields in the following illustrations for information.





Objects

Lists the locked objects that you want to remove the checkout for.

Module context

See Module Context Field.

Operate on whole module

If a module object is selected for the operation, determines whether to operate on the whole module or only the selected objects in the Objects field.

This option is inactive when a hierarchy operation is being performed. This option is not displayed when a whole library is selected. Smart module detection determines if the entire library is within a module and whether the operation should be run in a module centric or folder centric way.

Leave behind Unlocked copies

After the operation is over, keep an unlocked copy in your work area. This is the default unless your project leader has defined a default fetch state.

Because you have relinquished any lock you may have had on the file, someone else can check the file out from the vault with lock in order to modify it.

The SyncAdmin setting **Check out read only when not locking** on the **General tab** determines whether these files are read/write or read-only.

Leave behind Links to cache

Use this option to link to a shared copy of the design object in a cache directory. This option is available only on UNIX platforms.

Leave behind Links to mirror

Use this option to link to a shared copy of the design object in a mirror directory. This option is available only on UNIX platforms. This option is not available for module data.

Force overwrite of local Modifications

Select this option if you want to overwrite locally modified files with the vault version and remove managed objects that do not match the requested selector.

The intent of this mode is to make the workspace match the data being requested as closely as possible. Unmanaged data is never removed.

Operate on Design Hierarchy

See Operate on Design Hierarchy.

Switch Using

See Switch Using Field.

Switch List

See Switch List Field.

Stop List

See Stop List Field.

Switch Libraries

See Switch Libraries Field.

Names

See Library Name Field.

Process Views

See Process Views Field.

Names

See Process Names Field.

Process Files

See Process Files Field.

List Cell Views

See Hierarchy List.

Command Invocation

See Command Invocation Field.

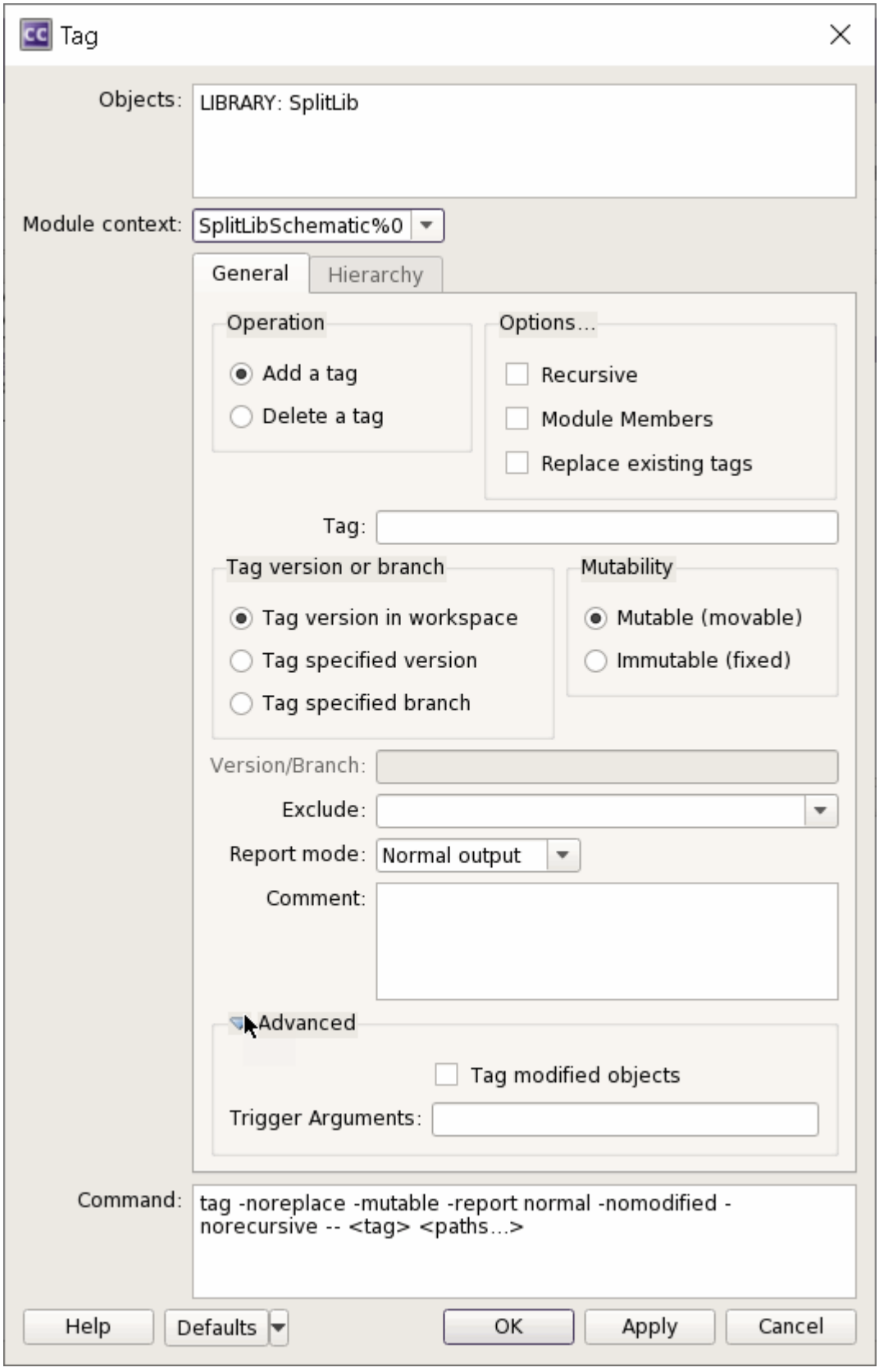
Command Buttons

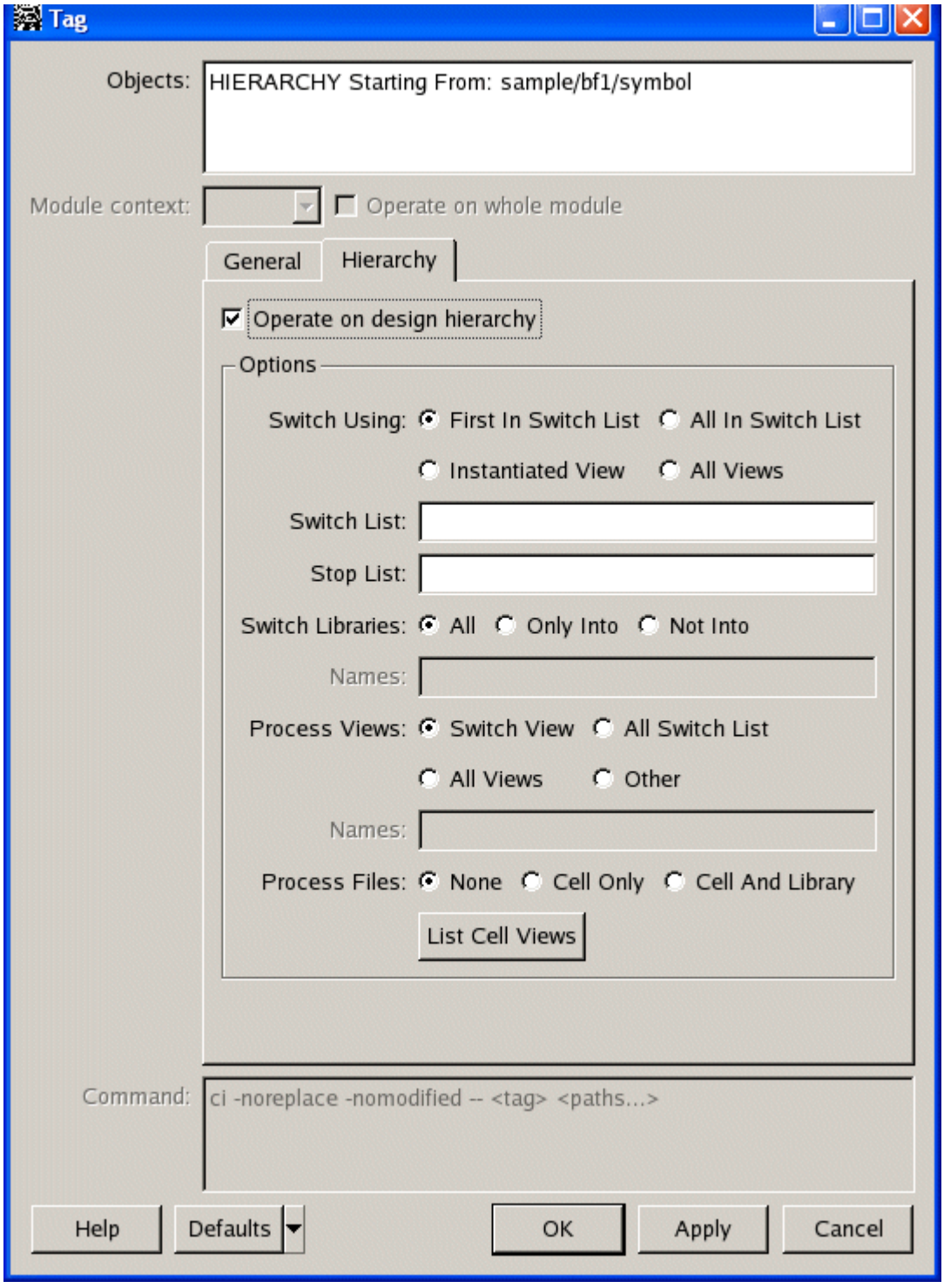
See Command Buttons.

Tag

Tagging is the application of a symbolic name, called a tag, to a version or a branch. Tags can only be applied to objects that are under revision control. For more information on the concept of tagging object versions or branches and naming conventions for tags see the DesignSync Help: Tagging Versions and Branches.

Click on the fields in either of the following illustrations for information.


Tag
Objects:
LIBRARY: SplitLib
Module context:
SplitLibSchematic%0
General
Hierarchy
Operation
Add a tag
Delete a tag
Options...
Recursive
Module Members
Replace existing tags
Tag:
Tag version or branch
Tag version in workspace
Tag specified version
Tag specified branch
Mutability
Mutable (movable)
Immutable (fixed)
Version/Branch:
Exclude:
Report mode:
Normal output
Comment:
Advanced
Tag modified objects
Trigger Arguments:
Command:
tag -noreplace -mutable -report normal -nomodified -norecursive -- <tag> <paths...>
Help
Defaults
OK
Apply
Cancel



Objects

Lists the objects selected for checkout.

Module context

See Module Context Field.

Operate on whole module

If a module object is selected for the operation, determines whether to operate on the whole module or only the selected objects in the Objects field.

This option is inactive when a hierarchy operation is being performed. This option is not displayed when a library is selected. Smart module detection determines if the entire library is within a module and whether the operation should be run in a module centric or folder centric way.

Add a tag

Select this option to add a tag.

Delete a tag

Select this option to delete a tag. Because a tag can exist on only one version of a file at a time, the **Tag version or branch** field is ignored when you delete a tag.

Recursive

See Recursive Field.

Module Members

Tagging individual module members creates a module snapshot, a collection of versionable module members identified as a collective (snapshot) view of the workspace at any given moment. The snapshot is stored on a special tagged branch, named SNAPSHOT_Tag. The specific module member tag on that branch is the Tag name. For more information on tagged module members, see the *ENOVIA Synchronicity DesignSync User's Guide*, Module Member Tags.

Replace existing tags

Move a tag to the version or branch specified in the **Version/Branch** field, even if that tag is already in use on another version or branch. For example, suppose that at the end of every week you want to select the latest files that produce a good demo and tag them "current_demo". To reuse the "current_demo" tag in this way, you must check the **Replace existing tags** check box.

By default (if you do not select **Replace existing tags**), a tag operation fails if the tag is already in use, because a tag can be attached to only one version or branch of an object at a time. Note that you can move a tag from a branch to a version or a version to a branch. DesignSync provides a warning message when you do so.

Tag modified objects

For modified objects in your work area, tag the version in the vault that you fetched to your work area. You might use this option, for example, if you have modified objects in your work area and you want to take a "snapshot" of them as they were before you made the changes.

If you do not specify this option, when the tag operation encounters a locally modified object, the operation displays an error for the object and does not tag any version of that object in the vault.

Note: This option affects modified objects only. If a work area object is unmodified, the tag operation tags the version in the vault that matches the one in your work area.

Tag

Enter a tag name here that is easily understood - for example, "Rel2.1", "ready _for _simulation", "current_demo", "Golden".

Tag names:

- Can contain letters, numbers, underscores (_), periods (.), hyphens (-), and forward slashes (/). All other characters, including white space, are prohibited.
- Cannot start with a number and consist solely of numbers and embedded periods (for example, 5, 1.5, or 44.33.22), because there would be ambiguity between the tag name and version/branch dot-numeric identifiers.
- Cannot end in --R. (The --R tag is reserved for use by legacy modules.)
- Cannot be any of the following reserved, case-insensitive keywords: Latest, LatestFetchable, VaultLatest, VaultDate, After, VaultAfter, Current, Date, Auto, Base, Next, Prev, Previous, Noon, Orig, Original, Upcoming, SyncBud, SyncBranch, SyncDeleted. Also, avoid using tag names starting with "Sync" (case-insensitive), because new DesignSync keywords conventionally use that prefix.

Tag version or branch

Specify which version or branch of the object to tag.

- To tag the version of the object in the vault that is the same as the one you have in your work area, select **Tag version in workspace**.
- To tag an object version in the vault different from the one in your work area, select **Tag specified version** and specify the version number or name in the **Version/Branch** field.
- To tag a specific branch of an object, select **Tag specified branch** and specify the branch name in the **Version/Branch** field.

Mutability

These options only pertain to module data. They are ignored when other types of objects are tagged.

When a tag is added, the new tag is marked as **Immutable (fixed)** or **Mutable (movable)**. A mutable tag can be replaced or deleted. To replace or delete an immutable tag, **Immutable (fixed)** must be selected. The default **Mutability** selection is **Mutable (movable)**.

Comment

Specify a branch creation comment (modules only). This comment is displayed in the object version history. Comments are limited to a maximum of 1024 characters.

Version/Branch

Specify the version or branch of the objects to tag. If you selected the **Tag specified version** option, you must specify a version selector or selector list. If you selected the **Tag specified branch** option, you must specify a single branch tag, a single version tag, a single auto-branch selector tag, or a branch numeric, but not a selector or selector list.

The **Version/Branch** field has a pull-down menu from which you can query for existing versions and branches. See Suggested Branches, Versions, and Tags for details.

Operate on Design Hierarchy

See Operate on Design Hierarchy.

Switch Using

See Switch Using Field.

Switch List

See Switch List Field.

Stop List

See Stop List Field.

Switch Libraries

See Switch Libraries Field.

Names

See Library Name Field.

Process Views

See Process Views Field.

Names

See Process Names Field.

Process Files

See Process Files Field.

List Cell Views

See Hierarchy List.

Command Invocation

See Command Invocation Field.

Command Buttons

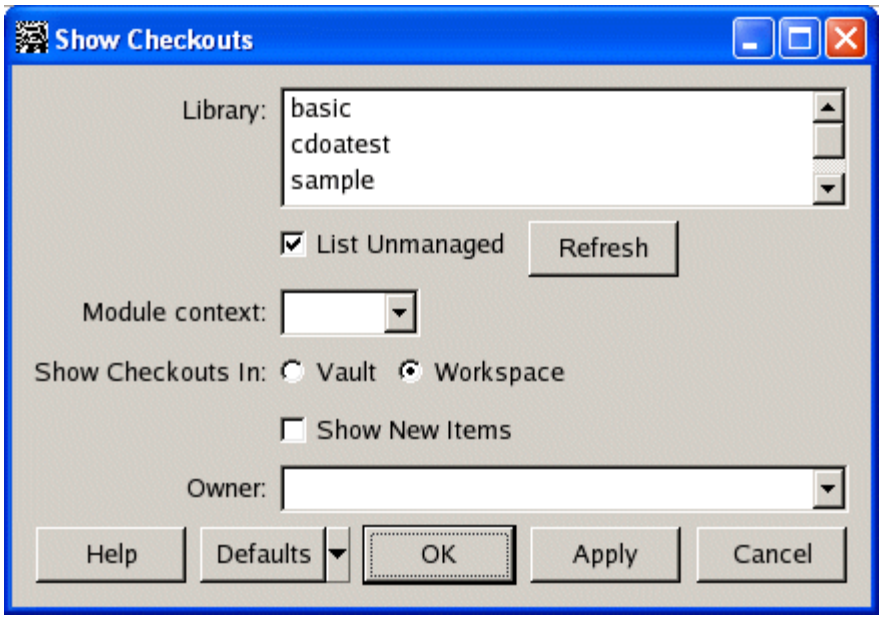
See Command Buttons.

Show Checkouts

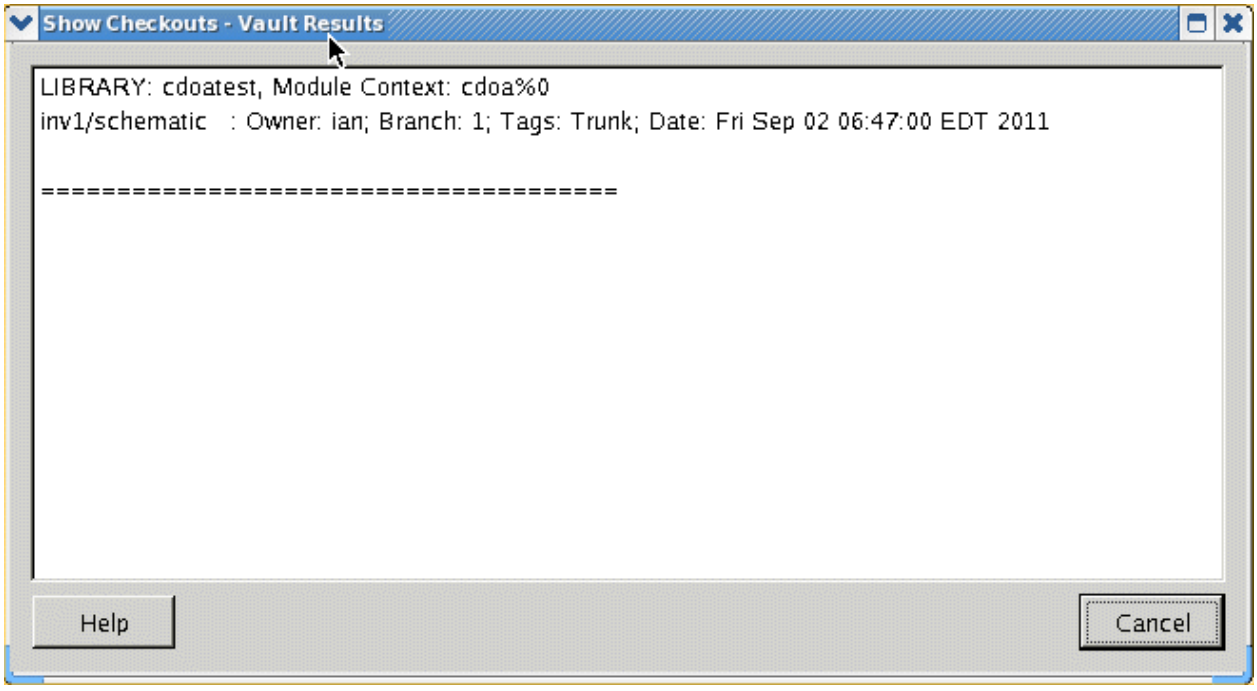
This topic describes the Show Checkouts dialog box which appears when you select the Show Checkouts command. It is used to list the objects that are locked in one or more design libraries (checked out for editing). You can display either the objects that are locked in your workspace or all the objects that have been locked in the vault. For objects locked in your workspace, you can use the Show Checkouts form to check in objects or to cancel your checkouts. You also can use the Show Checkouts form to find objects in your workspace that have never been checked into the vault.

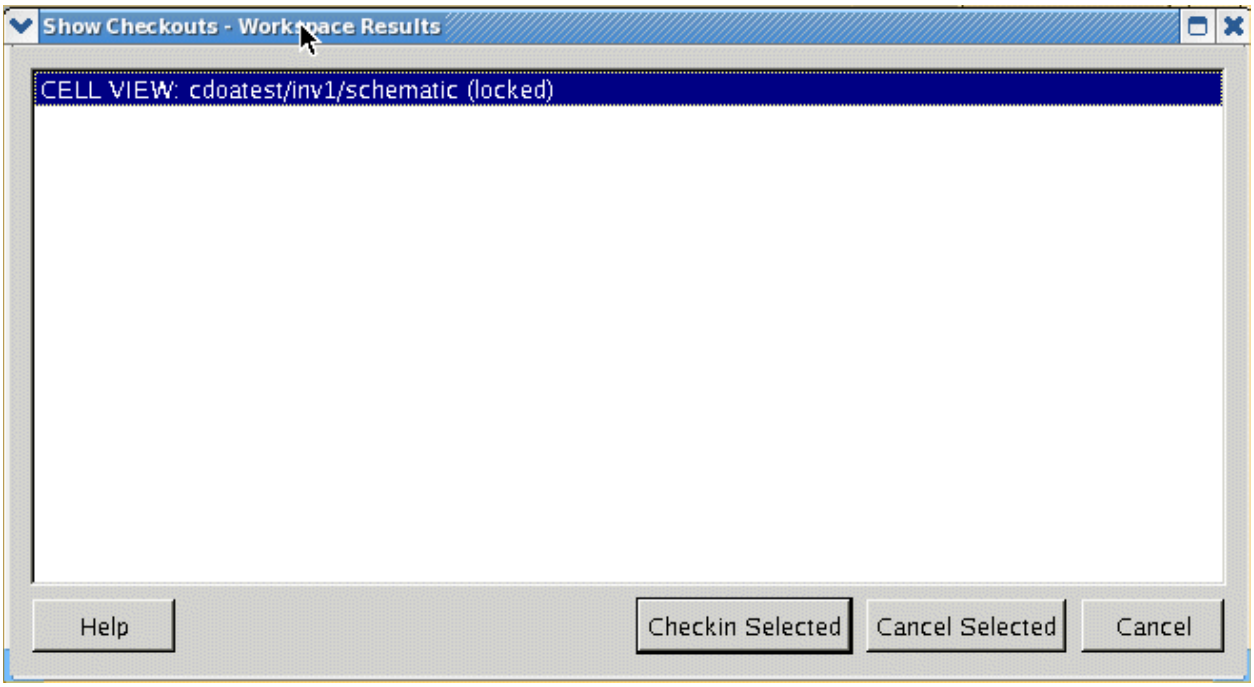
The results are displayed in a results form as a sorted, scrollable list. When this is run on a workspace, you can select one or more objects and checkin the selected locked objects or cancel the lock. When this is run on a vault, you can review the list.

Click on the fields in the following illustrations for information.



The following two illustrations show the show checkout results.





Library Name

Select a library whose checkouts you want to view. To select multiple libraries, use Control-left click, or to select a range of libraries, use Shift-left click. You must select at least one library.

List Unmanaged

The **List Unmanaged** option determines whether all libraries are displayed in the **Library Name** list box, or whether only managed libraries (libraries that are under revision control) are displayed. All libraries are displayed by default. When you deselect **List Unmanaged**, the **Library Name** list box updates to show only managed libraries. Note that there may be a delay while the system determines which libraries are managed.

Refresh

Refreshes the list according to the libraries that are known within the Custom Compiler system to pick up any new libraries.

Module Context

See the Module Context Field topic.

Show Checkouts In

Select whether to display only the objects checked out in your workspace (**Workspace**) or all the objects checked out in the vault (**Vault**).

If you select **Workspace**, the Show Checkouts form displays the objects locked in your workspace. You can use this form to check in or to cancel the checkouts (and thus release the lock) of any of the listed objects. When you select **Workspace**, the **Show New Items** checkbox is displayed. If you select **Show New Items**, the Show Checkout form will also display any objects in your selected workspace libraries or specified directory that can be edited but have never been checked in. You have an option to select the objects for check in.

If you select **Vault**, a list of all checked-out objects (locked by any user) from the selected libraries is displayed in a text window.

Show New Items

The **Show New Items** checkbox is only available when you have selected **Workspace** from the **Show Checkouts In** field. When you select **Show New Items**, the Show Checkouts form displays the checked-out objects and any objects in your selected workspace libraries that have never been checked in (unmanaged objects). You have the option to check in these objects.

Owner

Select a username from the pull-down list to restrict the replies to checkouts owned only by that user. By default, this is set to **All Users**, which shows all checkouts regardless of the owner. You may also select **Current User** to show all checkouts by the current user.

Results

For show checkouts on the vault, you see a non-selectable text field that shows:

- the locked objects
- the lock owner
- the locked branch locked
- the date the lock was obtained.

For show checkouts on the workspace, you see a list of all the checkouts in the workspace. From the list, you can select one or more of the files to checkin or cancel the checkout.

You can save the results to a file for later review.

Checkin

See Checkin.

Cancel

See Cancel

Command Buttons

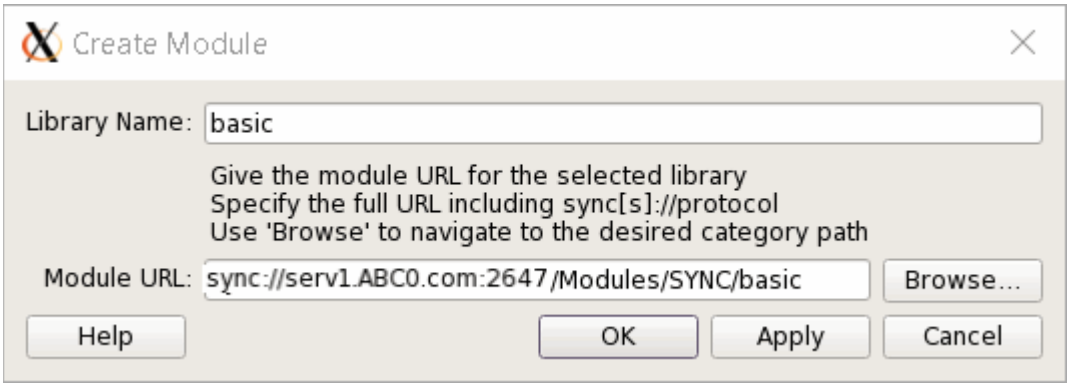
See Command Buttons.

Create Module

You create a module by selecting an unmanaged library and selecting the Create Module command. This command:

- creates the module on the server
- sets the module workspace
- performs an initial checkin with the required files that are part of the library, the `data.dm` and `oalib` files.

Click on the fields in the following illustration for information.



Library Name

Select a library to add to the newly created module. The library must exist and be unmanaged.

Module URL

The path to the module including server name, port number, and path to new module in the format:

`sync[s]://<host>:[<port>]/Modules/[<category>...]/<Module Name>`

Protocol specifies whether to use a standard connection or an SSL connection. For a standard connection use "sync" as the protocol. For an SSL connection, use "syncs" as the protocol.

Host specifies the hostname of the SyncServer for the new module.

Port specifies the port number of the SyncServer. If no port number is specified, DesignSync uses the default port, which is 2647 for a standard connection or 2679 for an SSL connection.

Modules is the virtual location on the server for all modules. All modules must be in the modules area.

Category is an optional field that provides a means of organizing modules. Specify a virtual path category to group related modules. The term category is used to indicate that this is a virtual path, rather than a physical path.

Module name specifies the name of the module.

Note: Module and category names have certain restrictions in terms of the characters allowed. For more information see *ENOVIA Synchronicity DesignSync User's Guide*: URL Syntax.

Tip: By convention, module names begin with an initial capital letter to allow users to distinguish them easily from workspaces, which, by convention, begin with an initial lower case letter.

To browse the list of known servers click the Server Browser button.

Browse

Launches the Server Browser dialog.

Command Buttons

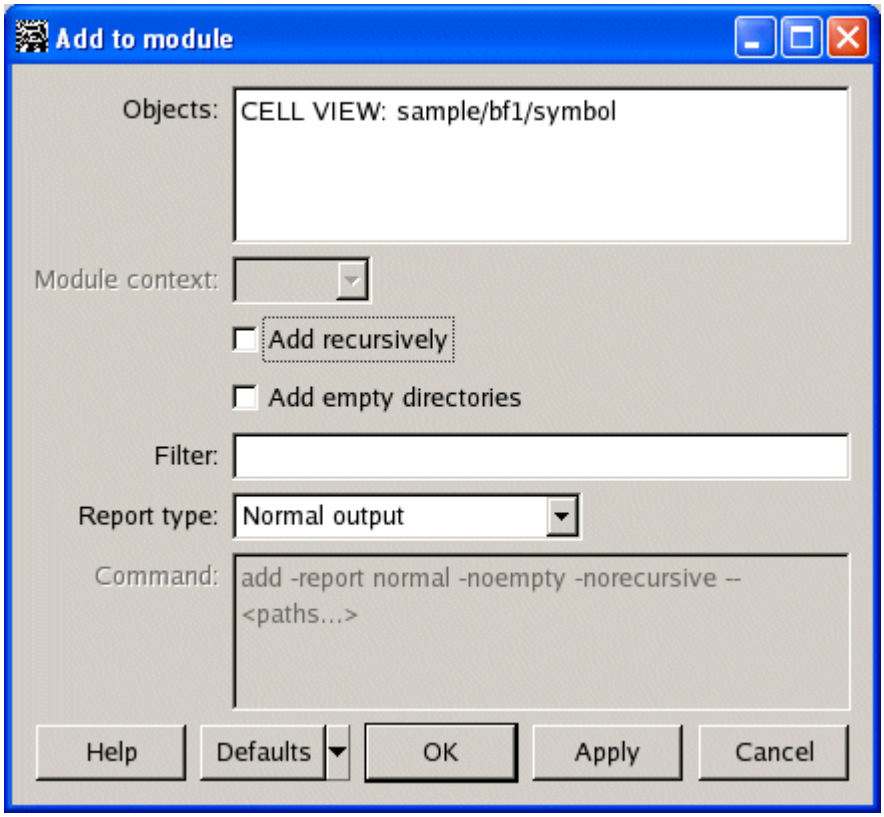
See Command Buttons.

Add to Module

When you add a new object or library to a module, you can either perform a check in operation with the New option, or, use the Add To Module command to add the object to

the module in a separate operation. The locally added objects are checked into the module version created by your next checkin operation.

Click on the fields in the following illustration for information.



Objects

Lists the objects to add to the module.

Module context

See Module Context Field.

Add recursively

For a folder, whether to add only the folder or also recursively add the folder's contents. Note that folders themselves can be module members. If you checked the Add empty directories option, adding a folder without content results in an empty folder as a module member.

By default, the option to **Add recursively** is not selected.

Add empty directories

This option is used with **Add recursively**. When adding members recursively, any folder that contains files is added to the module. The **Add empty directories** option specifies whether directories without content are added to the module.

For example, let's say you're recursively adding "dirA" to a module. "dirA" contains files, and an empty subdirectory "dirB". The option to **Add empty directories** controls whether the empty directory "dirA/dirB" is added.

Report type

From the pull-down, select the level of information you want to display in the output window:

Brief output: Lists errors generated when adding objects to the local module, and success/failure count.

Normal output: Lists all objects added to the local module, success/failure count, and beginning and ending messages for the add operation. This is the default output mode.

Verbose output: In addition to the information listed for the Normal output mode, lists:

- Skipped objects that are already members of the module.
- Skipped objects that are already members of a different module.
- Skipped objects that are filtered.
- Status messages as each folder is processed.

Errors and Warnings only: Lists errors generated when adding objects to the local module, and success/failure count.

Command Invocation

See Command Invocation Field.

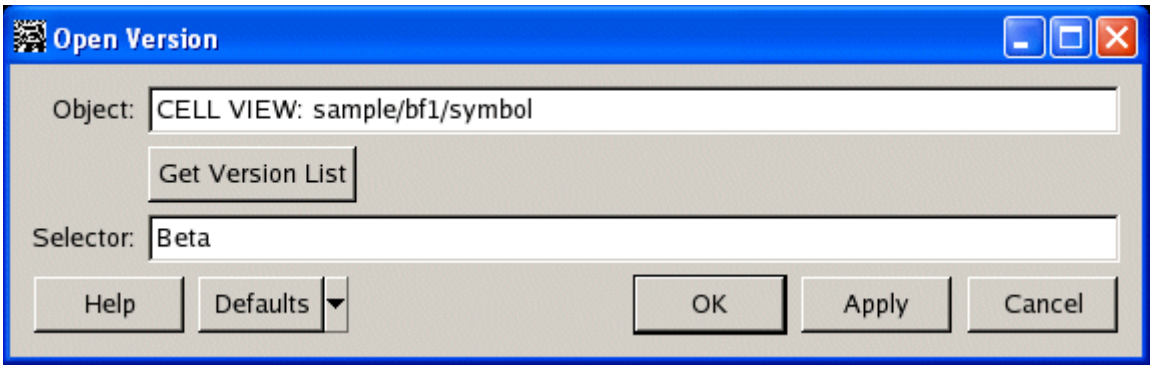
Command Buttons

See Command Buttons.

Open View Version

You can display cell view versions other than the version in your workspace. For example, you might want to compare two or more versions of the same cell view.

Click on the fields in the following illustration for information.



Object

Lists the object selected.

Get Version List / Branch / Version / Comment

In the initial invocation, DSCC displays the "Get Version List" button which queries the server for a list of versions. After the query has been run, DSCC displays a selectable list of versions, and a comment field.

Branch

Displays the list of available branches to report on. If there is only one branch, this option displays “ALL”and is not modifiable. If there is more than one branch, it provides the option to select “ALL” branches, only the “Current Branch,” or choose from the list of branch tags.

Version

Displays the versions available to view. Select the appropriate version.

Comments

Displays the check-in comment for the selected version.

Selector

Specify the selector or comma separated selectors, such as a version tag, that determines what view version to display. For example, specify "gold" to view the view version that is tagged "gold".

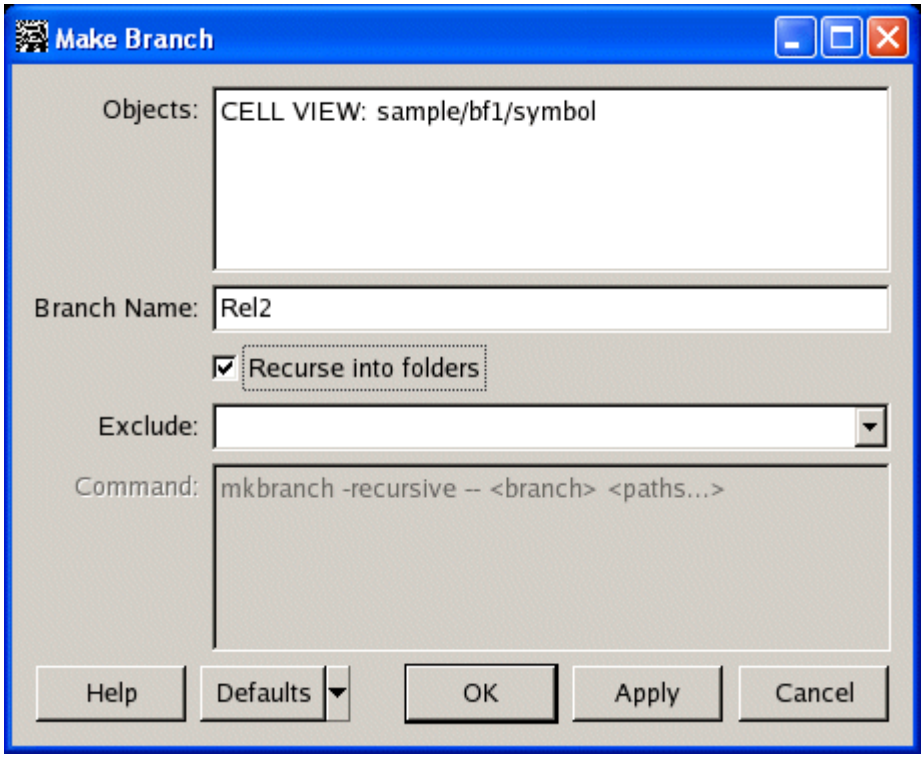
Command Invocation

See Command Invocation Field.

Branch

When a project requires a new branch, a single person, typically a release engineer or project manager, uses the **Make Branch** dialog box to branch all design objects at the same time. Users can then create a new work area for the new branch. The branch point version for make branch operations initiated within DSVS is always the version of the selected objects or module loaded into the workspace.

Click on the fields in the following illustration for information.



Objects

Lists the objects being branched.

Branch Name

Enter a branch tag name here that is easily understood - for example, "Rel2.1", "ready _for _simulation", "current_demo", "Golden".

Branch names:

- Can contain letters, numbers, underscores (_), periods (.), hyphens (-), and forward slashes (/). All other characters, including white space, are prohibited.

- Cannot start with a number and consist solely of numbers and embedded periods (for example, 5, 1.5, or 44.33.22), because there would be ambiguity between the tag name and version/branch dot-numeric identifiers.
- Cannot end in --R. (The --R tag is reserved for use by legacy modules.)
- Cannot be any of the following reserved, case-insensitive keywords: Latest, LatestFetchable, VaultLatest, VaultDate, After, VaultAfter, Current, Date, Auto, Base, Next, Prev, Previous, Noon, Orig, Original, Upcoming, SyncBud, SyncBranch, SyncDeleted. Also, avoid using tag names starting with "Sync" (case-insensitive), because new DesignSync keywords conventionally use that prefix.

Note: Referring to a branch by its branch number is not recommended. When the branch is created, the branch name is automatically created as branch tag. You can add additional tags to the branch using Tag.

Command Invocation

See Command Invocation Field.

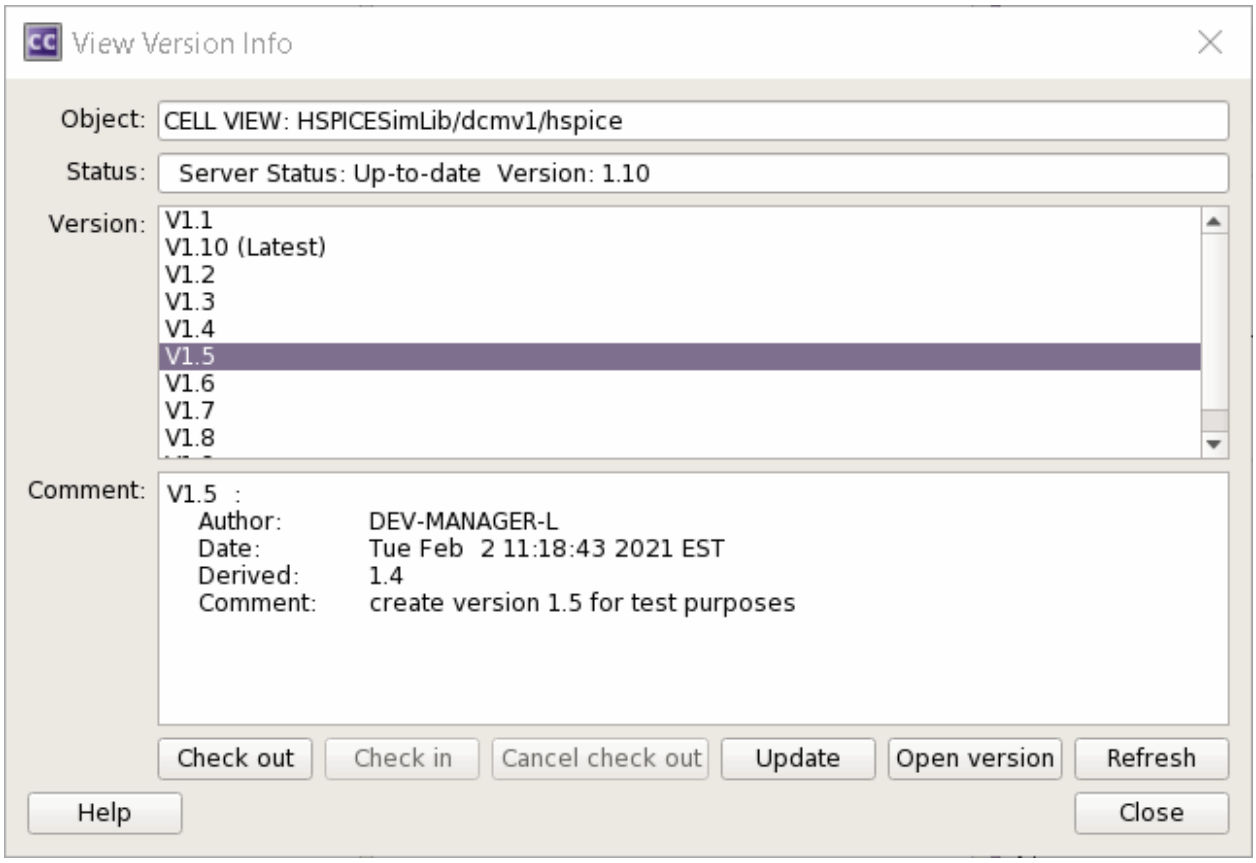
Command Buttons

See Command Buttons.

View Version Info

The View Version Info provides version information about a particular cell view and provides the ability to run certain revision control operations on a version.

Click on the fields in the following illustration for information.



Object

- Type: Type of the object.
- Library Name: Name of the library containing the selected object.
- Cell name: Name of the selected cell, if applicable.
- View name: Name of the selected view.

This example screen shows: *<cell view type><library name>/<cell name>/<viewname>*.

Status

Server status of the latest version on the server.

Version

Lists the versions available for the selected object. When you select a version, the Comment field is populated with information about the version.

You can select a single version on which to perform the possible operations.

Comment

Displays version information about the selection version:

- Version number. The key (Latest) is appended to the version on the branch.
- Version author
- Checkin date
- Version derivation information
- Checkin comment.

Check out

Launches the Checkout dialog box for the object. The object must be unlocked on the server and unmodified in the workspace. The check out operation assumes latest version and preselects Edit (locked copy).

Checking out Design Data

Check in

Launches the Check in dialog. The object must be checked out in the workspace with a lock. You do not need to select a version.

For more information on Check in, see Checking in Design Data.

Cancel

Launches the Cancel checkout dialog. The object must be checked out with a lock in the workspace. You do not need to select the object version.

For more information, see Canceling Checkout of Design Data.

Update

Launches the Check out dialog for the selected object version. The object version must be unlocked in the workspace and unmodified.

For more information, see Checking out Design Data

Open Version

Launches the Open View Version dialog with the version information prefilled from your selection.

For more information, see Opening View Version.

Refresh

Refreshes the version table and updates the status of the objects in the workspace.

Help

Launches this help.

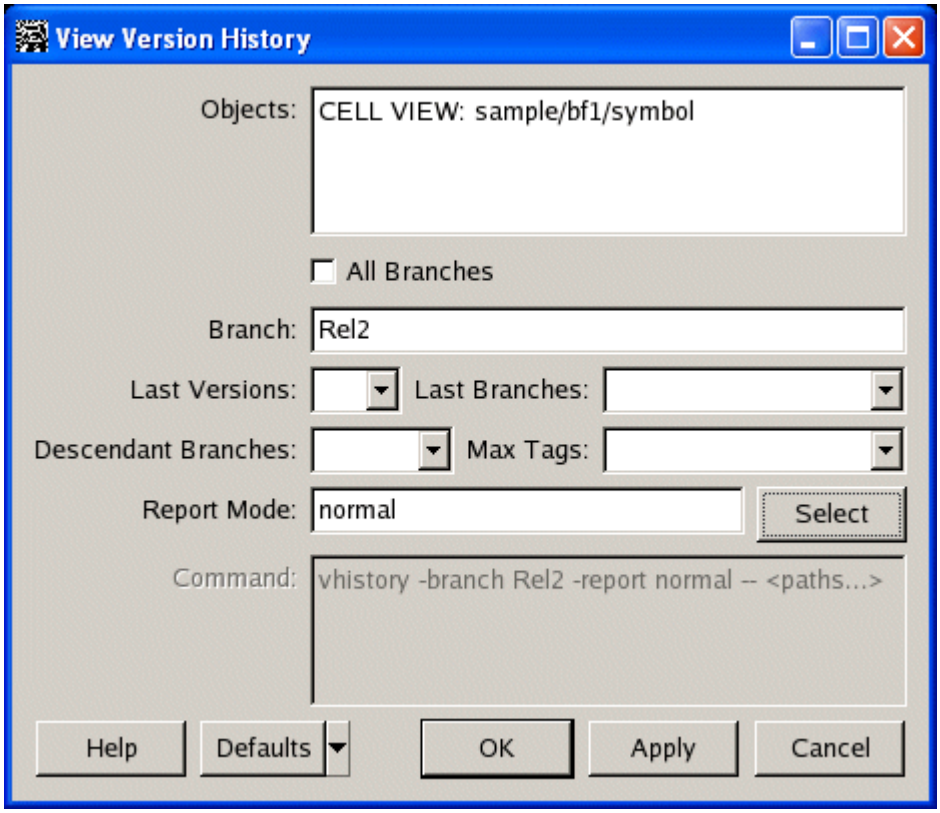
Close

Closes the window.

View Version History

The Version History report displays version history for managed objects.

Click on the fields in the following illustration for information.



Objects

Lists the objects for which the view history is fetched.

All Branches

When selected, displays version history for all branches for the selected object. When deselected, displays version history for the branch fetched into the local workspace or, for vault objects, branch 1 (Trunk:).

This is mutually exclusive with the Branch and Last Versions options.

Branch

Enter a branch tag or numeric. When selected, the command displays the version history for the specified branch only.

Last Versions

Sets the number of previous versions to report. By default, all versions on a branch are reported. You can select a value from the pull-down list, or type in a positive integer. Specifying the number of ancestor versions to report sets the Descendant branches to 0.

Last Branches

Sets the number of previous branches back to report. By default, only versions on the specified branch are reported. You can select a value from the pull-down list, or type in a positive integer. Specifying the number of ancestor branches to report sets the Descendant branch depth value to 0.

Descendant Branches

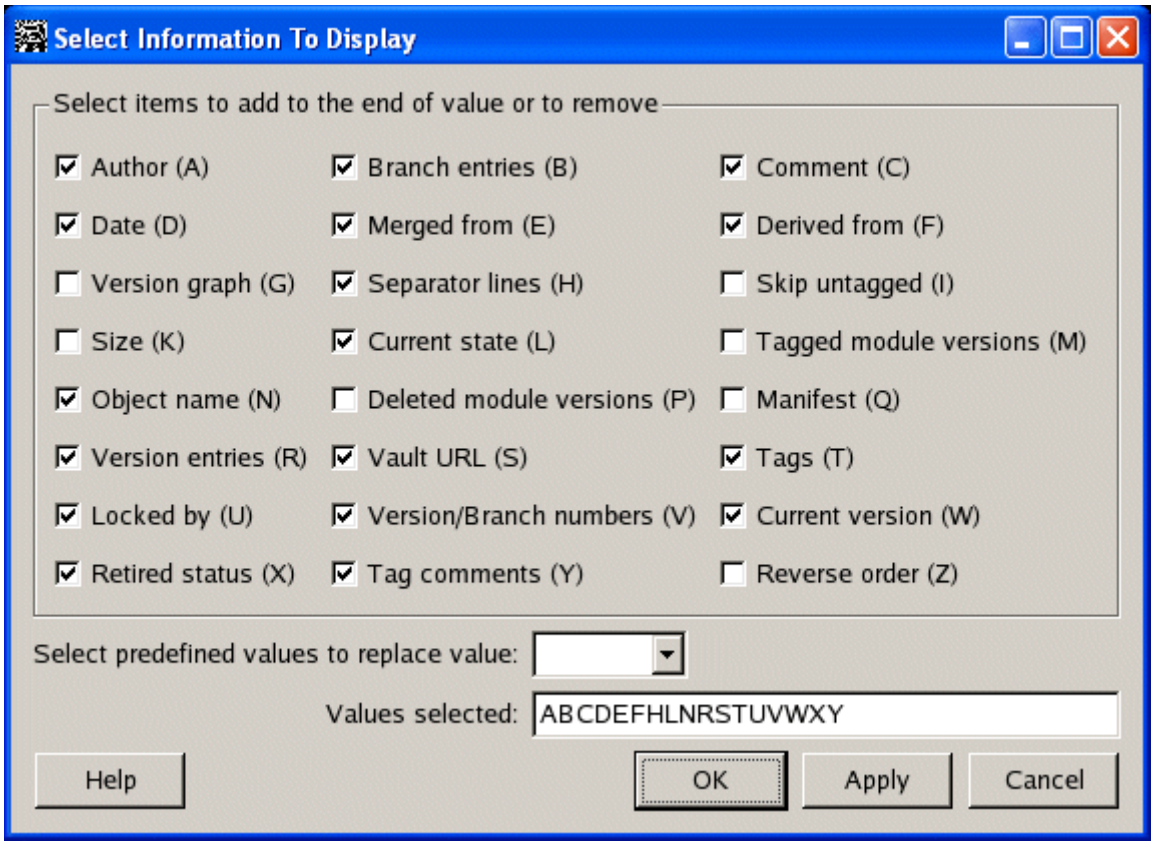
The number of levels of descendant branches to report, from the starting branch. By default, the report is limited to the starting branch (a value of 0). You can select a value from the pull-down list, or type in a positive integer. Specifying a descendant branch depth sets the Number of ancestor versions to report and the Number of ancestor branches to report to **all**.

Maximum tags reported

Sets the maximum number of tags shown for any object. You can select a value from the pull-down list, or type in a positive integer. By default, all tags are shown. This option is only available when Show tags is selected.

Report mode

Specify how the information is reported. The choices **Brief**, **Normal**, and **Verbose** represent defined reports. Selecting one of these defined reports automatically enables all of the report options that comprise the selected mode. Report options not in the selected mode are automatically disabled. The **Normal** report mode is selected by default. Select the **Select** button to specify your own combination of report options.



Show object name

Show the workspace path to the object, or to the vault URL.

Show vault URL

Show the vault URL associated with a workspace object.

Show current version

Show the version currently in the workspace.

Show current state

Show the fetched state in the workspace.

Show tags

Show branch and version tags. Immutable tags are shown with "(immutable)" appended.

Show tag comments

Show the comments associated with version and branch tags. This isn't applicable for Visual Studio data.

Show creation dates

Show a version's creation date.

Show derived version

Show the numerical parent version. This maintains the continuity between versions for merge and rollback operations.

Note: If a merge, skip, rollback or overlay operation occurs to create this version, the referenced version is shown as "Merged from" version.

Show version author

Show a version's author.

Show merged from

Show the version used to create the current version when the current version was created as the result of a rollback, merge, skip, or overlay operation requiring an alternate parent version.

Show check in comment

Show a version's check in comments, and any checkout comments. Checkout comments are only visible from the workspace in which the checkout occurred.

Show retired state

Show whether a branch is retired.

Show locked by

Show the lock owner of a locked branch and the "version -> upcoming version" information.

Show separators

Show separators between items and versions.

Show tagged module versions

Show module version that have tags,even if a module member being queried version has not been changed in that module version.

Show deleted module versions

Show module version that were purged or deleted.

Show size

Show the size of the object version in KB.

Note: Collections and module versions, both of which contain more than one object, display with a size of zero.

Show module manifest

For a module, show the manifest of changes in each version. For a module member, show only the changes to that member.

Show version/branch numbers

Show the version number for versions, and the branch number for branches. For branches, indicate whether any versions exist on the branch.

Display branch entries

Show information for branch objects.

Display version entries

Show information for version objects.

Ignore untagged entries

Do not show entries that have no tags.

Reverse display order

Show the versions/branches in reverse numeric order.

Represent graphically

Show a graphical representation of the version history, as a text graph.

Select predefined values to replace value

DesignSync comes with predefined values that contain a set of commonly used keys.

- brief - Report tagged versions/branches with their tags and numerics. This displays the following key values: "NBRIVT".
- normal - Report all available information, except for the module manifest. This displays the following key values: "ABCDEFHLNRSTUVWXY."
- verbose - Report all available information. This displays the following key values: "ABCDEFHKLMNQRSTUVWXY."

Values selected

This field displays the key values of the selected options. It matches the display shown on the View Version History dialog box.

Command Invocation

See Command Invocation Field.

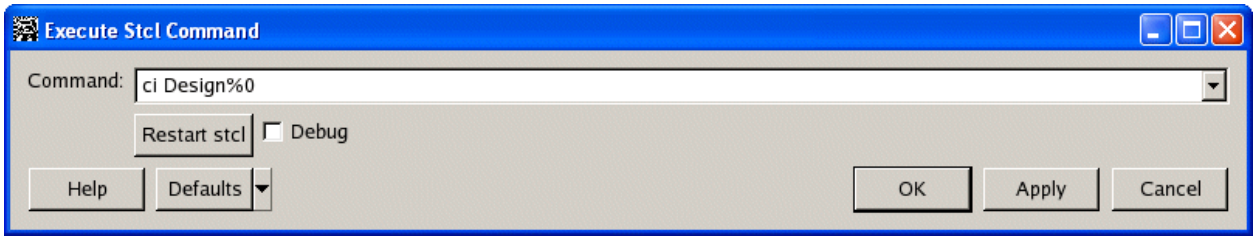
Command Buttons

See Command Buttons.

Execute stcl

This topic describes the Execute stcl dialog box, which appears when you select Execute stcl from the DesignSync menu. It is used to run DesignSync commands from the Custom Compiler interface

Click on the fields in the following illustration for information.



Command

Enter the command to execute in the **Exec stcl Command** window. To run the command, press **OK**, **Apply**, or press the RETURN key.

The command entry box maintains a history of the commands. Click the drop-down arrow at the end of the command line to bring up the command history. Select an item in the history list to copy it to the command line.

Restart stcl

Re-initializes the stclc process. If you change the user environment, for example with SyncAdmin, you can reload the environment setting by using the **Restart stcl** button.

Debug

Enables debugging of the interface to the stclc process. For more information on debugging in DSCC, see dscd::enable_debug.

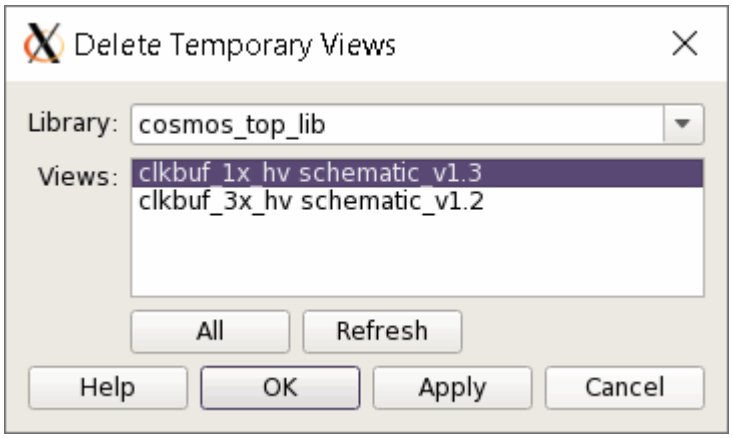
Command Buttons

See Command Buttons.

Delete Temporary Views

This topic describes the Delete Temporary Views dialog box, which appears when you select Delete Temporary Views from the list of libraries.

Click on the fields in the following illustration for information:



Description of Fields

Library

Select the library containing temporary cell views you want to delete. The library must already exist in your workspace. You may select a different library than the one selected when the dialog opens.

Views

List of temporary Views in the selected library. When you change your library, the list of views is automatically updated. If the library has no temporary cell views, nothing is displayed. The temporary views are displayed in the format: *<cellname> <viewname>*.

Select one or more cell views to delete. You can select more than one cell view by using the Control or Shift keys. You can also click the **All** button to select all the listed cell views.

If the available temporary views for the selected library have changed since the list was first displayed, click **Refresh** to rescan the library for temporary views.

Refresh

Refreshes the display of:

- Libraries in the **Library** field.
- Temporary views of the selected library. Click **Refresh** if the available temporary views for the selected library have changed since the form was displayed.

All

Click **All** to select all of the temporary cell views in the selected library. You can then deselect cell views by clicking the cell view while pressing the Control key.

Command Buttons

See Command Buttons.

Options

The options panel provides a way to customize the behavior of DSCC.

Click on the fields in the following illustration for information.

VC System ci options (auto-checkin options):

Specify the check in options to be used for every auto-check in operation. The options change takes place immediately. You do not have to restart stcl.

VC System populate options (auto-checkout options):

Specify the populate options to be used for every auto-check out operation. The options change takes place immediately. You do not have to restart stcl.

Show Tags in Library Manager Versions Column:

Specifies whether the interface should include version tags in the versions column in the Library Manager.

When True is selected, the Library Manager displays version tags in the versions column.

When False is selected, the Library Manager does not display named tags in the versions column, but only the numeric version of the objects.

Note: This option does not control the display of branch tags.

User defined tags list

Specifies the tags to be shown on the drop-down tag fields for the Checkout, Open View Version and Tag forms. The field is a string field containing comma-separated tag values which are validated by the system before being saved. The list reflects what is set in the registry setting and any changes made here will be reflected in the registry.

Workspace root path

Specifies a list, in TCL format, of workspace paths to search for modules. The list reflects what is set in the registry setting and any changes made here will be reflected in the registry.

Note: Modules associated with libraries are usually automatically found when the library is specified.

Directory for temporary files

Specifies the name of a directory we can use for temporary files. By default, this is set to the value "/tmp".

Show Revision Version in Editor

Specifies whether the editor should display the version along with the view name in the title bar of the editor window.

When True is selected, the editor displays the version followed by the view name. When you Descend or Ascend levels in the editor, the title bar updates to show the updated view name and version.

When False is selected, the editor displays only the name of the view being edited in the title bar. When you Descend or Ascend levels in the editor, the title bar updates to show the updated view name.

Command Buttons

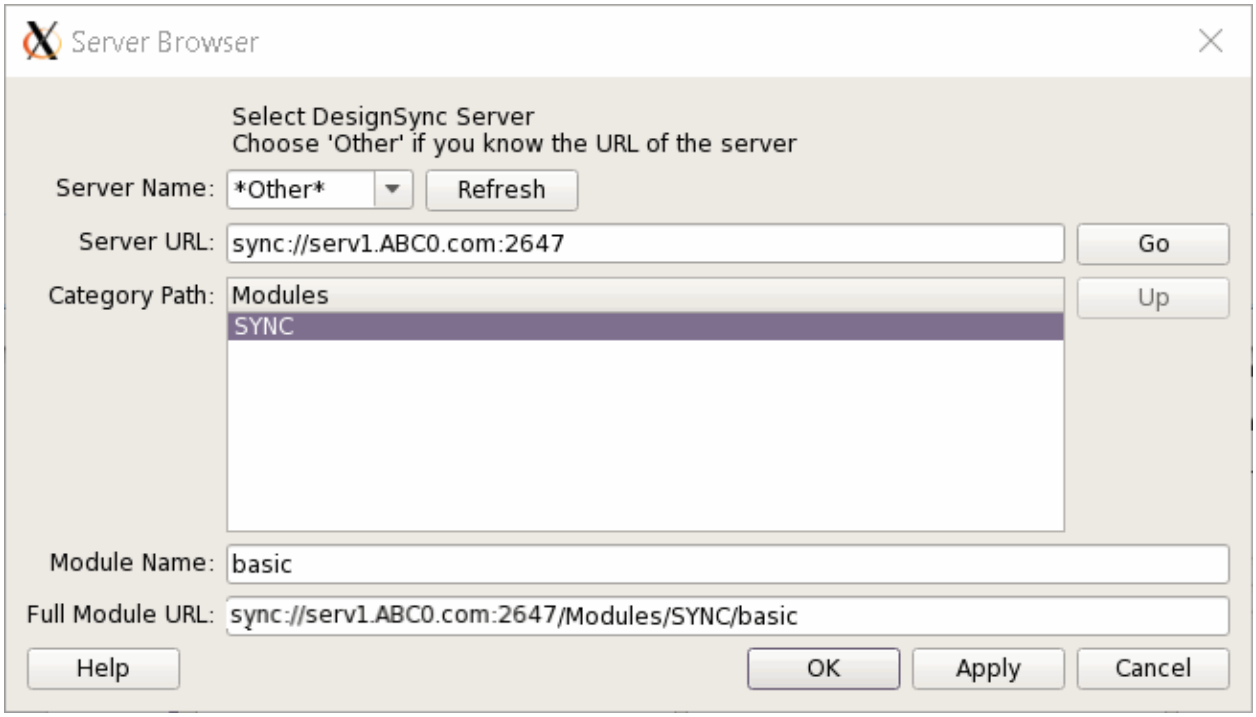
See Command Buttons.

Common Interface Concepts

Server Browser

When selecting a module on the server, you can invoke the Server Browser to show a list of all known servers, categories on the servers and modules. You can also manually enter the server URL.

Click on the fields in the following illustration for information.



Server Name

The list of known servers or *Other*. A server is known if it is defined in any of the `sync_servers.txt` files located in directories specified by these environment variables:

- SYNC_USER_CFGDIR
- SYNC_SITE_CNFG_DIR
- SYNC_ENT_CUSTOM

For more information on creating sync_server.txt files, see SyncServer List Files. For more information on setting configuration and custom environment variables, see Environment Settings for Users.

The *Other* option allows you specify a different server from the one in the server lists in the Server URL field.

Refresh

Refresh the list of known servers.

Server URL

The server URL in the format:

`sync[s]://<host>:[<port>]`

Protocol specifies whether to use a standard connection or an SSL connection. For a standard connection use "sync" as the protocol. For an SSL connection, use "syncs" as the protocol.

Host specifies the hostname of the SyncServer for the new module.

Port specifies the port number of the SyncServer. If no port number is specified, DesignSync uses the default port, which is 2647 for a standard connection or 2679 for an SSL connection.

Go

Retrieve the list of Categories from the server to populate the Category Path field.

Category Path

This field shows a list of categories on the server. The term category is used to indicate that this is a virtual path, rather than a physical path. The category path will always show the Modules category first and allow you select any categories within Modules. Click on a category to view sub-categories. Click Go to refresh the list of categories.

Up

Traverse one level back up the category path after you have descended down into subcategories. You can also click on `..` to travel one category back.

Module Name

Module name specifies the name of the module. By default, it is populated with the name of the selected library.

Note: Module and category names have certain restrictions in terms of the characters allowed. For more information see *ENOVIA Synchronicity DesignSync User's Guide*: URL Syntax.

Tip: By convention, module names begin with an initial capital letter to allow users to distinguish them easily from workspaces, which, by convention, begin with an initial lower case letter.

Full Module URL

This read-only field displays the composite module URL created by combining the Server URL, Category Path, and Module name. When you return to the dialog that called the Server Browser, this URL will be placed in the Module URL field.

Command Buttons

See Command Buttons.

Module Context Field

This field is only available when the library data being operated on is within the scope of one or more modules. Specifying a module context enables the operation to be run on a workspace folder that is below multiple modules or on a sub-folder of a module on a server.

The default module context value is <Auto-Detect>, which uses smart module detection to determine the target module. The field is a drop-down list that allows you to select the module context from an alphabetical list of possible module contexts.

Related Topics

Understanding Smart Module Detection

Retain Timestamp Field

Use this option if you want to retain the "last modified" time of when the file version was checked into the vault.

This option is meaningful only when working with physical copies, as is the case when you specify the **Unlocked copies** or **Locked copies** options. DesignSync automatically uses this "keep last modified time" behavior when linking to files in a mirror or cache directory; files in the mirror/cache retain their original timestamps. However, links in your work area to the cache/mirror have timestamps of when the links were created. If you specify the **References to versions** or the **Locked references** options, no object is created in your work area, so there is no timestamp information at all.

Note: If an object is checked into to the vault and the setting of the **Retain Timestamp** option is the only difference between the version in the vault and your local copy, DesignSync does not include the object in checkout operations.

DesignSync operations follow the SyncAdmin registry setting for Retain last-modification timestamps. By default, this setting is not enabled; therefore, the timestamp of the local object is the time of the check-in, check-out, or cancel operation. To change the default setting, your Synchronicity administrator can use the SyncAdmin tool. For information, see SyncAdmin Help: Command Defaults.

Comment Field

Enter notes for this operation here.

- Check-out comments are added to the Revision Log, which is used as part of the future check-in comments.
- Check-in comments are appended to any comments you have in your Revision Log (**File=>Properties**). Check-in comments become part of the version history.

  Depending on the DesignSync methodology your team adopts, your project leader may require that every check-in have a comment of a given length. If there is a minimum comment length defined with SyncAdmin, a tool-tip window will display in the lower-right portion of the Comment field. The tool tip will show you how many characters you have entered and the minimum character length of the comment. When you reach the minimum length, the tool tip will close.

  > Note: This does not display a tip for minimum comments length defined with Access Controls.

- Tag and Branch comments, which are only available for module data, become part of the version history.
- Comments specified when creating a new module are reported by the `showmods` command as explained in the `showmods` command description: Understanding the Output, and displayed in the List View when viewing modules on a server.

Exclude Field

Enter here the kinds of files or directories you want to exclude from the operation. Wildcards are allowed. Separate items with commas. For example, to exclude all log files from the operation, you would specify ***.log** .

Click on the button to the right of the **Exclude** text field to bring up a list of common exclude patterns (`tmp`,`*.o`,`*.obj`,`*.bak`,`*~`,`*.log`,`*.db`). Selecting a pattern will append the pattern to the current contents of the **Exclude** text field.

Do not specify paths in your arguments to **Exclude** . Before operating on each object (such as during a recursive operation), DesignSync compares the object's leaf name (path stripped off) to the items in the **Exclude** field to see if there is a match. Because the object's path is not considered, it will not match any item in the exclude list specified with a path. For example, if you specify **bin/*.exe** in the **Exclude** field, you will not successfully exclude `bin/test.exe` or any other `*.exe` file. You need to instead specify ***.exe** (or **test.exe** if you want to exclude only `test.exe`). This means, however, that you cannot exclude a specific instance of a file or folder -- you exclude all matching files and folders.

For details of how the **Exclude** value is used by a particular command, see the `-exclude` option description for the command invoked by the graphical interface.

See the SyncAdmin Help: Exclude Lists topic for information about global exclude lists.

Recursive Field

For a DesignSync folder, recursively operate on its contents. The set of objects operated on may be reduced by use of the Exclude field. By default, only the contents of the selected folder are operated on.

For a module, follow any hierarchical references and recursively operate on their contents. The set of objects operated on may be reduced by use of the Exclude field, the Filter field and/or the Href filter field. By default, only the contents of the selected module are operated on.

For a module folder, recursively operate on its contents. The Module context field determines the set of objects operated on. The set of objects operated on may be further reduced by use of the Exclude field and the Filter field. By default, only the contents of the selected folder are operated on.

Filter Field

Apply the specified expression, to identify the exact subset of module members on which the command will operate. Specify comma-separated object expressions, where an object expression takes the form:

```
[+|-]<path_expression>
```

where:

- `+/-` indicates whether a particular object expression indicates items to be included or excluded. The default is `"-"` , excluding items based on the `<path_expression>`
- `<path_expression>` is an extended glob expression that specifies object paths. 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".

For example, the expression `top/.../lib/*.v` matches any `*.v` file in a directory path that starts with `top`, then has any number of levels (zero or more), ending in a `lib` sub-directory.

Note: A glob expression of `*` for a `<path_expression>` is not always the same as `.../*` . For an exclude (`-*` or `*`), the `*` would match the folders in the top level directory, and therefore everything down the hierarchy. But for an include `(+*)`, the `*`

would only match the top level directories, and not match the lower level items, causing the lower level items to remain excluded.

Path expression matches are performed against the relative path of the objects from the starting point of the command.

If a match is performed against a full `sync:` URL, then a "..." value will match the `sync://<host>:<port>` at the start of the URL. For example, `.../Chip` matches `sync://<host>:<port>/Modules/Chip`.

Ordinarily, a command starts with everything included. But if the first entry in the **Filter field** is an include (`+<path_expression>`), then it is assumed that the intention is to start with everything excluded, as if the expression had started with `-.../*` . This makes it simple to write expressions that just include some items.

Notes:

- This field is only available when operating on module data.
- The Exclude field is applied in addition to the **Filter field**.
- When Populating Your Work Area, see Setting Persistent Populate Filters for how Populate uses the **Filter field**.

Href Filter Field

Apply the specified expression to filter out the hierarchical references followed, when operating on a module recursively. Specify comma-separated object expressions, where an object expression takes the form:

```
<href_expression>
```

where the `<href_expression>` is a simple glob expression.

The expressions are simple leaf names; it is not possible to specify a form of a "path" in order to filter out a specific href in a hierarchy.

The `<href_expression>` matches are applied to both the href name and the target module name, with the href being excluded if either matches. This is regardless of how far down the hierarchy the href is.

It is not possible to specify that a single instance of an href in a hierarchy be excluded, unless that href has a unique name. For example, if module `Chip` is being populated, and module `Chip` references `CPU`, which references `ALU`, which references `STD_CELLS`, then the href filter match is against `STD_CELLS`. And not, for example, against `Chip/CPU/ALU/STD_CELLS`.

Notes:

- This field is only available when operating on module data.
- When Populating Your Work Area, see Setting Persistent Populate Filters for how Populate uses the **Href filter field**.

Local Versions Field

Note: This option only affects objects of a collection defined by the Custom Type Package (CTP). This option does not affect objects that are not part of a collection or collections that do not have local versions.

When it fetches an object, the populate operation first removes from your workspace any local version that is unmodified. (To remove a local version containing modified data, specify **Force overwrite of local modifications**.) Then the populate operation fetches the object from the vault (with the local version number it had at the time of checkin).

The **Local Versions** option specifies the action that the populate operation takes with modified local versions in your 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.)

Specify the **Local Versions** option with one of the following values:

- **Save local versions**. If your workspace contains a local version other than the local version being fetched, the populate operation saves the local version for later retrieval. For more information on retrieving local versions that were saved, see the ENOVIA Synchronicity Command Reference: localversion restore command.
- **Delete local versions**. If your workspace contains a local version other than the local version being fetched, the populate operation deletes the local version from your workspace.
- **Fail if local versions exist**. If your workspace contains an object with a local version number equal to or higher than the local version being fetched, the populate operation fails. This is the default action. A DesignSync administrator can change this default setting. For more information, see SyncAdmin Help: Command Defaults.

  Note: If your workspace contains an object with local version numbers lower than the local version being fetched and if these local versions are not in the DesignSync vault, the populate operation saves them. This behavior occurs even when you specify **Fail if local versions exist**.

Keys Field

You can control keyword substitution by selecting one of the following options from the **Keyword Substitution** drop-down menu:

- **Update values and keep keys** expands keyword values and retains the keywords in the file (default option). For example: $Revision 1.4 $
- **Update values and remove keys** expands keyword values but removes keys from the file. This option is not recommended when you check out files for editing. If you edit and then check in the files, future keyword updates are impossible, because the value without the keyword is interpreted as regular text. For example, 1.4.
- **Remove values and keep keys** keeps the keywords but removes keyword values. This option is useful if you want to ignore differences in keyword expansion, such as when you are comparing two different versions of a file. For example, $Revision: 1.4$
- **Do not update** keeps exactly the same keywords and values as were present at checkin.

By default, revision control keywords in your ASCII (but not binary) files are expanded during a checkout or populate operation. You can also expand keywords in local copies of files that you leave in your working directory during a checkin.

If you perform this operation using a mirror or cache directory, keywords are automatically expanded in the file that remains in the cache or mirror directory and the keyword itself remains in the file -- as if the **Update values and keep keys** option were used.

Related Topics

Revision Control Keywords Overview

Operate on Design Hierarchy

You can either operate on specific views or operate on the entire design hierarchy at once. When you operate on the entire design hierarchy, DSCC must identify the cells in a design hierarchy by scaning the hierarchy according to your specifications in the **Hierarchy Specification** fields.

DSCC begins scanning at the top-level cell views you specify using the **Objects** field. Then, it descends into the views indicated by the **Switch Using** field. You can use the **Switch Using** field to specify that the operation descend into one or more views you specify in a switch list (using the **Switch List** field). You can instead have DSCC descend into all instantiated views or all views that exist for a cell.

Use the **Stop List** field to indicate at which views DSCC is to stop scanning. DSCC also offers other hierarchy controls, such as limiting which libraries are scanned using the

Switch Libraries field and limiting which views are checked out using the **Process Views** field.

List Category Cells

When performing category operations using Category forms, you can specify multiple categories and optionally descend into nested categories. Your specification may match many cells. Before executing the operation, you can confirm which cells match your specification.

To list the cells that match your category specification, click **List Cells** in the Category form.

A text window displays all of the cells that match your selection. If you specified particular cell views, that information is listed in the display window as well.

Hierarchy List

When performing hierarchy operations, you have a number of ways of controlling which cell views will be operated on. Before executing the operation, you can confirm which cell views match your specification.

To list the cell views that match your hierarchy specification, click **List Cell Views** in the Hierarchy form.

There will also be an indication in the output if you selected **Cell Only** or **Cell And Library** from the **List Files** field.

Switch Using Field

> The **Switch Using** field lets you specify how the design hierarchy is to be traversed. At the end of each description is the equivalent API setting for the `-switchusing` option..

You can choose one of the following:

- **First in Switch List** -- As the design is traversed, DSCC descends into the first view specified in the switch list that exists for a cell. Specify the switch list using the **Switch List** field. (`firstswitchlist`) This is the default value for the API.
- **Instantiated View** -- As the design is traversed, DSCC descends into each instantiated view. The **Switch List** field is greyed-out and ignored in this case. Descending into all instantiated views is useful for traversing a hierarchy with layout-type views in cases where the design team has used multiple view names to indicate the layout views. (`instanciatedview`)

- **All in Switch List** -- As the design is traversed, DSCC descends into each view of the cell in the workspace that matches a view in the switch list. Specify the switch list using the **Switch List** field.

  Descending into all views in the switch list is useful when you have a well-defined set of views and you need to process multiple hierarchies. For example, you might have a layout hierarchy and a schematic hierarchy that are not the same, and you want to process both. If you do not select the **All in Switch List** field, this scan requires multiple passes through the command.(`allswitchlist`)

- **All Views** -- As the design is traversed, DSCC descends into each view of the cell in the workspace that exists for each cell. The **Switch List** field is greyed-out and ignored in this case.

  Descending into all views is useful when the command must traverse all paths through the hierarchy. In this case, DSCC might scan some unnecessary cells instantiated by obsolete views. (`allviews`)

Switch List Field

Specify the names of the views to be scanned to identify the design hierarchy. Separate view names by spaces.

When used with the API, you specify the switch list field as:

```
-switchlist <view> [<view>...]
```

The Switch List field is required if you specify the **First in Switch List** or **All in Switch List** options in the **Switch Using** field. If the **Switch Using** field is set to **Instantiated View** or **All Views**, the **Switch List** field is ignored.

Switch Libraries Field

The options in this field control which libraries may be entered as the hierarchy is scanned. At the end of each description is the equivalent API setting for the -`switchlib` option. You can choose one of the following:

- **All** -- All libraries may be entered. (`all`) Default value. Default is all When this option is selected, the Switch Library Names Field (`switchlibnames`) is ignored.
- **Only Into** -- Only the libraries specified in the following **Names** field may be (`only`)

- **Not Into** -- All libraries except those specified in the **Names** field may be entered. (`not`)

Switch Libraries Names Field

Specify a list of library names, separated by spaces, for use by the **Switch Libraries** field. This field is not active if **All** is selected in the **Switch Libraries** field.

When used with the API, you specify the switch list field as:

```
-switchlibnames <lib> [<lib>...]
```

Stop List Field

Specify the names of views at which the hierarchy scanning should stop. View names should be separated by spaces. As the design is traversed, if the **Switch List** view that is opened for a cell is also in this list, then scanning stops at that point.

When used with the API, you specify the switch list field as:

```
-stoplist <view> [<view>...]
```

This field is optional.

Process Views Field

Once you have identified the hierarchy using the **Switch Using**, **Switch List**, and **Stop List** fields, the **Process Views** options control which views of the identified cells are processed. At the end of each description is the equivalent API setting for the `-processviews` option. You can choose one of the following:

- **Switch View** -- Each view that was switched into is processed. (0)
- **All Switch List** -- All the views specified in the **Switch List** that exist for the cell are processed. (`0`) This is the default API value.

  Note: Switch lists are not used if hierarchy traversal descends into instantiated views. Thus, if **All Switch List** is selected and the **Switch Using** field is set to **Instantiated View**, an error occurs.

- **All Views** -- All views that exist for the cell are processed. (`1`)
- **Other** -- All the views specified in the Process Views Names field that exist for the cell are processed. For the API, the names are specified with the `-processviews` option followed by the list of views.

Process Views Names Field

Specify a list of views, separated by spaces, for use by the **Process Views** field. This field is active only when the **Other** option is selected in the **Process Views** field.

When used with the API, you specify the switch list field as:

```
-processviews <view> [<view>...]
```

Process Files Field

This option controls whether cell- and library-level files are processed in addition to the specified cell views. At the end of each description is the equivalent API setting for the `-processfiles` option You can choose one of the following:

- **None** -- No cell- or library-level files are processed.("") An empty string, indicating no cell or library-level files being processed is the default value for the API.
- **Cell Only** -- Cell-level files are processed, but library-level files are not. This option selects only cell-level files for those cells on which you are operating. (cell)
- **Cell And Library** -- Cell- and library-level files are processed.(library)

Command Invocation

The DesignSync command that will be invoked by the graphical interface is shown at the bottom of the dialog box. As fields are selected in the dialog box, the command options corresponding to each field in the dialog box are added or removed from the command invocation.

See the ENOVIA Synchronicity Command Reference for definitions of command options. The description of a command notes if the command is subject to access control.

Note: The command line defaults system only pertains to the command line interface. Underlying commands that are automatically invoked by the DesignSync graphical interface do not use the command line defaults system.

Command Buttons

These buttons appear on many of the forms in the DSCC integration. Not all buttons appear on all forms.

Button	Description
Help	This button invokes help information for the dialog. You can also invoke help by pressing **F1** at any time.
Defaults	Click this button to save the option settings that you have selected. The saved settings are displayed the next time you bring up the dialog box, and the settings persist from one

	DSCC invocation to the next. For more information on using the defaults option, see the *Custom Compiler Platform User's Guide* provided with Custom Compiler.
OK	When you click on the **OK** button, your settings are executed and the dialog closes.
Apply	Performs the operation without closing the form.
Cancel	When you click on the **Cancel** button, the dialog closes without executing any of the settings in the dialog.

Understanding the DSCC Popups

DesignSync CD features context-sensitive help which appears on the menus, forms, and popups defined within the interface. For more information on DesignSync CD, see DesignSync Data Manager for Custom Compiler Overview.

API Reference

dscd::add

dscd::add <*path*> [<*path*>...] [-modulecontext <*module*>] [-silent] [-options <*options*>]

=> {n_pass n_fail}

Description

Adds specified objects to the specified module.

Arguments

path	The path to the object(s) being checked in. This must be an absolute path.
-modulecontext	Full path to the module, module name or module instance name of the module the objects are being added to. If no modulecontext is specified, DesignSync uses smart module detection to determine the module context. For information on how smart module detection determines the target module, see Understanding Smart Module Detection.
-silent	Run silently. (Default)
-options	A single string containing the list of desired Add to Module options. The complete list of checkin options is detailed in the ENOVIA Synchroncity Command Reference: add Command. **Note:** Since -modulecontext field is required for add it should be specified with the -modulecontext option described above, and not included in the -options list.

Value Returned

Function returns a count of the number of objects successfully added, and the number which failed, unless there is an error processing the arguments.

dscd::branch

```
dscd::branch <branchname> <path>[<path>...] [-silent] [-options
<options>]

=> {n_pass n_fail}
```

Description

Creates a branch for the specified cell.

Arguments

branchname	Name of the branch being created. This must comply with all the naming requirements for branch names. For branch name requirements, see Branch.
path	The path to the object(s) being checked in. This must be an absolute path.
-silent	Run silently. (Default)
-options	A single string containing the list of desired Branch options. The complete list of checkin options is detailed in the ENOVIA Synchroncity Command Reference: branch Command.

Value Returned

Function returns a count of the number of objects successfully branched, and the number which failed, unless there is an error processing the arguments.

dscd::cancel

```
dscd::cancel
  <path>[...][-silent] [-options <options>]
=> {n_pass n_fail}
```

For operations on categories, the function is:

```
dscd::cancel_category <library> <category1> [<category2>...] [-
silent] [-recursive] [-options <options>]=> {n_pass n_fail}
```

If you are operating on a design hierarchy, DSCC provides a number of additional options to control the command.

```
dscd::cancel_hierarchy
  <library> <cell> <view> [<view2>...][-silent] [-options
<options>] [-switchusing <val>] [-switchlist <view…>[...]] [-
stoplist <view>[...]] [-switchlibchoice <val>] [-switchlibnames
<lib>[...]] [-processviews <val>] [-processfiles <val>]
=> {n_pass n_fail}
```

Description

Cancels the checkout of one or more file objects, categories, or cell views.

Specify absolute file names, paths, and libraries.

Arguments

path	The path to the object(s) being checked in. This must be an absolute path.
library	Name of the library at which to begin the check in operation. (for checkin_category and checkin_hierarchy functions) You can only provide a single library name.
category	Name of the category or categories at which to being the checkin operation. (for checkin_category function)
cell	Name of cell at which to begin the check in operation. (for checkin_hierarchy function)
view	Name of the view or views at which to begin the check in operation. (for checkin_hierarchy function)
-silent	Run silently. (Default)
-recursive	Traverses the category list, checking in the cells within the category and all subcategories.
-options	A single string containing the list of desired cancel options. The complete list of checkin options is detailed in the ENOVIA Synchroncity Command Reference: cancel Command.
-switchusing	See Switch Using Field.
-switchlist	See Switch List Field.
-stoplist	See Stop List Field.
-switchlibchoice	See Switch Libraries Field.
-switchlibnames	See Library Name Field.
-processviews	See Process Views Field.
-processfiles	See Process Files Field.

Value Returned

Returns a list of pass and fail counts; the first integer represents the number of checkouts successfully canceled and the second integer represents the number of failures. The function raises an error if argument checking fails. In all other failure cases, the function either raises an error or returns nothing.

dscd::checkin

```
dscd::checkin
  <path>[...][-silent] [-options <options>]
=> {n_pass n_fail}
```

For operations on categories, the API function is:

```
dscd::checkin_category <library> <category1> [<category2>...] [-
silent] [-recursive] [-options <options>]
=> {n_pass n_fail}
```

If you are operating on a design hierarchy, DSCC provides a number of additional options to control the command.

```
dscd::checkin_hierarchy
  <library> <cell> <view> [<view2>...] [-silent] [-options
<options>] [-switchusing <val>] [-switchlist <view…>[...]] [-
stoplist <view>[...]] [-switchlibchoice <val>] [-switchlibnames
<lib>[...]] [-processviews <val>] [-processfiles <val>]
=> {n_pass n_fail}
```

Description

Checks in one or more file objects, categories, or cell views.

Specify absolute file names, paths, and libraries.

Arguments

path	The path to the object(s) being checked in. This must be an absolute path.
library	Name of the library at which to begin the check in operation. (for checkin_category and checkin_hierarchy functions) You can only provide a single library name.
category	Name of the category or categories at which to being the checkin operation. (for checkin_category function)
cell	Name of cell at which to begin the check in operation. (for checkin_hierarchy function)
view	Name of the view or views at which to begin the check in operation. (for checkin_hierarchy function)
-silent	Run silently. (Default)
-recusive	Traverses the category list, checking in the cells within the category and all subcategories.
-options	A single string containing the list of desired checkin options. The complete list of checkin options is detailed in the ENOVIA Synchroncity Command Reference: ci Command. **Note:** When checking in new objects to a module, if the -modulecontext option is not explicitly provided, smart module detection is used to determine the target module. For information on how smart module detection determines the target module, see Understanding Smart Module Detection.
-switchusing	See Switch Using Field.

-switchlist	See Switch List Field.
-stoplist	See Stop List Field.
-switchlibchoice	See Switch Libraries Field.
-switchlibnames	See Library Name Field.
-processviews	See Process Views Field.
-processfiles	See Process Files Field.

Value Returned

Returns a list of pass and fail counts; the first integer represents the number of objects successfully checked in and the second integer represents the number of failures. The function raises an error if argument checking fails. In all other failure cases, the function either raises an error or returns nothing.

dscd::checkout

```
dscd::checkout
  <path>[...][-silent] [-options <options>]
=> {n_pass n_fail}
```

For operations on categories, the API function is:

```
dscd::checkout_category <library> <category1> [<category2>...]
[-silent] [-recursive] [-options <options>]
=> {n_pass n_fail}
```

If you are operating on a design hierarchy, DSCC provides a number of additional options to control the command.

```
dscd::checkout_hierarchy
  <library> <cell> <view> [<view2>...][-silent] [-options
<options>] [-switchusing <val>] [-switchlist <view…>[...]] [-
stoplist <view>[...]] [-switchlibchoice <val>] [-switchlibnames
<lib>[...]] [-processviews <val>] [-processfiles <val>]
=> {n_pass n_fail}
```

Description

Checks out one or more file objects, categories, or cell views.

Specify absolute file names, paths, and libraries.

Arguments

path	The path to the object(s) being checked in. This must be an

	absolute path.
library	Name of the library at which to begin the check in operation. (for checkin_category and checkin_hierarchy functions) You can only provide a single library name.
category	Name of the category or categories at which to being the checkin operation. (for checkin_category function)
cell	Name of cell at which to begin the check in operation. (for checkin_hierarchy function)
view	Name of the view or views at which to begin the check in operation. (for checkin_hierarchy function)
-silent	Run silently. (Default)
-recursive	Traverses the category list, checking in the cells within the category and all subcategories.
-options	A single string containing the list of desired checkout options. The complete list of checkout options is detailed in the ENOVIA Synchroncity Command Reference: populate Command.
-switchusing	See Switch Using Field.
-switchlist	See Switch List Field.
-stoplist	See Stop List Field.
-switchlibchoice	See Switch Libraries Field.
-switchlibnames	See Library Name Field.
-processviews	See Process Views Field.
-processfiles	See Process Files Field.

Value Returned

Returns a list of pass and fail counts; the first integer represents the number of objects successfully checked out and the second integer represents the number of failures. The function raises an error if argument checkout fails. In all other failure cases, the function either raises an error or returns nothing.

dscd::fetch_locked

```
dscd::fetch_locked <libname> [-vault]

=> {object}[{object}...]
```

Description

Reports the objects that are locked in the specified library or directory, returning a list of locked objects. You can specify whether to check for a lock on the objects in the local

workspace or in the vault. For vault objects, the list includes the object's branch and the time the object was locked.

Arguments

libname	Library name, workspace module, or directory path. (Required)
-vault	Check whether the vault versions of the objects are locked and, for module members, what branch of the vault object is locked.

Value Returned

For a workspace (when –vault is NOT specified), the command returns a list of the locked objects. Each entry in the return list is a list in the form:

{*libname cellname viewname filename*}. The filename is {} (which is null or "" in Tcl) for a locked view object. For a locked file the viewname is {} for a locked file, if the file is not in a cell, the cell is also {}.

Returns an empty string if none of the objects in the library or directory are locked or if the library or directory is not under revision control. The function raises an error if argument checking fails. In all other failure cases, the function either raises an error or returns an empty string.

Example

This example shows three locked objects, the view mylib/mycell/schematic, the file mylib/mycell/data.dm, and the file mylib/invCells.Cat.

```
dscd::fetch_locked mylib

=> {{mylib mycell schematic {}} {mylib mycell {} data.dm} {mylib
{} {} invCells.Cat}}
```

dscd::fetch_version

```
dscd::fetch_version <libname> <cellname> <viewname>
[<versionname>] [-silent] [-open] [-moduleversion]
```

Description

Fetches a version of an object from the vault and creates a temporary view of the object, without affecting the workspace version of the object. You can also choose to open the temporary view of the version as part of the fetch operation. The function fetches the version only if it is not already available in a temporary view. The library, cell and view names must be provided, and the view must be present in the workspace.

This function lets you compare two or more versions of the same cell view by creating a temporary, unmanaged copy of the specified version in your workspace. The name of the temporary cell view version is `<view>_v<version>`, where `<view>` is the name of the cell view, and `<version>` is the version number. For example, opening version 1.3 of cell view `layout` creates a temporary cell view called `layout_v1.3`.

Arguments

libName	Library name. (Required) The library must be present in the workspace.
cellName	Cell name. (Required) The cell must be present in the workspace.
viewName	View name. (Required) The view must be present in the workspace.
versionName	Version to be fetched. (Required)
-silent	Run silently (t). (Default)
-open	Open the cell view version as part of the fetch operation. By default, the version is not opened automatically.
-moduleVersion	Use the module version instead of the module member version. By default (-moduleVersion is unspecfied), the version specified is the module member vault version. This option is ignored for non-module objects.

Value Returned

If the version is fetched successfully, returns the version number passed in or, if a selector is passed in, the version that corresponds to the selector; otherwise, returns an empty string.

dscd::enable_debug

```
dscd::enable_debug 0|1
=> 1/0
```

Description

Turns debug mode on or off. With debug mode enabled, DSCC displays all commands sent to the stclc process and the output from those commands.

Arguments

`enable`	Enable debugging (1). Default. To turn off debugging, set `enable` to '`0`'.

Value Returned

Returns `1` if debug mode is successfully enabled (`enable` set to '`1`'). Returns `0` if debug mode is successfully turned off (`enable` set to '`0`').

execute_stcl API

```
dscd::execute_stcl <"command">

=> result
```

Description

Executes the specified Tcl command in the stclc process used by DSCC.

Note: DesignSync commands invoked via execute_stcl API do not use the command line defaults system. The command line defaults system only pertains to DesignSync command line shells.

Arguments

<command>	Tcl command to be executed along with any command arguments. (Required)

Value Returned

`result`	Displays the result of the function; whether the command succeeded or failed. It does not return the output of the command.

Example

The following example calls the DesignSync `ci` command to check in a file.

```
dscd::execute_stcl "ci -keep -nocomment myfile.txt"
```

dscd::tag

```
dscd::tag <tagname> <path> [<path>...] [-silent] [-options
<options>]

=> {n_pass n_fail}
```

For operations on categories, the API function is:

```
dscd::tag_category <tagname> <library> <category> [-silent] [-
options <options>]

=> {n_pass  n_fail}
```

If you are operating on a design hierarchy, DSCC provides a number of additional options to control the command.

```
dscd::tag_hierarchy <tagname> <library> <cell> <view>… [-silent]
[-options <options>]
[-switchusing <value>] [-switchlist <view…>] [-stoplist <view…>]
[-switchlibchoice <value>]
[-switchlibnames <library...>] [-processviews <value>] [-
processfiles <value>]

=> {n_pass  n_fail}
```

Description

Tags (or removes a tag from) one or more file object(s) in the local workspace. Tags (or removes a tag from) one or more categories in the local workspace. Tagging by categories is just like expanding the categories manually, getting the list of paths to the cells, and calling the dscd::tag function with that set of paths.

You must specify absolute path names.

Arguments

tagname	Tag to apply to the workspace versions of the objects. See Tag for tag naming guidelines. (Required)
library	Name of the library the category is contained within. You can only provide a single library name.
category	Name of the category or categories at which to being the tag operation.
`g_silent`	Run silently (`t`). (Default)
-options	A single string containing the list of desired tag options. The complete list of checkin options is detailed in the ENOVIA

	Synchroncity Command Reference: tag Command.
-switchusing	See Switch Using Field.
-switchlist	See Switch List Field.
-stoplist	See Stop List Field.
-switchlibchoice	See Switch Libraries Field.
-switchlibnames	See Library Name Field.
-processviews	See Process Views Field.
-processfiles	See Process Files Field.

Value Returned

Returns a list of pass and fail counts; the first integer represents the number of objects successfully tagged and the second integer represents the number of failures. The function raises an error if argument checking fails. In all other failure cases, the function either raises an error or returns an empty string.

Troubleshooting

Move Operation Not Supported

Description

You tried to move DesignSync managed data in the Library Manager and received a message saying the operation was not supported and a TCL error.

Situation

DesignSync does not support moving data in the Library Manager at this time.

Solution

To move data to a new location, you must copy the data to the new location and, once you have verified the copy, delete the data in the old location.

Turbo Format Not Supported

Description

You receive a message when you create a library that your library format is being changed from Turbo to FileSys.

Situation

DesignSync can only operate on data stored in a "FileSys" disk structure. It cannot operate on data stored in "Turbo" disk structure. If you create a library with the Turbo format, the library is automatically converted to FileSys format. The conversion process automatically creates a backup folder containing the original library.

Solution

Once the conversion has completed, there is no reason to retain the backup folder. You can safely delete it.

Related Topics

Overview

Incorrect Object Selected

Description

You receive a message when you select an operation but have not selected a valid object for the option.

Situation

Select an object and re-run the desired command.

Errors specified in the options selected for a hierarchical command

Description

You receive an error when attempting to run a command on a specified object hierarchy.

Situation

Review the options you have selected for the command and verify that they are correct. For more information on the options for the commands, see the specific command.

Related Topics

Checkin

Checkout

Cancel

Tag

View Could Not Be Fetched

Description

You cannot fetch the list of versions for the view.

Situation

The set of versions that exist for the selected object could not be fetched. There is a detailed error in the dialog box that explains what specifically went wrong..

Data Not Reloaded

Description

You cannot update the data from the server.

Situation

You have performed an operation (such as Checkout) that has modified the data on disc, but that data is not re-loaded into memory as the in-memory data is modified.

No Such Selector

Description

You receive a "No Such Selector" message.

Situation

You have selected an invalid selector for the object(s). Verify that the selector is in the appropriate form and is a valid selector for the selected object(s).

Invalid Report Mode Specified

Description

Your version history report has invalid report modes specified.

Situation

You have manually entered a value for report mode that does not exist. Review the report mode options available for the Report_mode and retry the command.

Copy/Move Not Allowed

Description

You receive a "Copy/Move Not Allowed" error message.

Situation

One or more objects on the copy, move, or rename operation is revision controlled by DesignSync, but was not locked. All files at the destination location of a cell or cellview copy, move, or rename operation must be locked.

Related Topics

Copying Objects

Moving and Renaming Objects

Resolving Conflicts

When copying, moving, or renaming cells or cellviews, if a destination object with the same name exists, the Custom Compiler conflict management window launches. Custom Compiler allows you to review the objects, overwrite the existing cell or cellview, or cancel the operation. For more information on resolving name conflicts, see the Custom Compiler help.

Getting Assistance

Using Help

ENOVIA Synchronicity Product Documentation provides information you need to use the products effectively. The Online Help is delivered through WebHelp® , an HTML-based format..

Note:

Use SyncAdmin to change your default Web browser, as specified during the ENOVIA Synchronicity tools installation. See SyncAdmin Help for details.

When the Online 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 givens 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 Synchronicity DesignSync Data Manager 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

Index