

ENOVIA Synchronicity DesignSync Data Manager

V6R2013



MW User's Guide

Copyrights and Trademarks

© Dassault Systèmes, 1994 - 2012.

All rights reserved.

PROPRIETARY RIGHTS NOTICE: This documentation is proprietary property of Dassault Systèmes. This documentation shall be treated as confidential information and may only be used by employees or contractors with the Customer in accordance with the applicable Software License Agreement.

Adaplet, Compliance Connect, DesignSync, ENOVIA, ProjectSync, Synchronicity, Team Central, ENOVIA Collaboration Platform, ENOVIA Business Process Services, ENOVIA Platform Server, ENOVIA Modeling Studio, ENOVIA 3D Live, FCS, AEF, Application Exchange Framework, Application Development Kit, ENOVIA V6X-BOM Engineering, ENOVIA Library Central, ENOVIA Materials Compliance Central, ENOVIA Variant Configuration, ENOVIA Program Central, ENOVIA Sourcing Central, ENOVIA Specification Central, ENOVIA Supplier Central, ENOVIA Collaborative Interference Management, ENOVIA Semiconductor Accelerator for Team Compliance, ENOVIA Aerospace and Defense Accelerator for Program Management, ENOVIA Apparel Accelerator for Design and Development, ENOVIA Automotive Accelerator for Program Management, ENOVIA Medical Device Accelerator for Regulatory Compliance, ENOVIA X-BOM Cost Analytics, ENOVIA X-BOM Manufacturing, ENOVIA Synchronicity DesignSync DFII, ENOVIA Synchronicity DesignSync MW, ENOVIA Synchronicity DesignSync CTS, ENOVIA IP Gear, IconMail, Imagemcon and Star Browser are either trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the US and/or other countries.

Oracle® is a registered trademark of Oracle Corporation, Redwood City, California. DB2, AIX, and WebSphere are registered trademarks of IBM Corporation. WebLogic is a registered trademark of BEA Systems, Inc. Solaris, UltraSPARC, Java, JavaServer Pages, JDBC, and J2EE are registered trademarks of Sun Microsystems, Inc. Windows XP and Internet Explorer are registered trademarks of Microsoft Corp. HP and HP-UX are registered trademarks of HP. All other product names and services identified throughout this book are recognized as trademarks, registered trademarks, or service marks of their respective companies.

The documentation that accompanies ENOVIA Synchronicity DesignSync products describes the applications as delivered by Dassault Systèmes. This documentation includes readme files, online help, user guides, and administrator guides. If changes are made to an application or to the underlying framework, Dassault Systèmes cannot ensure the accuracy of this documentation.

NOTE: This manual was generated directly from the online help with minimal reformatting. The PDF version is optimized for printing and does not contain active cross-reference links or animated use cases. Because the intent of the help is to be an online guide, there may be shortcomings in its organization and general usability as a printed document. The PDF version was created before limited changes were made to WebHelp. For the most current information, see the product's online help.

Dassault Systèmes ENOVIA
175 Wyman Street,
Waltham, MA 02451
Telephone 781.810.3500
Email: enovia.info@3ds.com
<http://www.3ds.com>

Table Of Contents

Release Information	1
Documentation	1
Selecting the appropriate release	1
Available Release-Specific Documentation	1
Locating the Release Specific Documentation	1
DS MW Overview	3
DesignSync MW Overview	3
Scenarios for Using DS MW	7
Introductory Scenario.....	7
Handoff between RTL and Back End.....	7
Isolation between the Top Block and Lower Blocks.....	8
Managing Collateral/Reference Libraries for Layout Build.....	8
Hazel Scripts Her Place and Route	8
Bob Purges Old Versions in Order to Conserve Space in the Vault	9
Bob Checks In a Cell So His Teammates Can Check It Out	9
Checking In and Checking Out All Views of a Cell	10
Ted and Glinda Check For Cells That Are Locked.....	10
Glinda Checks In a Library So Teammates Can Check It Out.....	10
Bob, Ted, and Glinda Share the Same Local Version	11
Bob Modifies His Local Version and Checks It In Again	11
Ted Relates His Local Version to the Vault Version to Make Sure He Is Working with the Right One	12
Glinda Rolls Back to an Earlier Cell	12

DesignSync Data Manager MW User's Guide

Boris and Daniel Collaborate on a Milkyway Library	12
Team Members Fetch a Cell That Is Not the Latest Local Version.....	14
Orson Checks Dependencies Among Milkyway Objects	15
Bob Wants to Know If His Cell Is Too Large to Check In.....	16
Hazel Keeps Her Technology File Elsewhere.....	16
Setting Up	19
Setting Up Your Environment	19
Managing the Technology File.....	20
Setting Up a Library	21
Specifying Dependencies Between Views.....	25
Checking In	27
Checking In a Library.....	27
Checking In a Cell.....	29
Checking Out	33
Checking Out a Library	33
Checking Out a Cell.....	36
Cancelling a Checkout	41
Canceling a Library Checkout.....	41
Canceling a Cell Checkout	43
Populating Your Workspace.....	47
Displaying a Cell's Data Sheet	49
Field Descriptions	49
Displaying a Cell's Version History.....	51

Field Descriptions 51

Tagging Libraries and Cells..... 53

 Tagging Libraries 53

 Tagging Cells..... 57

Exporting and Importing Library Information 65

 Exporting Library Information..... 65

 Importing Library Information..... 68

Purging Versions from the Vault..... 71

DS MW and DesignSync..... 73

 DS MW Command Line Interface 73

 How DesignSync Handles Milkyway Data 73

The Scheme API 77

 The stcl command 77

 Blocking Other DesignSync Commands..... 78

 The dbRebuildLibDS Command 78

 Putting a Library into Reference Control File Mode 79

Customizing 81

 Controlling the Check-Out Operation's Handling of Local Versions..... 81

Getting Assistance 83

 Using Help 83

 Accessing Product Documentation 84

 Contacting ENOVIA..... 85

 Getting a Printable Version of Help..... 85

DesignSync Data Manager MW User's Guide

DesignSync Glossary 87

Index 123

Release Information

Documentation

Release-specific information is located on the Dassault Systèmes support website in the Program Directory (<http://media.3ds.com/support/progdir/>). The Program Directory contains release-specific information for all major DesignSync releases beginning with V6R2009x.

Selecting the appropriate release

1. Open the Program Directory (<http://media.3ds.com/support/progdir/>). You may be required to enter your username and password to access information on the 3ds support site.
2. Select the following options in the top bar:

Select Line: **Version 6**
Select Level: **V6R2013**
Select Sub-Level: (use default)

Note: By default, the sub-level is always the most current version of the Program Directory files for the selected Level. There should never be a reason that information you need for a release is not in the most current version.

Available Release-Specific Documentation

The documents listed in the following table are available.

Product Enhancement Overview	Contains the list of new features and enhancements for the release.
General and Open Issues	Contains any known release issues, platform support information, platform configuration information, and system configuration recommendations for the release.
Closed Issues	Contains a complete list of closed issues for the release.
Installation	Installation instructions for DesignSync clients on all supported platforms. For server configuration information, see the <i>ENOVIA Synchronicity DesignSync Administrator's Guide</i> .

Locating the Release Specific Documentation

Product Enhancement Overview

1. In the left frame, select **ENOVIA** in the **Product Enhancement Overview** Section. This opens the Product Enhancement Overview index in the right frame.
2. Navigate to the IP Work-in-Progress section and select **Synchronicity DesignSync Data Manager**, or use your browser search functionality to search for **Synchronicity DesignSync Data Manager**. Selecting **Synchronicity DesignSync Data Manager** opens the Product Enhancement Overview for DesignSync.

General and Open Issues

1. In the left frame, select **ENOVIA** in the **General and Open Issues** Section
2. Navigate to the IP Work-in-Progress: Semiconductor EDA section and select **Synchronicity DesignSync Data Manager (SYN)**, or use your browser search functionality to search for **Synchronicity DesignSync Data Manager**. Selecting **Synchronicity DesignSync Data Manager (SYN)** opens the General and Open Issues for DesignSync.

Closed Issues

1. In the left frame, select **List of Closed Issues** in the **Closed Issues** Section.
2. Use your browser search functionality to search for Synchronicity. This will bring you to the section of the closed issues list that includes the following products:

Synchronicity DesignSync (including DSclipse, and DSVS plug-ins)
Synchronicity DesignSync Add-On for DFII
Synchronicity DesignSync Add-On for DSMW
Synchronicity DesignSync Add-On for DSCD
Synchronicity DesignSync Add-On for CTS
Synchronicity ProjectSync

Note: Not all releases include closed issues for all DesignSync products.

Installation

1. In the left frame, select **ENOVIA Server** in the **Installation** Section
2. Select **ENOVIA Synchronicity DesignSync Data Manager** in the navigation links at the top of the page or use your browser search functionality to search for **ENOVIA Synchronicity DesignSync Data Manager**.
3. Select the **Installing Synchronicity DesignSync Data Manager** link to open the Installation document.

DS MW Overview

DesignSync MW Overview

ENOVIA Synchronicity DesignSync® Data Manager MW(TM) (DS MW) is the integration of many DesignSync design management capabilities into the environment of Milkyway-based tools. A Milkyway-based tool reads and writes to a set of files that conform to the Milkyway database, as defined by Synopsys. DesignSync MW provides features to manage large, geographically dispersed projects. Users can access DesignSync MW capabilities from the Synchronicity menu in their Milkyway-based tool.

The Milkyway database is organized into libraries, views, and cells. In the past, this organization of Milkyway data made version control of cells impossible to accomplish without managing the entire library as a "black box". The only option for managing the data was to use the UNIX tar command to package the entire Milkyway library and then put the tar file under version control with your configuration management tool of choice.

In addition to the inability to control versions at the cell level, the management of a tar file of the entire Milkyway library has other disadvantages:

- A significantly larger amount of data was stored than was needed. For example, a company would have a seven Gigabyte library where only one Gigabyte was actually needed.
- There was no easy way to manage reference libraries. Once a user took a design from configuration management in order to reuse the design, that user was responsible for redirecting the reference libraries to the correct location. DS MW offers several levels of reference library management, the most elaborate of which allows users to identify the versions of a Milkyway library that depend on other Milkyway libraries.
- After the library was checked in, there was no way to integrate new changes to the technology file.

DS MW Functionality

DesignSync MW offers seamless version control at the cell or library level while meeting the needs of the Milkyway database.

With DesignSync MW, users can:

- Check in and check out a cell from the UNIX command prompt, the Synchronicity menu, the DesignSync GUI, or a DesignSync command shell.

A check-in operation includes all of the cell's attached files and a check-out operation retrieves those same attached files, managing them as a complete set

(or collection). This checkin/checkout mechanism makes it impossible for users to accidentally check in only a portion of the set, thereby corrupting it.

- Check in and check out a library from the UNIX command prompt, the Synchronicity menu, the DesignSync GUI, or a DesignSync command shell.

A check-in operation checks in to the library all locally modified cells and a check-out operation checks out all cells in the library.

- Display information that maps the local version to vault versions.

Users can select **Synchronicity => Cell Version History...** or use the DesignSync **vhistory** command to display all vault versions of a cell and list their corresponding Milkyway local versions. The display also shows which vault version resides in the user's workspace.

- Trim disk space by deleting old versions using the **purge** command.
- Execute an stcl command and have results appear in the transaction window of the Milkyway tool. (DS MW provides a scheme interface to stcl commands for this purpose.)

Users of the Milkyway tool can access DS MW through the Synchronicity menu.



In addition, DS MW provides the following functionality:

- DS MW provides functionality that, for the first time, enables multiple users to collaborate using their own private copies of the Milkyway library, much the same as RTL designers do with any configuration management system today.

This new form of collaboration coordinates the user's local experimental versions with the shared versions stored in the DesignSync vault.

- Check-in and check-out forms provide an option to include all views of the specified cell.
- When fetching a cell from the DesignSync vault, DS MW always fetches the latest local version. If the user's workspace contains a higher local version of a cell, the fetch operation moves it to a hidden directory, unless the version is unmodified and managed (under DesignSync revision control). In that case, the fetch operation deletes the redundant local copy.
- When a DesignSync checkout or populate operation fetches a new version of the technology file or library references, the operation merges the files into the library file.
- DS MW provides Synchronicity menu items for users to perform check-in, check-out, and cancel operations. Users can also use DesignSync commands (**ci**, **co**, or **cancel**) for these operations. To populate a workspace, users can use the DesignSync **populate** command.

For more information, see [How DesignSync Commands Handle Milkyway Data](#).

- The check-in operation fails quickly if there is not enough disk space.
- Limited dependency management. DS MW maintains a map of views to the other views they each depend on. Users can specify a cell view's dependencies with the **mw dependency** command. Then they can select the **Check dependencies** option on the **Cell Check In Options** form. For more information, see [Specifying Dependencies Between Views and Checking In a Cell](#).

Limitations/Restrictions

DesignSync MW does not provide the following functionality:

- DS MW does not provide a merge capability for Milkyway objects. DS MW users cannot work on the same cell concurrently and then merge their changes at check-in time, as they can when modifying RTL code.

Tip: When two or more users may be editing the same cell on the same branch at the same time, users should check out with a lock the objects that they want to modify.

- DS MW does not provide complete dependency management. A DS MW cell check-in operation does not determine the correct order in which to check in the views. However, you can specify dependencies between views and have the cell check-in operation perform a basic timestamp check on those views. If you have specified that one cell view depends on another, second view, and if that second view has a more recent timestamp than the first view, DS MW displays a message and the checkin does not proceed. (For more information, see [Specifying Dependencies Between Views and Checking In a Cell.](#))
- DS MW does not handle cell names that start with a hyphen (-). Use of other leading characters has not been tested, but cell names that start with the following leading characters are also suspect:

```
$ [ { % # @ ! ^ & * ( + - _ = ) ~ ` ] } : " > | , ' ?
```

For other restrictions on object names, see [URL Syntax](#) in [DesignSync Help](#).

- DS MW does not handle cell names or paths with spaces.
- DS MW does not support nested libraries. (A nested library is one whose root directory is a subdirectory of another library.)
- DS MW users should not put RCE revision control keywords (such as `$Revision$`) in library attached files. If a library attached file contains a revision control keyword, there is no guarantee that DS MW will handle the keyword properly. For information on revision control keywords, see [Revision Control Keywords Overview](#) in [DesignSync Data Manager User's Guide](#).

Supported Platforms

See the [Release Notes](#) for information on supported platforms and required patches or hot fixes.

Scenarios for Using DS MW

Introductory Scenario

Bob is a place and route engineer whose team is spread out over several sites. Bob receives a netlist from Francine, who leads the RTL team. He creates a place and route job using Francine's netlist and names the cell ALU. He generates three runs of the place and route, creating three local versions.

Handoff between RTL and Back End

This scenario shows how Francine's netlist reaches Bob and how he sends post-layout data back to Francine. (**Note:** This scenario shows a recommended practice, not a requirement.)

Francine organizes the RTL data so that in the directory for each block there is a subdirectory called `netlist`. Francine has a project in the DesignSync vault for each block's netlist. (In this scenario, the directory is called the netlist module.) In her RTL directory for the block, Francine creates a hierarchical reference to the netlist module. (To create the hierarchical reference, Francine can use either a DesignSync REFERENCE or HCM commands.) The hierarchical reference causes the netlist module to be populated into the `netlist` subdirectory.

Using ProjectSync's email notification, Bob registers to be notified each time a tag operation occurs on the netlist module. Francine checks in the netlist anytime she wants to preserve it. To maintain traceability between the RTL source and the netlist, she then uses the DesignSync tag operation to tag the netlist along with all of the RTL source files. The netlist may be used in a simulation or timing analysis before the team decides it is ready for place and route. When Francine tags the netlist with the tag `rdy4pr`, Bob gets an email notification from the tag operation and he then knows he has a new version for which to create a route.

Bob's Milkyway library is organized to have a netlist view directory (not the reserved NETL view). In reality, this netlist view directory is a hierarchical reference to Francine's netlist module. Bob directs his place and route scripts to perform VerilogIn on the netlist from this location.

The post-layout handoff works in a similar way. Alongside Francine's netlist directory is a post-route directory, which is a module managed by Bob. This module is also referenced in the Milkyway library by means of a hierarchical reference. Bob's scripts direct VerilogOut and other steps to place extracted files (that is, post-layout timing files) in this directory. Bob then checks in the files from the directory. Bob tags the post layout data along with the Milkyway objects. The tag operation causes an email notification to be sent to Francine.

To identify new data that the other has created, Bob and Francine can use the DesignSync **compare** command and look for objects that have a configuration version but no workspace version.

Isolation between the Top Block and Lower Blocks

Carlo is integrating several design blocks to form the place and route of the top of the chip; Bob's block is included in this integration. Bob makes some changes to his pin placement and block size that cause his abstract to change. But Carlo does not want to experience those changes, so before his company began using DS MW, Carlo would use the copy operation to copy Bob's block to a different name (for example, `BlockB_design_good`). Then Carlo would link his design to that renamed block. This method was error prone because Carlo had to remember to rename the block with its original name later on.

Now that his company uses DS MW, Carlo uses his own workspace for Bob's library. He can continue working with a known good configuration of that library until Bob releases a new version to him. In addition, Carlo avoids taking up costly disk space with this new workspace by using the DesignSync cache.

Users Work on Layout Integration Tasks at Functional Blocks and at the Mega Cell, Core, or Full Chip Level

Alice, Martin, and John share design hierarchy blocks throughout integration; they also share the same reference library. As the design integration/build gets refined there may be multiple intermediate releases or snapshots for verification tasks such as LVS, DRC, or NAC. By using the DesignSync vault as a repository and the tag operation to create releases and snapshots, Alice, Martin, and John can better coordinate the release/snapshot process.

Managing Collateral/Reference Libraries for Layout Build

Bella is responsible for building and releasing Libraries (standard cells, memories, I/ Os, PLLs, Oscillators, hierarchical design blocks, etc.) which internal customers use to perform floor planning, power grid and normal APR flow. Reference library changes have the potential for destabilizing designs and it is often necessary for Bella to keep several versions around. By using DS MW, Bella can enable design libraries to reference the correct version of the reference library. She can also save disk space by using the DesignSync cache. In addition, Library developers can also use the DesignSync tag operation to create library subsets targeted at different audiences.

Hazel Scripts Her Place and Route

It is 7 P.M. and Hazel is ready to join some friends for dinner. She starts a large place and route job, which is controlled by a scheme script read by Astro.

Since acquiring DS MW, Hazel has improved her scripts. Previously, to save good states of the design, the script saved the cell as `cellname_stage` after every significant stage in the design. With Hazel's improvements, the script saves good states of the design by performing a save of the cell (without creating a new cell name), then calling a check-in operation on the original cell name, followed by a tag operation.

After dinner, Hazel goes back to work to find that her place and route job failed. The last good state was after the clock tree synthesizer operation had finished. To get back to a known good state, Hazel performs a checkout with a version selector of `cts` (the tag that the script applied). She then fixes the problem, starts another job, and heads for home.

Bob Purges Old Versions in Order to Conserve Space in the Vault

After several checkins of a cell, Bob determines that he does not need to keep many of his old versions. He uses the **purge** command to delete these old versions.

Bob Checks In a Cell So His Teammates Can Check It Out

Bob and other Place and Route team members use DesignSync to check in and check out files; they now want to perform the same operations on Milkyway cells. Specifically, they want to:

- Check in cells when the team has arrived at good results it wants preserved
- Check in cells when team members have received a new netlist
- Check in and check out objects when team members want to share objects with others

Bob now wants to be able to use a GUI or command line to check in the ALU cell. The check-in operation should check in only the latest local version of ALU (version 3, for example) and ignore earlier versions (1 and 2). After the checkin, Bob's cell should be available for Bella (a user at another site) to check out.

Before Bob can check in the cell for the first time, he uses the DS MW **Library Setup Options** form to associate his library with a DesignSync vault and ensure that his technology file and references to his reference libraries are put in the vault. By setting up the library in this way, Bob makes it possible for Bella to recreate Bob's cell at another location, in isolation from the rest of Bob's libraries.

Bob and Bella determine ahead of time where the reference libraries will reside at each of their sites. In the **Library Setup Options** form, Bob specifies his reference libraries either as relative paths or in relation to an environment variable that he and Bella have agreed upon. Before Bella checks out the cell for the first time, she also uses the

Library Setup Options form to specify the same vault location where Bob checked in the cell.

Checking In and Checking Out All Views of a Cell

When Bob is ready to check in a cell, he selects **Synchronicity => Cell Check In....** To check in all views of the cell, he clicks **All** in the Views section. The check-in operation checks in all views that were checked out for that cell in the current library. To check out all views of a cell, Bob can click **All** in the Views section of the **Check Out Options** form.

Ted and Glinda Check For Cells That Are Locked

Ted needs to make changes to some of Glinda's cells. He first wants to make sure that neither Glinda nor anyone else has those cells already checked out with a lock. Ted changes directory to the root directory of his Milkyway library and from the console window of his Milkyway tool, he uses an stcl command:

```
stcl "scd /users/Ted/Data/Libraries/macro_cell_library"  
stcl "ls -recursive -locked -report U [url vault .]"
```

The command output lists cells or other files currently locked by Ted or other users. From the output, Ted sees that Glinda has locked one of the cells he wants to change. He sends her email asking her to check in the cell so he can check out the cell with a lock and make his changes.

Glinda has completed her work and wants to make sure that everything she has been working on has been checked in to the vault. To see if she has any cells or other files in her workspace that are locked, she changes directory to the root of her workspace and from the console window of her Milkyway tool, she uses an stcl command:

```
stcl "scd /users/Glinda/WorkProjects"  
stcl "ls -recursive -locked -workspace"
```

The command output lists cells or other files that Glinda has that are still checked out to her workspace with a lock. (**Note:** Using the **-workspace** option gives Glinda a quick listing of her locked objects; the option does not show objects locked by other users.)

She views the output and sees that she has two cells still locked. She uses DS MW to check in the cells. The check-in operation also releases the locks.

Glinda Checks In a Library So Teammates Can Check It Out

Glinda is working with Bob on the library that contains the ALU cell. She works on other cells as well and likes to stay current with Bob's latest work on ALU. All of the operations he wants to perform on cells (described above), she wants for the whole library. Before she can share her library with her teammates, she must use the DS MW **Library Setup Options** form to associate her library with a DesignSync vault.

One of Glinda's teammates, Ted, is interested only in a set of 10 cells from Glinda's library. To identify those 10 cells for Ted, Glinda uses DesignSync to tag the cells with the tag `for_Ted`. Ted uses the tag to populate only those 10 cells to his workspace with the **populate -version** command. (Ted also can use the **Library Check Out Options** form and specify the `for_Ted` tag in the **Version** field.) Ted sees this method as a big improvement over his earlier method of copying Glinda's entire directory tree.

Ted uses DesignSync (List View or the **ls** command) to determine if Glinda has checked in modifications to any of the cells that he is interested in.

Ted appreciates how large Milkyway objects are, so he has configured DesignSync to use its cache. Whenever he populates his local workspace with Glinda's libraries (using **populate -share**), he is fetching only a symbolic link to each of her cells, not a copy.

Bob, Ted, and Glinda Share the Same Local Version

After his third run of the ALU place and route, Bob checked in local version 3, which is version 1.1 in the DesignSync vault. For the checkin, Bob selected **Leave Behind... Locked Data (checkin -lock)**, so that he can keep working and create more local versions if he chooses.

Ted checks out the version 1.1 (read only, no lock) to his workspace. In Ted's workspace, DS MW identifies the cell as local version 3.

Bob continues working on the ALU cell, creating two more local versions (4 and 5). Then he checks the cell in again. The check-in operation checks in local version 5 to the DesignSync vault, creating vault version 1.2.

Ted again checks out the ALU cell from the vault. The operation deletes local version 3 from his workspace, replacing it with local version 5. (DesignSync deals only with the latest local version -- in this case, version 5 -- and shows the cell as vault version 1.2.)

Glinda checks out (no lock) the entire library. The operation fetches only the latest version of the ALU cell (local version 5, which is vault version 1.2).

When Bob, Ted, and Glinda refer to the ALU cell, they are always able to refer to the local version in their individual workspaces. So far, this is version 5 for all three of them.

Bob Modifies His Local Version and Checks It In Again

After a time, Bob decides to run place and route again. His place and route script causes local version 5 of the ALU cell to be modified, instead of generating local version 6. He checks in the modified version 5, creating vault version 1.3. Until Ted and Glinda check out the cell again, Bob has a local version 5 different from theirs. When Ted and Glinda check out the cell, the operation fetches vault version 1.3, which is the modified local version 5 that Bob checked in.

Ted Relates His Local Version to the Vault Version to Make Sure He Is Working with the Right One

Bob and Ted are both using local version 5 of the ALU cell. However, they each have a different local version 5 because Bob modified his local version 5 (see *Bob Modifies His Local Version and Checks It In Again*).

Using DesignSync **List View** or the **ls** command, Ted can see that the ALU cell he has in his workspace is not the latest version in the vault. But he wonders how that can be when he has the same local version as Bob.

Ted uses the DS MW **Cell Version History Options** form to show the local version number for each vault version of ALU. (He also can use the DesignSync **vhistory** command to show the same information.) The version history operation shows the current vault version of ALU and lists each version of the object in the vault, along with its version tags.

From the output of the version history operation, Ted sees that ALU's current vault version is 1.3 and that version 1.3 in the vault has the tag `MW_5`, identifying it as local version 5. Since Ted's local version 5 is only vault version 1.2, he knows he must update his workspace to get the latest vault version.

Glinda Rolls Back to an Earlier Cell

Bob performs another run of place and route on ALU, creating local version 6. Using DS MW, he checks in that version, creating vault version 1.4. Glinda fetches vault version 1.4 from the DesignSync vault but then hears from Bob that everyone needs to go back to local version 5.

Glinda knows that to get back to local version 5 in her workspace, she needs to check out the DesignSync version that matches local version 5. To fetch local version 5, she can use **co** command with its **-version** option (`co -version MW_5`) or she can use the **Cell Check Out Options** form and type `MW_5` in the **Version** field.

DS MW fetches the Latest vault version that matches local version 5.

Boris and Daniel Collaborate on a Milkyway Library

Boris and Daniel are collaborating on a Milkyway FRAM library, each using a workspace at a different site. This will become a reference library for future design libraries.

Daniel has fetched three cells from the DesignSync vault to his workspace and is working with those cells. Boris adds 30 new cells to the library and checks them in. However, Daniel is not ready to fetch all those cells yet, so he doesn't check out the library or populate his workspace.

Boris expresses the fact that cell A and cell B are logically equivalent. He wants Daniel to have this information, so he exports his Other Library Information to the vault.

Boris can perform the export in either of two ways:

- If he has changed only the Other Library Information, he can use the **Export Library Information** form and select the option **Export "other" library information (lib file and attached files)**.
- If he has changed some cells as well as Other Library Information, he can put both kinds of data back into the vault using the **Library Check In Options** form. On that form, he can select the **Export "other" library information prior to checkin** option.

Boris uses the **Export Library Information** form to perform the export. After the export is done, he lets Daniel know about the export.

Daniel needs the information about the logical equivalence of cell A and cell B, so he imports the Other Library Information into his workspace (using the **Import Library Information** form). When the import operation completes, Daniel can see the logical equivalence of cells A and B, but he does not see the 30 cells Boris added. (The import Other Library Information operation updates only the Other Library Information. The operation also erases any local changes to the technology data and the list of reference libraries. The operation has no effect on the set of cells in Daniel's workspace.)

When he is ready to work with the 30 cells that Boris added to the reference library, Daniel can populate his workspace (with the **populate** command) or use DS MW to check out the library.

If Daniel wanted to fetch the 30 new cells at the same time that he imported the Other Library Information, he can use the **Library Check Out Options** form. On the form he would select the **Force overwrite of local modifications** option. He can also use the **populate -force** command.

Note: Since the force option replaces Other Library Information in the workspace with information from the vault, these methods should be used with care. See Boris Becomes Controller of the Other Library Information.

Boris Becomes Controller of the Other Library Information

From their collaboration on the FRAM library, Daniel and Boris see that they need to control changes made to Other Library Information. For example, suppose that Daniel had expressed the logical equivalence of cells C and D before he imported Other Library Information from the vault. After the import, the information about the logical equivalence of cells C and D would be lost from the library in Daniel's workspace because the import operation replaces it with the Other Library Information from the vault (in this case, the logical equivalence of cells A and B).

To prevent inadvertent removal of Other Library Information from either of their workspaces, Daniel and Boris decide on a plan for collaboration on the library. While they both can contribute cells to the library, Boris will control the export of Other Library Information by checking out the library's shadow file (`shadow.sync.mw.lib`) with a lock (using the DesignSync **co-lock** command). This action prevents Daniel and other users from exporting the library's Other Library Information to the vault until Boris unlocks the file.

As controller, Boris can choose to make all of the changes to the Other Library Information. He periodically exports the Other Library Information, keeping that information current in the vault.

Boris can also unlock the library's shadow file so that Daniel can export Other Library Information to the vault. To unlock the file, Boris cancels the checkout of the `shadow.sync.mw.lib` file (using the DesignSync **cancel** command).

Team Members Fetch a Cell That Is Not the Latest Local Version

Day One:

Darien uses the DesignSync tag operation to tag a Milkyway library with a tag called Gold. The tag operation tags the adder cell's vault version that corresponds to local version 3 in Darien's workspace. Subsequently, local version 5 of the adder cell is checked in, creating a vault version that is now the latest version in the DesignSync vault.

Serena intends to populate the design using the Gold tag but she forgets to specify Gold as the version selector for the **populate** command. So the populate operation fetches local version 5 of the adder cell (the latest version in the vault) to Serena's workspace.

Realizing her mistake, Serena populates her workspace again, this time specifying Gold as the version selector (`populate -recursive -version Gold`). The populate operation deletes local version 5 of cell adder from Serena's workspace (because local version 5 is under revision control and can always be refetched from the vault). Now

Milkyway-based tools recognize local version 3 (the earlier local version of the adder cell) as the default cell to use.

Day Two:

Hazel creates a new workspace and uses the DS MW **Library Setup Options** form to associate it with the DesignSync vault location for Darien's library. Then Hazel uses a brand new netlist and starts the route for the adder cell all over again from scratch, creating local versions 1 and 2. Hazel checks in the adder cell, using the **ci** command with the **-skip** option to make local version 2 the latest in the vault.

Meanwhile, Darien has local versions 1, 2, 4, and 5 of the adder cell in his workspace. He has not checked in local version 1, 2, or 4, but he has checked in local version 5, creating vault version 1.2. Now Darien checks out the adder cell, using the **Force overwrite of local modifications (-force)** option of the **Cell Check Out Options** form. The check-out operation deletes local version 5. (Because it is under revision control, it can always be refetched from the vault.) In addition, the check-out operation moves local versions 1, 2, and 4 into a hidden subdirectory called **.MWcells**. (**Note:** If Darien uses **co -report verbose** to perform the checkout, the output of the command includes a message about the deletion and the move.)

Darien can choose to reinstall the hidden local versions or he can delete them. If he chooses to reinstall a local version from the hidden subdirectory, he can use the DesignSync **localversion restore** command. (For information, see the **ENOVIA Synchronicity Command Reference: localversion**.) Darien likes having the choice to reinstall local versions.

Day Three

Unlike Darien, Hazel would rather that the check-out operation fail than move the cells into a hidden subdirectory in her workspace. Having the checkout fail gives her the choice of deleting the cells up front or moving them. So Hazel modifies a DesignSync registry setting for **SaveLocal** to change the default behavior of checkout to the failure behavior. Darien leaves the default behavior as is. (For information, see Controlling the Check-out Operation's Handling of Local Versions)

Day Four:

Hazel reruns a route, creating a modification to the adder cell's local version 2. Later on, she tries to fetch to her workspace a version of the adder cell that Darien checked in (local version 3). However, the check-out operation fails because Hazel has a locally-modified copy. She performs the checkout again, using the **Force overwrite of local modifications** option, which deletes her local changes.

Orson Checks Dependencies Among Milkyway Objects

Often in the use of Milkyway-based tools dependencies are formed among views of a cell. For example, Orson uses the Hercules tool to perform a run on a cell named `route`; the run produces a FILL view of the cell. The FILL view of `route` depends on the CEL view; if the FILL view is checked in, the CEL view should be checked in as well.

Orson specifies the dependency between the FILL and CEL views by adding the **mw dependency** command in a `.tcl` startup file that is executed at DesignSync startup time. (For more information, see *Specifying Dependencies Between Views*.) For example, to specify the FILL view dependency on CEL, Orson adds the following line to the `.tcl` startup file:

```
mw dependency FILL -dependson CEL
```

Orson decides to check in the FILL and CEL views of `route`. He uses the DS MW **Cell Check In Options** form, specifying the `route` cell for checkin and selecting the FILL and CEL views. By default, the **Check dependencies** option is also selected. When Orson clicks **OK** to begin the checkin, the check-in operation compares the timestamp of the CEL view (on which FILL depends) to the timestamp of the FILL view. If the CEL view is newer (has a more recent timestamp) than the FILL view, DS MW displays a message and does not proceed with the checkin. If the CEL view is older than the FILL view, the check-in operation proceeds. In this instance, Orson's checkin succeeds because the FILL view is newer than the CEL view.

Orson continues place and route of the `route` cell. Before running Hercules to produce another fill view, he uses the DS MW **Check In Options** form to again check in the FILL and CEL views. Because the CEL view is now newer than FILL, DS MW displays a message and does not proceed with the checkin.

Bob Wants to Know If His Cell Is Too Large to Check In

At the start of a check-in operation, Bob wants to know whether his ALU cell can fit into the vault. If there is not enough room in the vault, the checkin fails before sending the file to the server.

Hazel Keeps Her Technology File Elsewhere

Note: The following scenarios relate to various situations in which one controlled technology file is used for multiple Milkyway libraries. Method 1 and Method 2 assume that the technology file is being managed as a separate file in the library. See *Managing the Technology File* for information.

Hazel already has a place on her LAN where she keeps the master technology file for a library she is about to create. She does not want to be forced to copy this file into every Milkyway library without any link back to the master. There are three methods that Hazel can use:

Method 1: Use Symbolic Links to Keep the tech.tf File Elsewhere

To keep the `tech.tf` file in another location, users can also use DesignSync symbolic links. For example, suppose that in the `tech` directory of her library, Hazel creates a symbolic link named `tech.tf` which resolves to the place where the actual `tech.tf` file resides. Hazel checks in the `tech.tf` symbolic link.

Note:

- This scenario pertains to a situation in which all designers on the project are located on the same local area network (LAN).
- For Hazel to use the symbolic link method, her team leader first must ensure that SyncAdmin Symbolic Links Option is set to either **Store links to files as links...** or **Store both links to files and directories as links....**

Checking in the link gives Hazel more flexibility but also results in some loss of functionality. If she uses the symbolic link method, DesignSync is not aware of technology file changes from one populate operation to the next and so the changes are not integrated into the lib file until the next populate. Therefore, Hazel must be responsible for periodically making sure that her teammates incorporate the `tech.tf` file changes on a regular basis. She can do this by checking in the unchanged symbolic link using the **-force** option. This action makes DesignSync aware of changes in the `tech.tf` file and DesignSync then integrates the changes into the users' local library files.

In contrast, when the technology file physically resides in the library's `tech` directory (or when there is a module that includes the file, as described above), team members experience a more seamless update of technology file changes. The more seamless update occurs because each time team members populate their libraries, the **populate** operation integrates the `tech.tf` file changes into each person's local library file.

Method 2: Use Replace Tech File and DS MW Export

There is a third way for Hazel to keep her technology file in a location other than the library's `tech` directory. (**Note:** This method relies on the fidelity of Synopsys' Dump technology file operation.)

With this third method, the technology file can reside anywhere on the LAN; the file does not need to reside in a `cmosX` directory. All that matters is that Hazel knows the location of the file. (One approach might be to use DesignSync to manage this file in a DesignSync vault. Then Hazel could use ProjectSync to register for email notification when the file changes).

When the technology file for `cmosX` changes, Hazel uses Synopsys' Replace Tech File command and then uses DS MW to export either the technology file or the Other Library Information. (See Managing the Technology File for information.) The export operation

regenerates the `tech/tech.tf` file. When Hazel or her teammates populate the library into their workspaces, the **populate** operation causes the library to reread the correct technology file.

Setting Up

Setting Up Your Environment

Setup Tasks for Administrators or Project Leaders

The SyncServer installation includes DS MW. To run DS MW, you need to obtain a license.

Note: The installation operation sets the Vault Type registry setting to select Smart Vault as the vault type for objects with a colon (:). For more information, see [How DesignSync Handles Milkyway Data](#).

Before you can use DesignSync MW, you must set up the DS MW environment. (For information about DS MW functionality, see [DesignSync MW Overview](#).)

Note: Typically, a DesignSync administrator or a project leader performs these steps. For the steps individual users need to perform, see [Setup Tasks for Users](#).

To set up the DS MW environment:

1. If you have not already done so, enable DesignSync recognition of Milkyway objects. You can enable recognition of Milkyway objects in one of two ways:
 - o To enable recognition during installation, select [5] Configure for use with Synopsys' Milkyway tools and data.
 - o To enable recognition after installation, use the **Third Party Integration** panel of the SyncAdmin tool. For information, see [Third Party Integration Options in SyncAdmin Help](#).
2. Add lines to the `.avntrc` file to source the `<SYNC_DIR>/snps/scm/dsInit.scm` file.

Installation creates a new directory, `<SYNC_DIR>/snps/scm`. The directory contains a file, `dsInit.scm`. As DesignSync administrator, you must source the `dsInit.scm` file from the central `.avntrc` file. To source the file, add these two lines to the `.avntrc` file:

```
(load (string-append (getenv "SYNC_DIR") "/snps/scm/dsInit.scm"))
(dsInit)
```

For your convenience, DesignSync includes a sample file (`.avntrc.example`) from which you can copy the lines to your `.avntrc` file. The location of the file is: `<SYNC_DIR>/snps/scm/.avntrc.example`

3. Optionally, specify dependencies between cell views.

You specify dependencies between views so they can be checked each time users perform a checkin of a cell. Typically, a DesignSync administrator or project leader specifies dependencies for the team working on the project. See [Specifying Dependencies Between Views](#) for information.

4. Set up libraries for initial checkin. For information, see [Setting Up a Library](#).
5. Customize DS MW as needed and perform optional set-up tasks.
 - In SyncAdmin in the **Default Fetch State Options** pane, make sure that the **Default fetch state** is set to either **Unlocked copies**, **Links to cache**, or **Links to mirror**. See [SyncAdmin Help: Default Fetch State Options](#).
 - The DesignSync reference functionality is not available in DesignSync MW. If your project leader has defined the default fetch state as Reference, then the Setup Library, Import and Export operations use a fetch mode of Get/Keep (local copy) and the forms for these operations default to Get/Keep as the default state.

Note: Although SyncAdmin allows Reference and Lock Reference to be selected as valid states for a form, DS MW forms never display these states.

- Optionally, change the default registry setting to control the check-out operation's handling of higher local versions. See [Controlling the Check-Out Operation's Handling of Local Versions](#).
- Optionally, you can determine the options users can select for object state upon checkin, checkout, or cancellation of a checkout. See [SyncAdmin Help: States Options](#).
- Optionally, you can determine how DS MW will manage the technology file (`tech.tf`). By default, DS MW manages this file as separate file and not part of the Other Library Information. As DesignSync administrator, you can change this default setting. See [Managing the Technology File](#) for information.

The DS MW environment is now ready for users to set up their workspaces and use DS MW to check in and check out libraries and cells.

Setup Tasks for Users

1. Use the **Library Setup Options** form to prepare your workspace for initial checkout of libraries and cells. See [Setting Up a Library](#).
2. Optionally, change the default registry setting to control the check-out operation's handling of higher local versions. See [Controlling the Check-Out Operation's Handling of Local Versions](#).

Managing the Technology File

DS MW provides the ability to manage the technology file (`tech.tf`) as a separate entity or as part of the Other Library Information. As DesignSync administrator, you can

specify how DS MW manages the technology file by using the **Manage technology file separately** option on the SyncAdmin Third Party Integration form. By default, this option is enabled.

In most cases, the technology file can be managed separately. However, one reason to manage the technology file as part of the Other library information is that the technology file may be using features available in the latest Synopsys software; however, the technology file read/write software may not recognize these features. In such a case, it is necessary to manage the technology file as part of the Other library information by disabling the **Manage technology file separately** option.

If the setting is disabled:

- When you perform a library setup operation, DS MW does not capture the technology file information in an ASCII file (`tech/tech.tf`) and check in the file to the vault.
- When you perform a library setup operation to fetch a library from the vault for the first time, DS MW does not fetch the technology file.
- The Export Library Information form does not show the Export technology file option.
- The Import Library Information form does not show the Import technology file option.
- Checking out an existing `tech.tf` does not automatically import that file into the library.

Related Topics

SyncAdmin Help:Third Party Integration Options

Setting Up a Library

Before populating or checking out Milkyway objects to your workspace for the first time or before checking in workspace objects for the first time, you must use the **Library Setup Options** form to set up your library. In addition, you can use this form if you want to change the paths to Reference Libraries.

If you use **Library Setup Options** on an empty workspace, the library setup operation associates the workspace with the DesignSync vault. In addition, the library setup operation also imports the technology file (`tech.tf`), the list of reference libraries (`refs.txt`), and Other Library Information to your workspace, if any such information has been exported to the vault. (**Note:** DS MW does not import the technology file if the **Manage technology file separately** option on the SyncAdmin Third Party Integration form is disabled. See Managing the Technology File for information.)

If you use **Library Setup Options** on a workspace containing Milkyway objects that you want to check in for the first time, the library setup operation not only associates the workspace with the DesignSync vault but also:

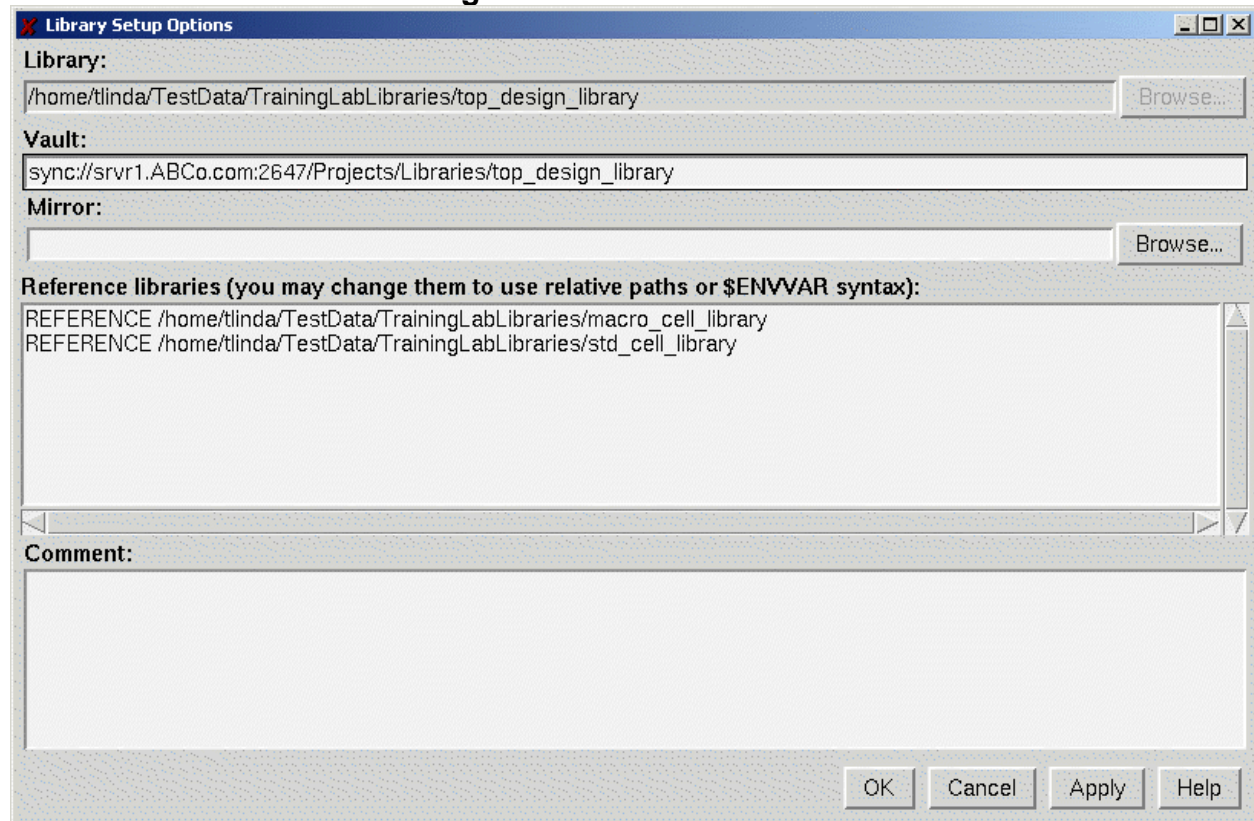
- Captures the technology file information in an ASCII file (`tech.tf`) in the `tech` directory and checks in the file to the vault. (**Note:** DS MW does not perform this action if the **Manage technology file separately** option on the SyncAdmin Third Party Integration form is disabled. See *Managing the Technology File* for information.)
- Captures the Reference Library information in an ASCII file named `refs.txt` and checks in the file to the vault.
- Captures Other Library Information in shadow files and checks in those files to the vault.

Once the library is set up, you can populate or check out Milkyway objects to your workspace or check in Milkyway objects from your workspace to the vault.

To set up a library:

1. Select **Synchronicity => Library Setup...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **Apply** or **OK**.

Click the fields in the following illustration for information.



The screenshot shows a dialog box titled "Library Setup Options". It contains several input fields and a list of reference libraries. The fields are:

- Library:** `/home/tlinda/TestData/TrainingLabLibraries/top_design_library` (with a "Browse..." button)
- Vault:** `sync://srvr1.ABCo.com:2647/Projects/Libraries/top_design_library`
- Mirror:** (empty field with a "Browse..." button)
- Reference libraries (you may change them to use relative paths or \$ENVVAR syntax):**
 - REFERENCE `/home/tlinda/TestData/TrainingLabLibraries/macro_cell_library`
 - REFERENCE `/home/tlinda/TestData/TrainingLabLibraries/std_cell_library`
- Comment:** (empty text area)

At the bottom right, there are four buttons: **OK**, **Cancel**, **Apply**, and **Help**.

Field Descriptions:

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Note: If you have a library open, you can set up only that open library. To set up a different library, you must first close the library currently open and then open the other library.

Vault:

Type the vault URL in this field. If a vault association has been set up for the library or a parent directory and if there is a `lib` file present in the library, DS MW displays the vault URL for the library in the **Vault** field.

Note: If you are working in a multi-branch environment, you can optionally specify the vault with a branch selector by following the vault URL with `@<selector>`. For example:

```
sync://srvr1.ABCo.com:2647/Projects/Libraries/top_design_library@Rel2:Latest
```

For information on selectors, see [DesignSync Data Manager User's Guide: What Are Selectors?](#).

Mirror:

Use this field to associate your library directory with a mirror directory. **Note:** Before you can associate your library directory with a mirror, a DesignSync administrator or project leader must set up the mirror directory and associate it with the DesignSync vault.

Click **Browse...** to select a mirror path or type the path to the mirror in this field.

When you click **Apply** or **OK**, DS MW associates the mirror directory with your library directory.

Comment:

Use this field to specify a comment that will be attached to the specified objects when they are checked in.

Note: Your project leader may require that every checkin have a comment of a minimum length.

Reference libraries:

When you select **Synchronicity => Library Setup...**, the **Library Setup Options** form does not display this field until you click **Browse...** and select a library from the Library Browser.

After you select a library from the Library Browser, if there is a library file, DS MW displays information about the Reference libraries in the **Reference libraries** field.

The **Reference libraries** field lets you change the prefix of the reference libraries. For example, suppose it is a practice in your company that at all sites the reference library is located next to an individual user's library. Because the location is the same from site to site, you can change your reference library path from:

```
REFERENCE /disk1/libraries/reflibs/cmos09
```

to:

```
REFERENCE ../reflibs/cmos09
```

You can also use an environment variable in the path to reference libraries. For example, your project leader might define the `MW_REFLIB` environment variable as a path where all reference libraries for a project reside. Then you can use the variable to specify the path to your reference library as:

```
REFERENCE $MW_REFLIB/reflibs/cmos09
```

Note: If a library already exists in your workspace and is in reference control file mode, the Reference Libraries cannot be edited.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Related Topics

Populating Your Workspace

Checking In a Library

Checking In a Cell

Specifying Dependencies Between Views

In the design process, the data of one view can depend on the data in one or more other views.

Using the **mw dependency** command, you can specify these dependencies so that DesignSync recognizes them. Then when you check in a view, you can specify that the check-in operation perform a dependency check. This check compares the timestamp of the view you are checking in to the timestamps of other views on which the first cell depends. The checkin fails if any of those views are newer than the view you are checking in. The checkin also fails if any of those views is not also specified for checkin.

You can also use this command to display a list of views that a specified view depends on.

To specify the dependencies of a view:

1. Create a .tcl startup file or open your existing .tcl startup file.
2. To the file, add the **mw dependency** command for each view's set of dependencies that you want to specify.

For example, to specify that the FILL view depends on the data in the CEL view, you would specify the following command in a .tcl startup file:

```
mw dependency FILL -dependson CEL
```

3. In the SyncAdmin tool, use the **Startup** options pane to specify that your .tcl file is a startup script. DesignSync runs the startup script when users invoke any DesignSync client.

When you use the DS MW **Cell Check In** dialog to check in the FILL view, the **Check Dependencies** option is selected by default. This option causes the check-in operation to compare the timestamp of the FILL view you are checking in to the timestamp of CEL. The dependency check works as follows:

DesignSync Data Manager MW User's Guide

- If the FILL view is not specified for checkin but the CEL view is, no check is done for FILL.
- If the FILL view is specified for checkin but the CEL view is not, the checkin does not proceed because FILL depends on CEL.
- If both the FILL view and the CEL view are specified for checkin, the checkin does not proceed because FILL depends on CEL and CEL is newer than FILL.

Related Topics

ENOVIA Synchronicity Command Reference: mw dependency Command

Checking In a Cell

Checking In

Checking In a Library

You can use the Library Check In Options form to check in all library data in one operation or you can optionally select specific views you want to check in.

To check in a library:

1. Select **Synchronicity => Library Check In...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information:

The screenshot shows the 'Library Check In Options' dialog box. The title bar reads 'Library Check In Options'. The dialog is organized into several sections:

- Library:** A text input field with a 'Browse...' button to its right.
- Views:** A dropdown menu currently showing 'All', followed by a 'Selected:' text input field with its own 'Browse...' button.
- Leave behind...:** A list of four items, each with a diamond-shaped selection icon:
 - Unlocked copies
 - Locked copies
 - Links to cache
 - Links to mirror
- Checkboxes:** A list of four options, each with an unchecked checkbox:
 - Allow check in of new items
 - Force check in
 - Perform dry run only
 - Export "other" library information prior to checkin
- Exclude (comma separated list):** A text input field.
- Comment:** A large text area for entering notes.
- Buttons:** 'OK', 'Cancel', 'Apply', and 'Help' buttons are located at the bottom right of the dialog.

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library.

Views:

Specify the views you want to check in.

To check in all views in the library, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in your workspace.)

Leave Behind...

Select the state in which you want your workspace objects left after checking in design files:

- **Unlocked copies** - Leaves unlocked objects in your workspace after checkin. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions.
- **Locked copies** - Leaves locked objects in the workspace after checkin. You can continue to make changes; other users cannot check out the files for editing.
- **Links to cache** - Leaves a link to a shared copy of the object in a cache directory.
- **Links to mirror** - Leaves a link to a shared copy of the object in a mirror directory.

Allow check in of new items

Select this option if the library has files that have never been checked in. If you try to check in a library that contains new files without selecting this option, the checkin proceeds, but for each new object, DS MW displays a message that the object is not under revision control and you must use the **-new (Allow check in of new items)** option.

Force check in

Select this option to create a new version in the vault, even if the version you are checking in is identical to the old version.

Note: 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.

Perform dry run only

Select this option to have DesignSync treat the operation as a trial run; no objects are actually checked in. This option helps detect problems that might prevent the checkin from succeeding.

Export "other" library information prior to checkin

Select this option to export Other Library Information to the vault when the library data is checked in.

Exclude:

Specify a comma-separated list of objects to be excluded from the check-in operation. Spaces between the objects are allowed. Wildcards are allowed.

Comment:

Use this field to specify a comment that will be attached to the specified objects when they are checked in.

Note: Your project leader may require that every checkin have a comment of a minimum length.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

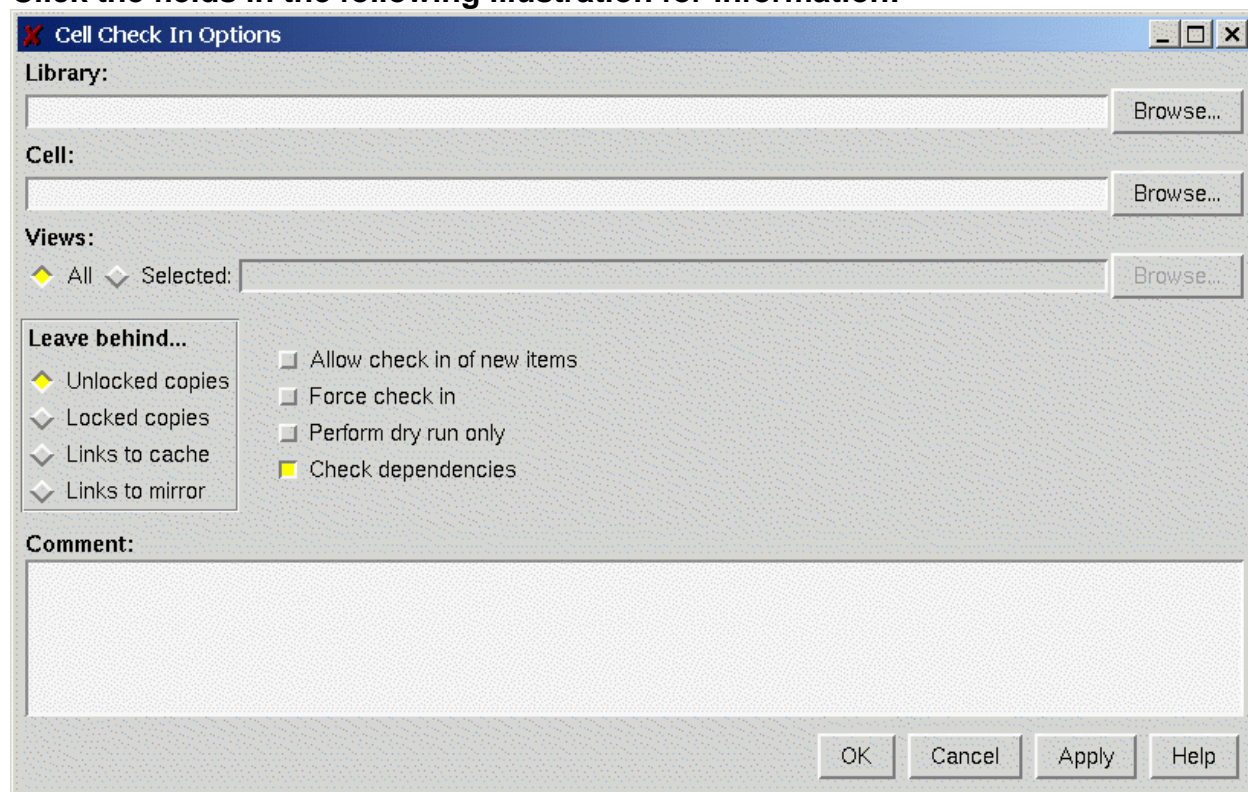
Checking In a Cell

Note: To check in a cell, you must first set up the library. See Setting Up a Library for information.

To check in a cell:

1. Select **Synchronicity => Cell Check In...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information:



Note: The cell check-in operation displays an error if you try to check in a cell for which the technology file (`tech.tf`) has never been checked in to the DesignSync vault. (DS MW checks in the file when you set up a library with the **Library Setup Options** form or export the technology file with the **Export Library Information** form.)

The `tech.tf` file does not need to be in your local workspace as long as it is managed (exists in the vault) and is unretired on the current branch. The managed file can also be modified locally.

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Cell:

Click **Browse...** to select a cell to check in or type a cell name in this field. If you have any cells open, the **Cell** field displays the most recently opened cell.

Views:

Specify the views you want to check in.

To check in all views in the library, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in your workspace.)

Leave Behind...

Select the state in which you want your workspace objects left after checking in design files:

- **Unlocked copies** - Leaves unlocked objects in your workspace after checkin. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions.
- **Locked copies** - Leaves locked objects in the workspace after checkin. You can continue to make changes; other users cannot check out the files for editing.
- **Links to cache** - Leaves a link to a shared copy of the object in a cache directory.
- **Links to mirror** - Leaves a link to a shared copy of the object in a mirror directory.

Allow check in of new items

Select this option if the cell has never been checked in. If you try to check in new objects without selecting this option, the checkin proceeds, but for each new object, DS MW displays a message that the object is not under revision control and you must use the **-new (Allow check in of new items)** option.

Force check in

Select this option to create a new version in the vault, even if the version you are checking in is identical to the old version.

Note: 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.

Perform dry run only

Select this option to have DesignSync treat the operation as a trial run; no objects are actually checked in. This option helps detect problems that might prevent the checkin from succeeding.

Check dependencies

This option (selected by default) causes the check-in operation to compare the timestamps of views to those of other views they depend on. If a view depends on other views and if any of those other views has a more recent timestamp than the view that depends on them, DS MW displays a message and the checkin does not proceed.

Note: To use this option, you must first specify the dependencies between views. See [Specifying Dependencies Between Views](#) for information.

Comment

Use this field to specify a comment that will be attached to the specified objects when they are checked in.

Note: Your project leader may require that every checkin have a comment of a minimum length.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Checking Out

Checking Out a Library

To check out an entire library or only specified views from a library:

1. Select **Synchronicity => Library Check Out...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information.

The screenshot shows a dialog box titled "Library Check Out Options". It has a standard Windows-style title bar with a close button. The dialog is organized into several sections:

- Library:** A text input field followed by a "Browse..." button.
- Views:** A dropdown menu currently showing "All", followed by another text input field and a "Browse..." button.
- View Selection:** A list of view types, each with a diamond-shaped selection icon:
 - Unlocked copies
 - Locked copies
 - Links to cache
 - Links to mirror
- Options:** Two checkboxes:
 - Force overwrite of local modifications
 - Unify workspace state
- Version:** A text input field followed by a "Browse..." button.
- Exclude (comma separated list):** A text input field.
- Buttons:** A row of four buttons at the bottom: "OK", "Cancel", "Apply", and "Help".

Note:

When it checks out a Milkyway collection object, the DS MW check-out 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 check-out operation fetches the object from the vault (with the local version number it had at the time of checkin).

By default, the DS MW check-out operation fails if you have a modified local version in your workspace (other than the current, or highest numbered, local version) and if that version has a local version number equal to or higher than the local version being fetched.

A DesignSync administrator can change this default setting. For information, see [Controlling the Check-Out Operation's Handling of Local Versions](#).

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Views:

Specify the views you want to check out.

To check out all views in the library, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the vault.)

Unlocked copies

Select this option to have unlocked objects in your workspace after the checkout. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions or Locked references. (See DesignSync Data Manager User's Guide: Defining a Default Fetch State for more information.)

Locked copies

Select this option to have locked objects in your workspace after the checkout. Others cannot check out the files for editing.

Links to cache

Select this option to have in your workspace links to shared copies in a cache directory.

Links to mirror

Select this option to have in your workspace links to shared copies in a mirror directory.

Force overwrite of local modifications

Select this option if you want to overwrite locally modified files in your workspace.

Note:

- You may need to select the **Force overwrite of local modifications** option in order to allow the object being checked out to overwrite a locally modified copy of the object. (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.) For example, if you check out a Milkyway collection object, create a new local version of the object in your workspace, then try to check out the latest version of the object from the vault, the checkout operation fails because DesignSync considers the object to be locally modified. To successfully check out a locally modified object, you must select **Force overwrite of local modifications**.
- Selecting this option also imports Other Library Information from the vault. If you have changed Other Library Information in your workspace (for example, if you have expressed logical equivalences of cells or modified properties), then that information is replaced by the Other Library Information from the vault. The import of the Other Library Information occurs only if there is a newer version in the vault that is fetched as part of the checkout.

Unify workspace state

Select this option to set 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).

Version

Specify the version to check out, for example, 1.1. Or click **Browse** to display a list of version selectors.

If you click **Browse**, DS MW displays a list of selector names. Click a name and click **Select**. DS MW displays the selector in the **Version** field. The version selector list can include:

- Tags specified by a Project Leader through SyncAdmin.
- Configuration names and selectors.
- Standard selectors such as Trunk and Latest, and special specifiers such as Date(), VaultDate() and auto().

Tip: You can use this option to check out a cell by its local version number. For example, suppose there are two local versions checked in for a cell: 1.1 (local version 2) and 1.2, (local version 5). To check out local version 2, you can type 1.1 or MW_2 (the tag for local version 2).

Exclude:

Specify a comma-separated list of objects to be excluded from the check-out operation. Spaces between the objects are allowed. Wildcards are allowed.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

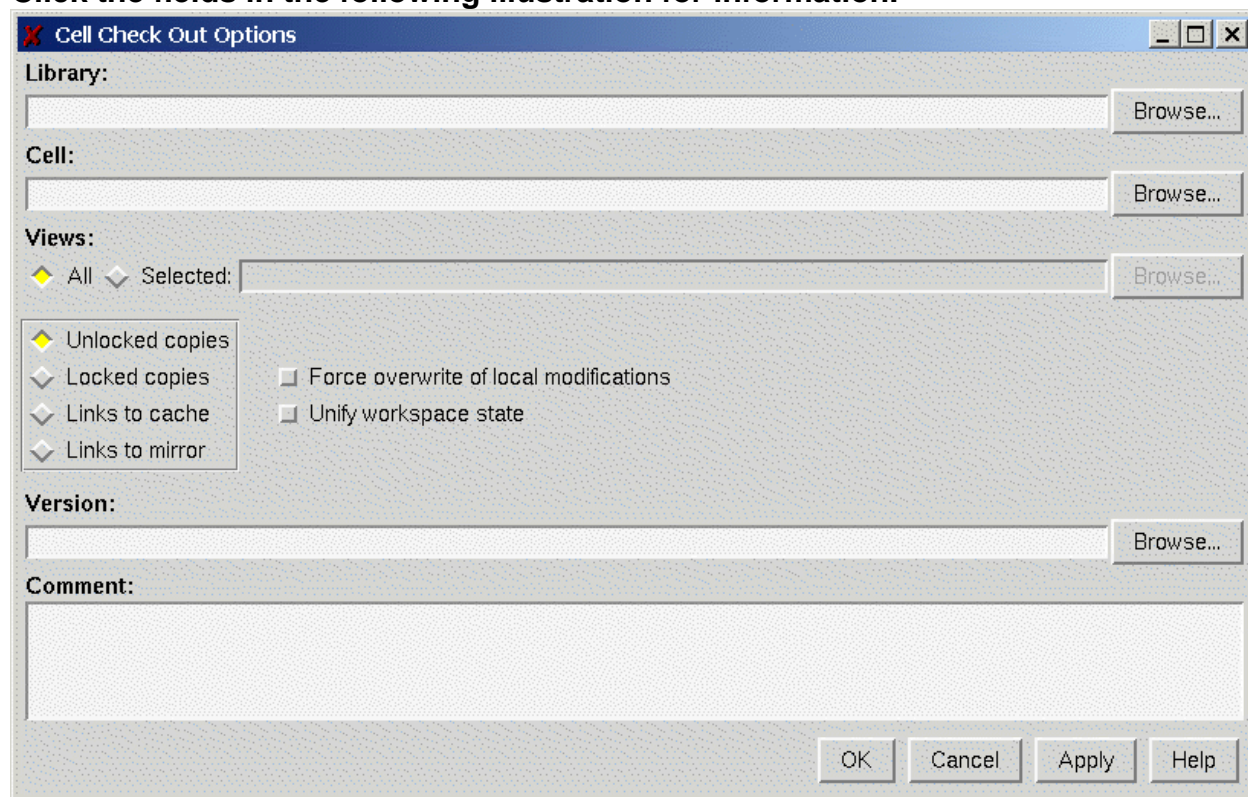
Click to display the DS MW online help for this form.

Checking Out a Cell

To check out a cell:

1. Select **Synchronicity => Cell Check Out...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information.



Note:

When it checks out a Milkyway collection object, the DS MW check-out 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 check-out operation fetches the object from the vault (with the local version number it had at the time of checkin).

By default, the DS MW check-out operation fails if you have a modified local version in your workspace (other than the current, or highest numbered, local version) and if that version has a local version number equal to or higher than the local version being fetched.

A DesignSync administrator can change this default setting. For information, see [Controlling the Check-Out Operation's Handling of Local Versions](#).

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Cell:

Click **Browse...** to select a cell or type a cell name in this field. (**Note:** The list of available cells displayed is based on the cells that exist in the vault.)

If you have any cells open, the **Cells** field displays the most recently opened cell.

Views:

Specify the views you want to check out.

To check out all views in the library, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the vault.)

Unlocked copies

Select this option to have unlocked objects in your workspace after the checkout. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions or Locked references. (See DesignSync Data Manager User's Guide: Defining a Default Fetch State for more information.)

Locked copies

Select this option to have locked objects in your workspace after the checkout. You reserve the right to create the next version of the object when you check in your design files. Others cannot check out the files for editing.

Links to cache

Select this option to have in your workspace links to shared copies in a cache directory.

Links to mirror

Select this option to have in your workspace links to shared copies in a mirror directory.

Force overwrite of local modifications

Select this option if you want to overwrite locally modified files in your workspace.

Note: You may need to select the **Force overwrite of local modifications** option in order to allow the object being checked out to overwrite a locally modified copy of the object. (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.) For example, if you check out a Milkyway collection object, create a new local version of the object in your workspace, then try to check out the latest version of the object from the vault, the checkout operation fails because DesignSync considers the object to be locally modified. To successfully check out a locally modified object, you must select **Force overwrite of local modifications**.

Unify workspace state

Select this option to set 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).

Version

Specify a version of a cell to check out, for example, 1.1. Or click **Browse** to display a list of selectors.

If you click **Browse**, DS MW displays a list of version selector names. Click a name and click **Select**. DS MW displays the selector in the **Version** field. The version selector list can include:

- Tags specified by a Project Leader through SyncAdmin.
- Configuration names and selectors.
- Standard selectors such as Trunk and Latest, and special specifiers such as Date(), VaultDate() and auto().

Tip: You can use this option to check out a cell by its local version number. For example, suppose there are two local versions checked in for a cell:1.1 (local version 2) and 1.2, (local version 5). To check out local version 2, you can type1.1 or MW_2 (the tag for local version 2).

Comment:

Use this field to specify a comment that will be attached to the specified objects when they are checked out.

Note: Your project leader may require that every checkout have a comment of a minimum length.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Canceling a Checkout

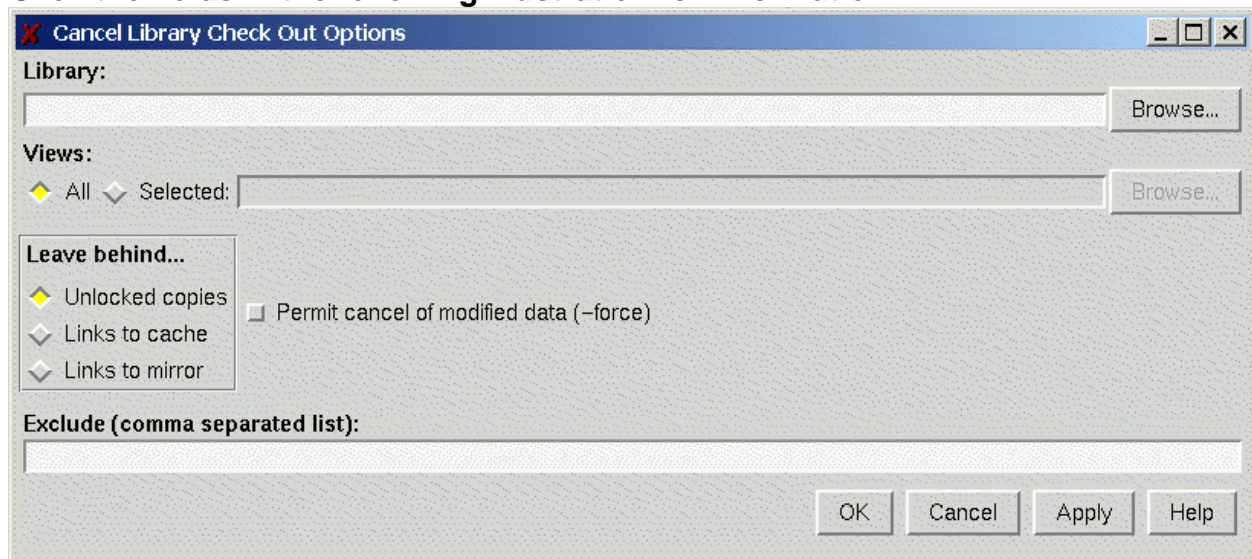
Canceling a Library Checkout

Canceling a checkout releases the locks that you have on objects, without checking in any changes you have made, and re-fetches the objects into your workspace.

To cancel the checkout of all objects in a library or of specified cell views of a library:

1. Select **Synchronicity => Library Cancel Check Out...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information.



Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Views:

Specify the views for which you want to cancel the checkout.

To cancel the checkout of all views in the library, click **All** (the default selection).

To select specific views for which you want to cancel the checkout, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the workspace.)

Leave Behind...

Select the state in which you want your workspace objects left after cancelling the checkin:

- **Unlocked copies** - Leaves unlocked objects in your workspace. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions. (See *DesignSync Data Manager Administrator's Guide: Default Fetch State* for more information.)
- **Links to cache** - Leaves a link to a shared copy of the object in a cache directory.
- **Links to mirror** - Leaves a link to a shared copy of the object in a mirror directory.

Permit cancel of modified data (-force)

Select this option to have the cancel operation overwrite the objects in your workspace when it re-fetches them from the vault, even if you have modified those objects in your workspace.

Exclude:

Specify a comma-separated list of objects to be excluded from the cancel operation. Spaces between the objects are allowed. Wildcards are allowed.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Canceling a Cell Checkout

Canceling a checkout releases the locks that you have on design files, without checking in any changes you have made, and re-fetches the files into your workspace.

To cancel the checkout of a cell:

1. Select **Synchronicity => Cell Cancel Check Out...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information.

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Cell:

Click **Browse...** to select a cell or type a cell name in this field. (**Note:** The list of available cells displayed is based on the view directories that exist in the workspace.)

If you have any cells open, the **Cells** field displays the most recently opened cell.

Views:

Specify the views for which you want to cancel the checkout.

To cancel the checkout of all views of the specified cell, click **All** (the default selection).

To select specific views for which you want to cancel the checkout, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the workspace.)

Leave Behind...

Select the state in which you want your workspace objects left after cancelling the checkin:

- **Unlocked copies** - Leaves unlocked objects in your workspace. This is the default unless your project leader has defined a default fetch state. This is also the default if the default fetch state is References to versions. (See DesignSync Data Manager User's Guide: Defining a Default Fetch State for more information.)
- **Links to cache** - Leaves a link to a shared copy of the object in a cache directory.
- **Links to mirror** - Leaves a link to a shared copy of the object in a mirror directory.

Permit cancel of modified data (-force)

Select this option to have the cancel operation overwrite the objects in your workspace when it re-fetches them from the vault, even if you have modified those objects in your workspace.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Populating Your Workspace

To fetch Milkyway objects to your workspace, use the DesignSync populate operation. This operation fills, or populates, a local folder or entire hierarchy with objects from the corresponding revision-control vault.

Note: Before populating Milkyway objects to your workspace for the first time, you must use the **Library Setup Options** form to set up your library. For information, see [Setting Up a Library](#).

The populate operation fetches Milkyway objects:

- When it fetches Milkyway collection 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).
- You can use the **Local Versions (-savelocal)** option to specify 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.) For information, see [DesignSync Data Manager User's Guide: Populating Your Work Area](#) or the [ENOVIA Synchronicity Command Reference: populate](#).
- In addition to fetching objects, the populate operation also loads information from the `tech/tech.tf` and `refs.txt` files into the library file.
- During an initial populate of a workspace, or if you use **populate -force** and that populate fetches a new version of the Other Library Information, the populate operation imports that new Other Library Information into the newly created library.

You can perform the populate operation in either of two ways:

- From DesignSync List View, select the work area directory and then select **Revision Control => Populate**. For more information see [DesignSync Data Manager User's Guide: Populating Your Work Area](#).
- Use the **populate** command. For more information, see [ENOVIA Synchronicity Command Reference: populate](#).

Note: If a library is in reference control file mode and you populate the library into a new workspace with a different name, DS MW cannot determine the name of the library and therefore cannot update the library reference information in the attached file. You must edit the attached file and correct the library references manually.

Related Topics

DesignSync Data Manager MW User's Guide

Setting Up a Library

DesignSync Data Manager User's Guide: Populating Your Work Area

ENOVIA Synchronicity Command Reference: **populate**

Displaying a Cell's Data Sheet

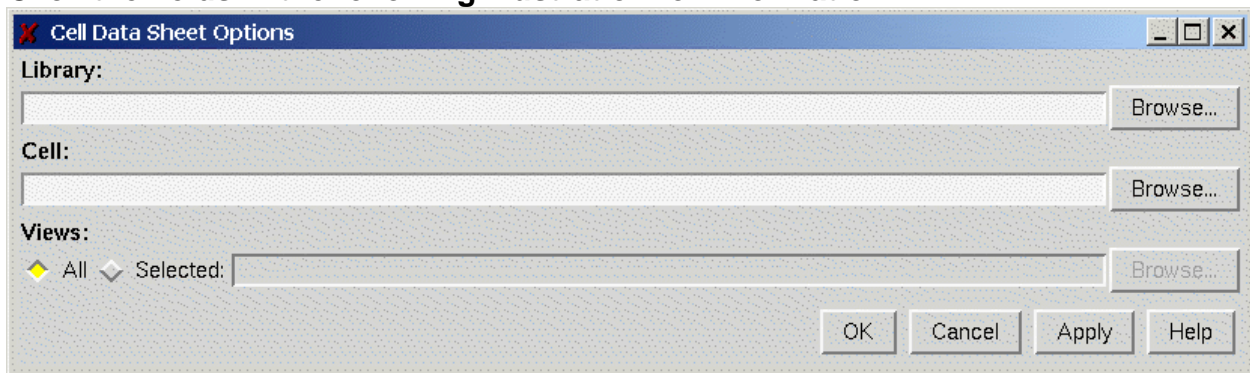
You can display information about the views of a cell by displaying their datasheets. The data sheet shows the vault associated with the cell, attached notes, and revision control information.

Note: The cell must be checked in from or checked out to your workspace in order for you display its datasheet.

To display a cell's data sheet:

1. Select **Synchronicity => Cell Data Sheet...** from your Milkyway-based tool.
2. Modify the fields of the dialog as needed.
3. Click **OK**.

Click the fields in the following illustration for information.



Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Cell:

Click **Browse...** to select a cell or type a cell name in this field. (**Note:** The list of available cells displayed is based on the cells that exist in the vault.)

If you have any cells open, the **Cells** field displays the most recently opened cell.

Views:

Specify the views for which you want to display data sheets.

To display the data sheets of all views of the specified cell, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the vault.)

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Displaying a Cell's Version History

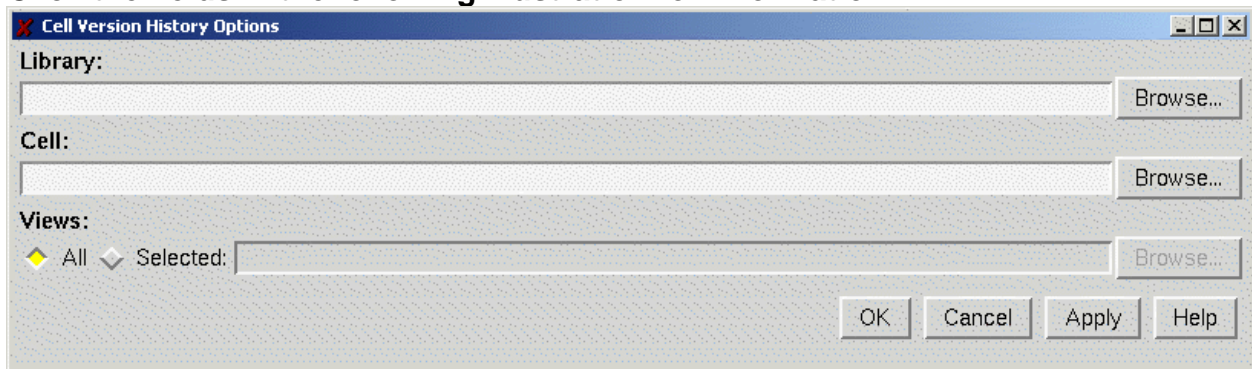
You can display status information and version history for a cell. The display also shows the local version for each version of an object in the DesignSync vault. (In the version history output, local versions have the version tag `MW_<n>`, where `<n>` is a local version number.)

Note: The cell must be checked in from or checked out to your workspace in order for you display its version history.

To display the version history of a cell:

1. Select **Synchronicity => Cell Version History...** from your Milkyway-based tool.
2. Modify the fields of the dialog as needed.
3. Click **OK**.

Click the fields in the following illustration for information.



Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Cell:

Click **Browse...** to select a cell or type a cell name in this field. (**Note:** The list of available cells displayed is based on the cells that exist in the vault.)

If you have any cells open, the **Cells** field displays the most recently opened cell.

Views:

Specify the views for which you want to see a version history.

To display the history of all views of the specified cell, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in the vault.)

Note: You can also use the DesignSync **vhistory** command to display a cell's version history, including the local version for all versions of an object in the DesignSync vault. For information, see **vhistory** in the ENOVIA Synchronicity Command Reference.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Tagging Libraries and Cells

Tagging Libraries

Tagging is the application of a symbolic name, called a tag, to a version or a branch in DesignSync. Tags can be applied only to objects 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.

If you want to tag an unmanaged object or a locally modified object (whether locked or not) in your work area, you must first check in your changes to create a new version of the object in the vault. Then you can tag that version. If you do not check in a locally modified object before you use the tag command, the tag operation fails for that object and does not tag any version of it in the vault. This is the default behavior of the tag operation.

If you want to tag the version in the vault from which a locally modified object was derived (instead of tagging the new version that contains your changes), you can select the Tag modified objects check box.

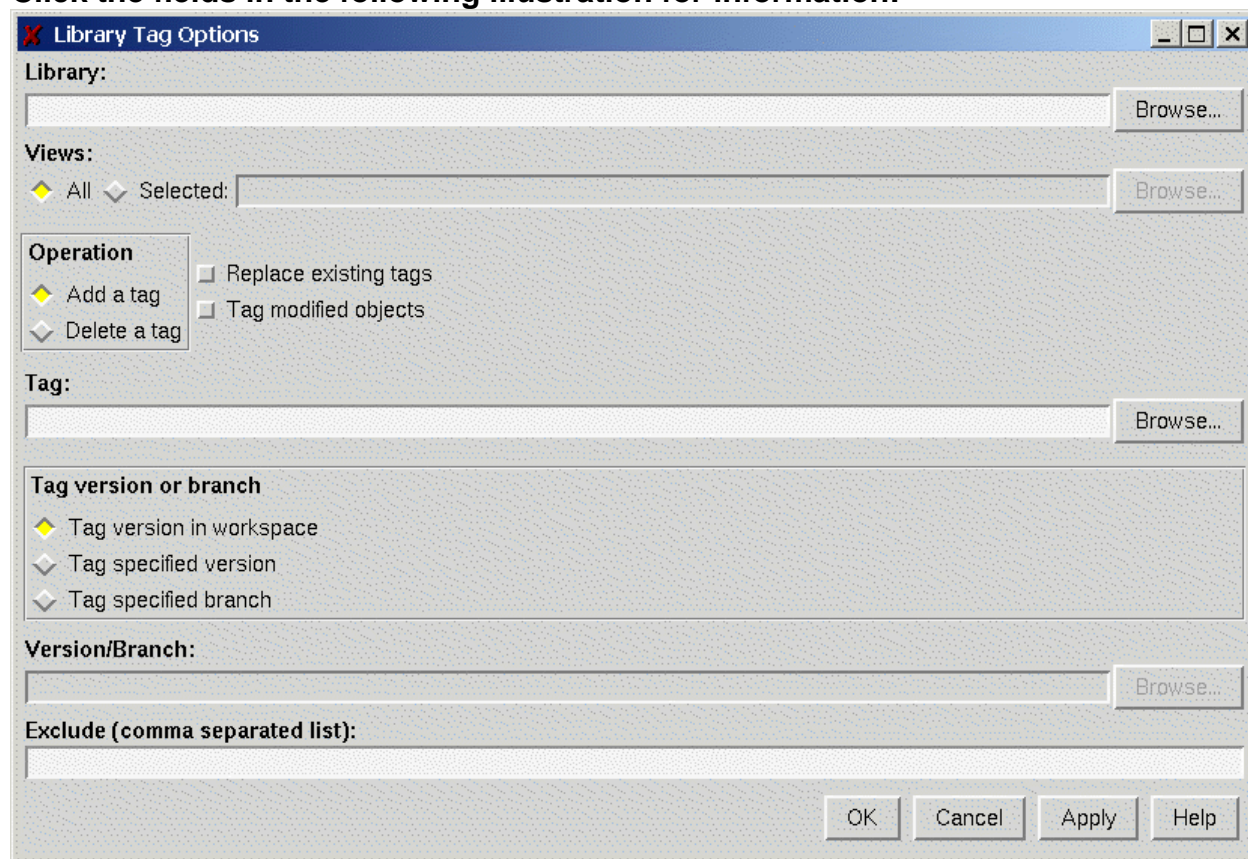
You can use the **Tag Library Options** form to tag all library data in one operation or you can optionally select specific views you want to tag.

To tag (or delete a tag from) a library:

1. Select **Synchronicity => Library Tag...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Note: The **Tag specified version** and **Tag specified branch** options work in conjunction with the **Version/Branch** field. If you select either option, you need to type a version number or name or a branch name in the **Version/Branch** field. To display a list of versions and branches to choose from, click **Browse**.

Click the fields in the following illustration for information:



Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library.

Views:

Specify the views you want to tag.

To tag all views in the library, click **All** (the default selection).

To select specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in your workspace.)

Add a tag

Select this option to add a tag.

Delete a tag

Select this option to delete a tag that is already assigned. Because a tag can exist on only one version of a file at a time, DS MW ignores the Tag version or branch field when you delete a tag.

Replace existing tags

Moves 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 move the tag, you must select the **Replace existing tags** option. The **Delete a tag** and **Replace existing tags** options are mutually exclusive.

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. You can move a tag from a branch to a version or a version to a branch; however, DesignSync provides a warning message when you do so.

Tag modified objects

For modified objects in your work area, this option tags the version in the vault that you fetched to your workspace. You might use this option, for example, if you have modified objects in your workspace 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 workspace object is unmodified, the tag operation tags the version in the vault that matches the one in your workspace.

Tag:

Type the tag in this field. It is a good idea to use tags that are 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 whitespace, 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 the Hierarchical Configuration Manager software.)
- 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 Naming Conventions

Branch tags and version tags share the same name space. To distinguish version selectors from branch selectors, you append `:<versiontag>` to the branch name; for example, `Gold:Latest` is a valid branch selector. You can leave off the Latest keyword as shorthand; for example, `Gold:` is equivalent to `Gold:Latest`. The selector `Trunk` is also a valid branch selector. `Trunk` is a shorthand selector for `Trunk:Latest`.

You cannot assign the same tag name to both a version and a branch of the same object. For example, a file called `top.v` cannot have both a version tagged `Gold` and a branch tagged `Gold`. However, `top.v` can have a version tagged `Gold` while another file, `alu.v`, can have a branch tagged `Gold`.

Consider adopting a consistent naming convention for branch and version tags to reduce confusion. For example, you might have a policy that branch tags always begin with an initial uppercase letter (`Rel2.1`, for example) whereas version tags do not (`gold`, for example).

Tag version or branch

Select the version of the object that you want to tag.

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

Note: If you choose **Tag specified version** or **Tag specified branch**, you can display a list of versions and branches for the object you selected to tag. Click in the **Version/Branch** field and then click **Browse** to display the pull-down menu.

Version/Branch

Specify the version selector or branch tag or click **Browse** to select a tag name. This field works in conjunction with the **Tag version or branch** options:

- If you select the **Tag specified version** option, you must specify a version selector or selector list. For information on selectors, see *What are Selectors?* in *DesignSync Data Manager User's Guide*.
- If you select 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.

If you click **Browse**, DS MW displays a list of tag names. Click a name and click **Select**. DS MW displays the selected tag in the **Tag** field. The tag list can include:

- Tags specified by a Project Leader through SyncAdmin.
- Configuration names and selectors.
- Standard selectors such as Trunk and Latest, and special specifiers such as Date(), VaultDate() and auto().

Exclude

Specifies the kinds of files or directories you want to exclude from the tag operation. Wildcards are allowed; for example, to exclude all log files from the tag operation, you would specify *.log in the Exclude field. Separate items with commas.

Do not specify paths in your arguments to **Exclude**. Before operating on each object, DesignSync compares the object's leaf name (path stripped off) to the items in **Exclude** to see if there is a match. Because it does not consider the object's path, DesignSync does not match any item in the exclude list specified with a path.

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Tagging Cells

Tagging is the application of a symbolic name, called a tag, to a version or a branch in DesignSync. Tags can be applied only to objects 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.

If you want to tag an unmanaged object or a locally modified object (whether locked or not) in your work area, you must first check in your changes to create a new version of the object in the vault. Then you can tag that version. If you do not check in a locally modified object before you use the tag command, the tag operation fails for that object and does not tag any version of it in the vault. This is the default behavior of the tag operation.

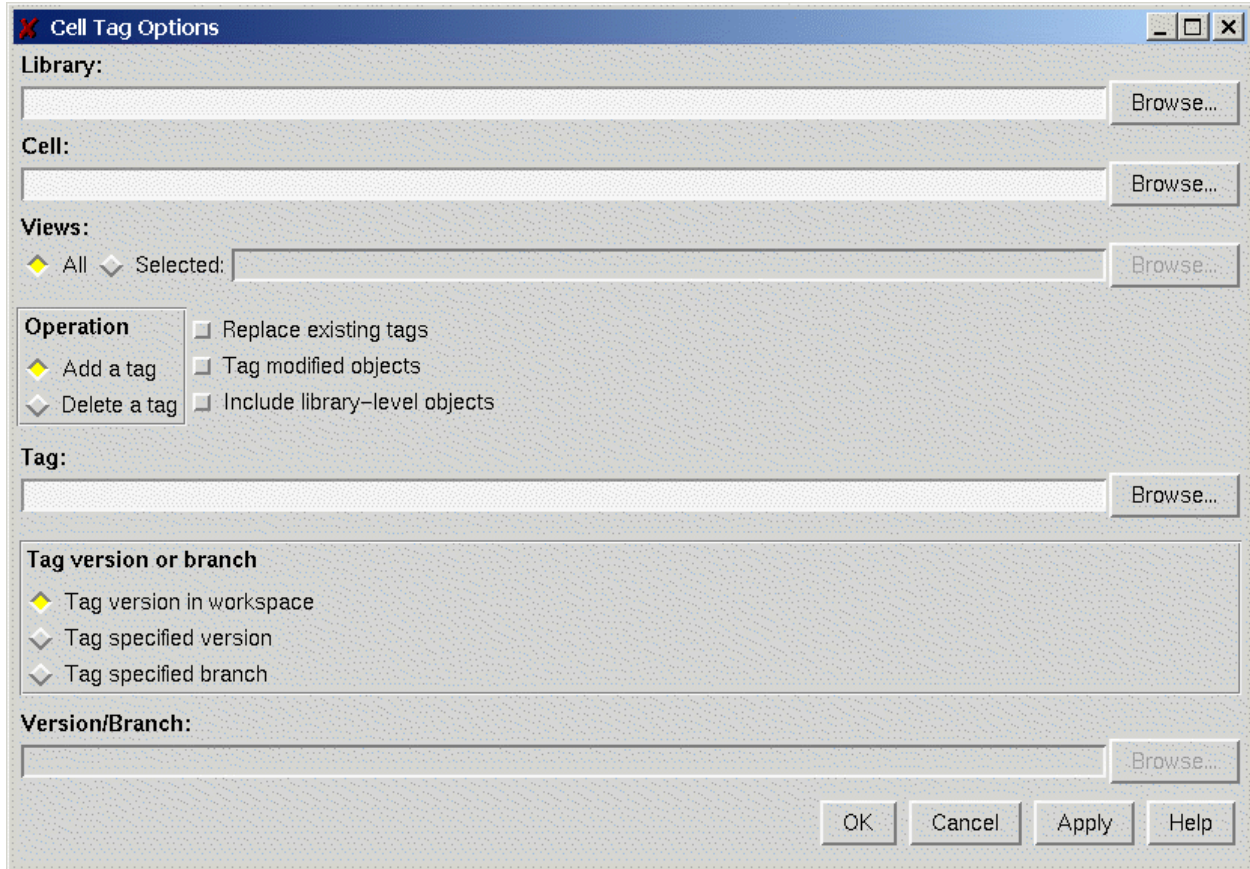
If you want to tag the version in the vault from which a locally modified object was derived (instead of tagging the new version that contains your changes), you can select the Tag modified objects check box.

You can use the **Cell Tag Options** form to tag cells. You can tag all views for a cell in one operation or you can optionally select specific views you want to tag.

To tag (or delete a tag from) a cell:

1. Select **Synchronicity => Cell Tag...** from your Milkyway-based tool.
2. Modify the fields of the form as needed.
3. Click **OK**.

Click the fields in the following illustration for information.



Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library.

Cell:

Click **Browse...** to select a cell to tag or type a cell name in this field. If you have any cells open, the **Cell** field displays the most recently opened cell.

Views:

Specify the views you want to tag.

To tag all views in the cell, click **All** (the default selection).

To tag specific views, click **Select** and then click **Browse...** to select one or more views. (**Note:** The list of available views displayed is based on the view directories that exist in your workspace.)

Add a tag

Select this option to add a tag.

Delete a tag

Select this option to delete a tag that is already assigned. Because a tag can exist on only one version of an object at a time, DS MW ignores the **Tag version or branch** field when you delete a tag.

Replace existing tags

Moves 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 move the tag, you must select the Replace existing tags option. The **Delete a tag** and **Replace existing tags** options are mutually exclusive.

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. You can move a tag from a branch to a version or a version to a branch; however, DesignSync provides a warning message when you do so.

Tag modified objects

For modified objects in your work area, this option tags the version in the vault that you fetched to your workspace. You might use this option, for example, if you have modified objects in your workspace 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 workspace object is unmodified, the tag operation tags the version in the vault that matches the one in your workspace.

Include library-level objects

This option tags the library objects (`tech.tf`, `refs.txt`, and `shadow.sync.mw.lib`).

Tag:

Type the tag in this field. It's a good idea to use tags that are easily understood, for example: `Rel2.1`, `ready_for_simulation`, `current_demo`, `Golden`. Tag names:

Tag names:

- Can contain letters, numbers, underscores (`_`), periods (`.`), hyphens (`-`), and forward slashes (`/`). All other characters, including whitespace, 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 the Hierarchical Configuration Manager software.)
- 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 Naming Conventions

Branch tags and version tags share the same name space. To distinguish version selectors from branch selectors, you append `:<versiontag>` to the branch name; for example, `Gold:Latest` is a valid branch selector. You can leave off the `Latest` keyword as shorthand; for example, `Gold:` is equivalent to `Gold:Latest`. The selector `Trunk` is also a valid branch selector. `Trunk` is a shorthand selector for `Trunk:Latest`.

You cannot assign the same tag name to both a version and a branch of the same object. For example, a file called `top.v` cannot have both a version tagged `Gold` and a branch tagged `Gold`. However, `top.v` can have a version tagged `Gold` while another file, `alu.v`, can have a branch tagged `Gold`.

Consider adopting a consistent naming convention for branch and version tags to reduce confusion. For example, you might have a policy that branch tags always begin with an initial uppercase letter (`Rel2.1`, for example) whereas version tags do not (`gold`, for example).

Tag version or branch

Select the version of the object that you want to tag.

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

Note: If you choose **Tag specified version** or **Tag specified branch**, you can display a list of versions and branches for the object you selected to tag. Click in the **Version/Branch** field and then click **Browse** to display the pull-down menu.

Version/Branch

Specify the version selector or branch tag or click **Browse** to select a tag name. This field works in conjunction with the **Tag version or branch** options:

- If you select the **Tag specified version** option, you must specify a version selector or selector list. For information on selectors, see *What are Selectors?* in *DesignSync Data Manager User's Guide*.
- If you select 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.

If you click **Browse**, DS MW displays a list of tag names. Click and name and click **Select**. DS MW displays the selected tag in the **Tag** field. The tag list can include:

- Tags specified by a Project Leader through SyncAdmin.
- Configuration names and selectors.
- Standard selectors such as Trunk and Latest, and special specifiers such as Date(), VaultDate() and auto().

OK

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Exporting and Importing Library Information

Exporting Library Information

The Export Library Information form lets you export these items of library information to the DesignSync vault:

- A library's references
- The contents of the technology file
- Other Library Information

To export one or all items of library information:

1. Select **Synchronicity => Export library information...** from your Milkyway-based tool.
2. Select the library for which you want to export information.
3. Select the items of library information you want to export. (By default, all items are selected.)
4. Click **OK**.

Click the fields in the following illustration for information.

Export Library Information

Library:

Export "other" library information (lib file and attached files)

Export technology file

Export library references

Reference libraries (you may change them to use relative paths or \$ENVVAR syntax):

Comment:

Field Descriptions

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Press **Return** or move the cursor outside the form to display the current references for the library.

Export "other" library information (lib file and attached files)

Select this option to export Other Library Information (for example, logical equivalences of cells). This option is selected by default. If you have a library open, the export operation prompts you to close the library.

Export technology file

Select this option to export the contents of the `tech` technology file to the DesignSync vault. This file is an ASCII file (`tech/tech.tf`) created and modified by DS MW with the purpose of making DesignSync aware of the library technology data for a design. This option is selected by default.

Notes:

- If you have a library open, you can export the technology file only for that open library.
- To export the technology file of a different library, click **Browse** to select the library. If you have a library open, the export operation prompts you to close the library.
- This option is not available if your DS MW administrator has disabled the **Manage technology file separately option** in SyncAdmin. For information, see SyncAdmin Help: Third Party Integration Options

Export library references

Select this option to export a library's references. An export of a library's references is a DS MW operation that extracts information in the library file about reference libraries, puts the information in a file called `refs.txt`, and then checks that file into the DesignSync vault. This option is selected by default.

Reference libraries

This field lets you change the prefix of the reference libraries. For example, suppose it is a practice in your company that at all sites the reflibs library is located next to individual users' libraries. Because the location is the same from site to site, you can change your reference library path from:

```
REFERENCE /disk1/libraries/reflibs/cmos09
```

```
to REFERENCE ../reflibs/cmos09
```

You can also use an environment variable in the path to reference libraries. For example, your project leader might define the `MW_REFLIB` environment variable as a path where all reference libraries for a project reside. Then you can use the variable to specify the path to your reference library as:

```
REFERENCE $MW_REFLIB/reflibs/cmos09
```

Notes:

- You can edit the Reference libraries field only if you first select **Export library references**.
- If a library already exists in your workspace and is in reference control file mode, the Reference Libraries field cannot be edited.

Comment:

Use this field to specify a comment that will be attached to the specified objects when they are checked in.

Note: Your project leader may require that every checkin have a comment of a minimum length.

OK

Click to close the form and perform the operations you selected.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Importing Library Information

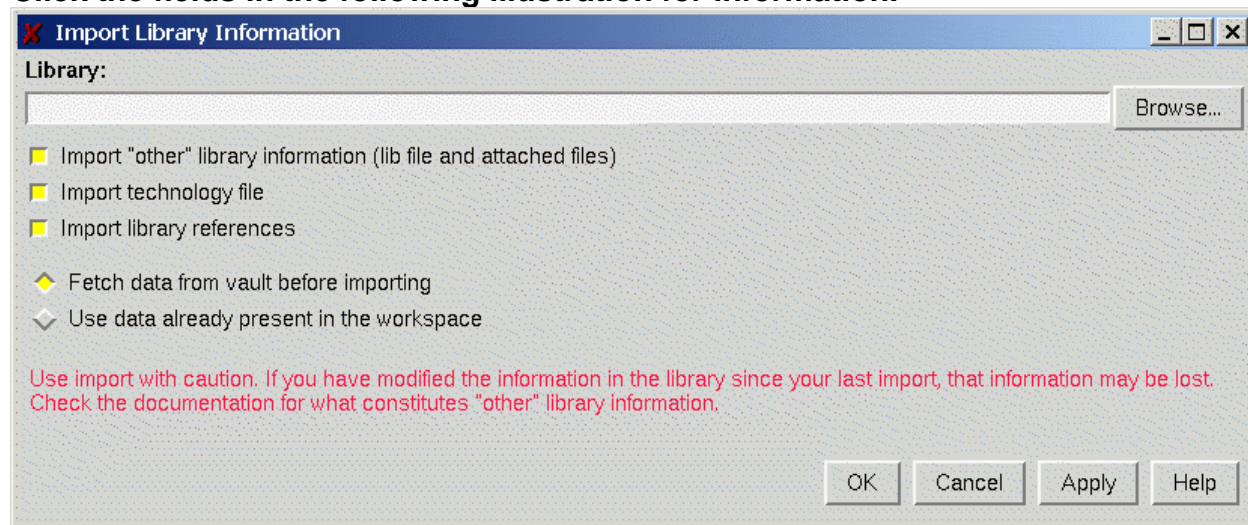
The **Import Library Information** form lets you bring the following library information into the library file in your workspace:

- A library's references
- The contents of the technology file
- Other Library Information

To import one or all items of library information:

1. Select **Synchronicity => Import Library information...** from your Milkyway-based tool.
2. Select the library into which you want to import information.
3. Select the items of library information you want to import. (By default, all items are selected.)
4. Click **OK**.

Click the fields in the following illustration for information.



Field Description

Library:

Click **Browse...** to select a library or type the path to the library in this field. (You can type the full path to the library or a path relative to the directory from which you invoked your Milkyway tool.)

If you have a library open, the **Library** field displays the path to that library in your workspace.

Import "other" library information (lib file and attached files)

Select this option to import Other Library Information (for example, logical equivalences of cells). The import operation integrates this information into the library file. (This option is selected by default.)

Note: If a library is in reference control file mode and you import the library from the vault to a different library name in your workspace, DS MW cannot determine the name of the library and therefore cannot update the library reference information in the attached file. You must edit the attached file and correct the library references manually.

Import technology file

This option imports the contents of the technology file (`tech/tech.tf`) from the vault into your library file. (This option is selected by default.)

This option is not available if your DS MW administrator has disabled the **Manage technology file separately option** in SyncAdmin. For information, see SyncAdmin Help: Third Party Integration Options

Import library references

The import of this information imports the `refs.txt` file from the vault into your library file. This option is selected by default.

Fetch data from vault before importing

Select this option to fetch the data you selected for import to your workspace and then import it into the library. This option is selected by default.)

Use data already present in the workspace

Select this option to have the data (library references, technology file, and Other library information) that is already available in your workspace imported into the library.

Caution: Use the Import operation with care. If you have modified the library references, the technology file, or the other library information since you last imported library information, the information may be lost. For example, if you created new logical equivalences since the last time you imported Other Library Information, you will lose those equivalences.

See Boris Becomes Controller of the Other Library Information for a scenario of how you can avoid unintentional removal of Other Library Information from your workspace.

OK

DesignSync Data Manager MW User's Guide

Click to close the form and perform the operation.

Cancel

Click to close the form without performing the operation.

Apply

Click to perform the operation without closing the form.

Help

Click to display the DS MW online help for this form.

Purging Versions from the Vault

You can clean up the vault by deleting old versions of objects using the **purge** command. This command deletes specified versions of an object on a single branch in the vault.

You can use the purge command on files, Milkyway objects, Cadence collection objects, or folders. The command also provides options that let you delete all versions of an object except the last <n> number of versions or deletes versions older than <n> number of days.

For more information on the **purge** command, see **purge** in the ENOVIA Synchronicity Command Reference.

Note: To use the purge command, users must have access control rights to delete versions (using the Delete action). For an example of such an access control, see the ENOVIA Synchronicity Access Control Guide: Sample Server Access Controls.

DS MW and DesignSync

DS MW Command Line Interface

The SyncServer architecture generalizes the way DesignSync handles all collection objects and exposes that capability to customers and third party users, so they can integrate their own special data handlers. DesignSync uses this architecture to process Milkyway collection objects.

DesignSync commands and the DesignSync GUI can be used to manage Milkyway collection objects. For example, to check in a Milkyway collection object you can use the **ci** command or the DesignSync **Check In** dialog, as well as the **Cell Check In...** selection of the Synchronicity menu from the Milkyway tool.

In addition, DS MW provides DS MW-specific commands that allow access to DS MW functionality: **mw dependency**, **mw doc**, **mw rebuildlib**, **mw version**, **mw export**, **mw import**, and **mw setuplib**.

Note: DS MW does not use the command line defaults system. The command line defaults system only pertains to DesignSync command line shells.

How DesignSync Handles Milkyway Data

Handling of Milkyway Data in General

DS MW has added a new object type to DesignSync: the Milkyway cell view collection. This object represents as a single entity the latest local version of the cell in the workspace and all its attached files. DesignSync treats Milkyway objects in the following manner:

- DesignSync displays this object as `<cell_name>.sync.mw` in the **List View**. See DesignSync Display of Milkyway Objects for an example.
- With DS MW, Milkyway data (as represented by an object with a colon ":") is checked out into the workspace.
- When an object is first checked in, DesignSync uses the Vault Type registry setting to select a vault type for the object. By default, DesignSync selects Smart Vault as the vault type for objects with a colon (:). For example, `route:1` will be assigned to the Smart Vault vault type.

The `*:*` is a very general pattern. Objects can easily match other vault patterns in the Vault Type registry. Currently, all of the supported vault patterns are, by default, assigned to the lowest priority (9). Therefore, the DS MW installation sets the `*:*` pattern to a priority of (9).

For information, see About Vault Types in the DesignSync System Administration Help.

- When a Milkyway object is checked in, a tag is applied which indicates its local version number. If the local version is 6, the tag MW_6 will be applied. You can use this tag to determine the DesignSync vault version that corresponds to a particular local version.
- **tag** - DS MW treats the MW_* tag as a reserved tag.
- DS MW does not support the shared workspace model (a setup of a team's UNIX environment such that team members can for share a single workspace).
- Because Synopsys commands do not clean up DesignSync local metadata, using Synopsys commands that delete or rename cells causes DesignSync to display those cells as references in **List View** and in output from the **ls** command.

Mirror Directories and Milkyway Data

Users of a DesignSync mirror directory should regularly populate their workspaces in order to update the workspace with files that other users added to or removed from the design configuration. Regularly populating a workspace is especially important for users using a mirror directory for DS MW cell view collection objects because a version of a Milkyway collection object can have different sets of member files. This situation occurs when individual users add or remove attached files of the collection object or when users use a different local version number for the object in their workspaces. For example, if Alice has a Milkyway collection object checked out in mirror mode and Ben checks in a new version of the object with different member files, Ben's checkin is equivalent to adding or removing files from the mirror. Links from Alice's workspace are broken or incomplete. To avoid this problem, Alice should populate her workspace on a regular basis.

For users working with DS MW collection objects, using a cache may be preferable to using a mirror. Using a cache provides the same gains in disk space as a mirror, with the added benefit that links in a user's workspace are never broken. The disadvantage of a cache is that users must populate their workspaces regularly to pick up changes. However, the populate operation can be performed with a script that performs a periodic **populate -incremental -share** of the workspace, so that changes are picked up automatically. How often the script should be run depends on the individual use model, rate of change of the data, and performance of the system.

Note: DS MW does not support the use of a mirror directory as a reference library.

DesignSync Display of Milkyway Objects

DesignSync displays both the Milkyway cell view collection (<cell>.sync.mw) object and its member files in the List View. The List View shows the <cell>.sync.mw object with an SNPS icon and a Type of MW Cell View Collection.

Example DesignSync List View Display

Name	Type	Size	Modified	Version	Status	Locker	Branch
bufbd3.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
bufbd3.1	MW Cell View File	3198	08/26/2003 07:27:58		-		
nd02d0.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
nd02d0.1	MW Cell View File	3479	08/26/2003 07:27:58		-		
nd02d1.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
nd02d1.1	MW Cell View File	3482	08/26/2003 07:27:58		-		
nd02d2.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
nd02d2.1	MW Cell View File	3421	08/26/2003 07:27:58		-		
nd02d4.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
nd02d4.1	MW Cell View File	3819	08/26/2003 07:27:58		-		
sdnrq1.sync.mw	MW Cell View Collection	1 files		1.1	Up-to-date		Trunk
sdnrq1.1	MW Cell View File	8074	08/26/2003 07:27:58		-		

The Scheme API

The stcl command

The DesignSync MW environment provides a command called **stcl**, which takes a quoted string and passes it to DesignSync. (**Note:** stcl is tcl with DesignSync extensions, so the stcl command takes any tcl expression.) Output goes to the console window.

Note: DesignSync commands invoked via DesignSync MW's **stcl** command do not use the command line defaults system. The command line defaults system only pertains to DesignSync command line shells.

The **stcl** command has the following limitations:

- You cannot use the **stcl** command with the **run** command. For example:

```
stcl "run /home/hazel/dss_0929003_20457.log"
```

- The **stcl** command does not accept input from the user and therefore you must use it with care. If you select a command that requires a prompt from the user, the DesignSync plug-in from inside the Milkyway environment hangs. To prevent such a problem, the plug-in attempts to block any of the built in DesignSync commands from being run this way and some of the native tcl commands such as **gets**. However, if you introduce a new tcl command or if you encounter a native command that prompts, the Milkyway-based tool still runs but you cannot issue DesignSync operations without exiting the tool. (**Note:** One exception to this situation is the prompt that occurs if the user interacts with a server configured for username/password challenge. In this case, the user is prompted without a problem.) To minimize the possibility of a DesignSync plug-in hang, DS MW supports the ability to flag commands as being blocked. When a command is blocked, passing it to stcl results in an error message instead of a hang.
- Upon DS MW installation, the following commands are blocked:
 - **ci** when neither the **-comment** nor the **-nocomment** option is specified. **Note:** To make the **ci** command work with comments, you must escape the double quotes. For example:

```
ci -comment \"fixed routing problem\"
```

- **co -lock** when neither the **-comment** nor the **-nocomment** option is specified.
- **more**
- **gets**
- **exit**

Blocking Other DesignSync Commands

To block other DesignSync commands, add calls to **stcl_blocked** in your `.avntrc` file. This file is sourced when a DesignSync client process starts.

stcl_blocked has the syntax:

```
stcl_blocked "<space-separated command list>"
```

For example, if you want to block `command1` and `command2`, you would put the following lines in your `.avntrc` file:

```
(stcl_blocked "command1 command2")
```

The `stcl_blocked` mechanism does not cover the instance where the command that hangs the DesignSync plug-in is specified in the middle of a tcl expression. For example:

```
command;hanging_command
```

```
set variable [hanging_command]
```

Each command in the argument is compared with the first word in the command supplied by the user. In the example, if user enters the following command line, the DesignSync plug-in displays an error but does not hang.

```
stcl "command1 -switch1 -value1 -switch2 -value2 arg1 arg2"
```

Note:

- If a double quote is required inside the command, you must escape the double quote with a backslash (`\`). For example:

```
stcl "ci -new -comment \"Testing\" file 2"
```

- If you want to nest a double quote inside another quoted segment, you must use a `\\` sequence. For example:

```
stcl "ci -new -comment \"Testing \\\"nested\\\"\" \" file2"
```

The `dbRebuildLibDS` Command

To rebuild a library, DS MW users should use the **dbRebuildLibDS** command instead of the Synopsys `dbRebuildLib` command.

The **dbRebuildLibDS** command, like the **stcl** command, runs within the DS MW environment. This command performs the same function as the Synopsys **dbRebuildLib** command, except that the **dbRebuildLibDS** operation processes cells that are on the file system, regardless of whether they are read only or read/write. (The **dbRebuildLib** command does not process read only files. See Synopsys documentation for information on **dbRebuildLib**.)

The syntax of the **dbRebuildLibDS** command is:

```
dbRebuildLibDS <library path>
```

Where:

<library path> is the absolute path to your library. Enclose the path in double quotation marks (" ").

For example:

```
dbRebuildLibDS "/home/ted/ProjData/Libraries/macro_cell_library"
```

Note: You can also perform a rebuild of a library from a DesignSync shell. For information, see the **mw rebuildlib** Command in the ENOVIA Synchronicity Command Reference.

Putting a Library into Reference Control File Mode

To put a library into reference control file mode, use these Scheme calls:

```
dbSetLibRefCtrlFileMode "libname" #t
dbSetRefLibControl "libname" "refs.txt"
```

Where:

libname is the name of the library

The `refs.txt` file contains an entry of the form:

```
LIBRARY /path_to_library
REFERENCE /path_to_reference
```

To clear the reference control file mode on a library, use this Scheme call:

```
dbClearRefLibControl "libname"
```

Note: If you want to check in these changes, you must export the Other Library Information. (From the Synchronicity menu, select **Export Library Information**.)

Customizing

Controlling the Check-Out Operation's Handling of Local Versions

When it checks out a Milkyway collection object, the DS MW check-out operation first removes from your workspace any local version that is unmodified. Then the check-out operation fetches the object from the vault (with the local version number it had at the time of checkin).

To determine how to handle modified local versions in your workspace (other than the current, or highest numbered, local version), the check-out operation follows the DesignSync registry setting for **SaveLocal**. By default the setting is **Fail if local versions exist (-savelocal fail)**.

As DesignSync administrator, you can change this default behavior to instead either save these local versions for later retrieval or delete them. Use the **Local Versions (-savelocal)** setting on SyncAdmin **Command Defaults** options pane to change the registry setting. For information, see Command Defaults in SyncAdmin Help.

Users can override the default behavior from DesignSync in either of two ways:

- From a DesignSync command shell: Specify the **-savelocal** option of the DesignSync **co** and **populate** commands. For information, see the ENOVIA Synchronicity Command Reference **co** or **populate** commands.
- From the DesignSync GUI: Select an option from the **Local Versions** section of the **Check Out** or **Populate** dialogs. For information, see DesignSync Data Manager User's Guide: Checking Out Design Files and Populating Your Work Area.

Getting Assistance

Using Help

ENOVIA Synchronicity DesignSync Data Manager 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 ENOVIA Synchronicity DesignSync Data Manager tools installation. See SyncAdmin Help for details.

To bring up the the *DesignSync Data Manager MW User's Guide* from the tool you are using, do one of the following:

- Select **Synchronicity => DesignSync MW On-line Help** from the tool you are using. The help system opens in your default browser. The Contents tab displays in the left pane and the corresponding help topic displays in the right pane.
- Click **Help** on forms. The help system opens to the topic that describes the form.

To bring up the *DesignSync Data Manager MW User's Guide* from a browser, do one of the following:

- Enter the following URL from your Web browser:

```
http://<host>:<port>/syncinc/doc/Milkyway/milkyway.htm
```

- where <host> and <port> are the SyncServer host and port information. Use this server-based invocation when you are not on the same local area network (LAN) as the DesignSync installation.
- Enter the following URL from your Web browser:

```
file:/// $SYNC_DIR/share/content/doc/Milkyway/milkyway.htm
```

where SYNC_DIR is the location of the DesignSync installation. Specify the value of SYNC_DIR, not the variable itself. Use this invocation when you are on the same LAN as the DesignSync installation. This local invocation may be faster than the server-based invocation, does not tie up a server process, and can be used even when the SyncServer is unavailable.

Bringing Up DesignSync Data Manager MW User's Guide from the Command Line

DesignSync MW (DS MW) has a command-line interface. To bring up the online help from the command line:

1. Invoke a DesignSync command-line shell (dssc, stcl, stclc). For example:

```
% stclc
stcl>
```

2. Enter the **mw doc** command:

```
stcl> mw doc
```

DS MW invokes your default Web browser and displays the *DesignSync Data Manager MW User's Guide*.

Finding Information in the Online Help

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 gives you access to the Contents, Index, and Search navigation tools.
- **<<** and **>>**: Clicking these buttons moves you to the previous or next topic in a series within the help system.

You can also use your browser navigation aids, such as the **Back** and **Forward** buttons, to navigate the help system.

Related Topics

[Accessing Product Documentation](#)

Accessing Product Documentation

In addition to the online help, access to the complete documentation set is provided through the ENOVIA Synchronicity DesignSync Data Manager Product Documentation page.

To access the Product Documentation page, click ENOVIA Synchronicity DesignSync Data Manager Product Documentation.

To access the Product Documentation page from outside this online help system:

- **On Windows**, choose **Start =>Programs =>ENOVIA Synchronicity DesignSync Data Manager V6R2013 =>ENOVIA Synchronicity Documentation**.
- On **UNIX**, point your browser to:

```
file:/// $SYNC_DIR/share/content/doc/index.html
```

- Using a browser on any platform, point to:

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

Related Topics

Using Help

Contacting ENOVIA

- For solutions to technical problems, please use the 3ds web-based support system:
 - <http://media.3ds.com/support/>
- From the 3ds support website, you can access the Knowledge Base, General Issues, Closed Issues, New Product Features and Enhancements, and Q&A's. If you are not able to solve your problem using this information, you can submit a Service Request (SR) that will be answered by an ENOVIA Synchronicity Support Engineer.
- If you are not a registered user of the 3ds support site, send email to ENOVIA Customer Support requesting an account for product support:
- enovia.matrixone.help@3ds.com

Related Topics

Using Help

Getting a Printable Version of Help

DesignSync Data Manager MW User's Guide

The *DesignSync Data Manager MW 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.

DesignSync Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

acadmin

A command-line interface for managing access controls. DesignSync also provides a graphical Web-based interface, `Access_Administrator`.

accelerator key

An accelerator key is a key associated with a menu choice. Pressing the accelerator key is acts as a shortcut to selecting the menu choice. For example, pressing **F7** has the same effect as selecting **Revision Control => Check In**.

Access Administrator

A web-based interface for managing access controls graphically. DesignSync also provides a command-line interface, `acadmin`.

access control

A means of protecting design objects so that only an authorized user or user group can perform operations on objects. The access control commands also let you limit the operations and design objects these users can access. See also the Access Control Guide.

active selection

At any given time, both the tree view and list view can contain a set of selected objects. However, only one of these sets is the active selection. Most menu choices operate on the active selection. The view that does not contain the active selection indicates this by using a faded color to highlight selected objects. You can make either the tree view or the list view the active selection by clicking on the view or setting the keyboard focus to that view.

add

An operation to designate an object not currently under revision control as a member of a module.

administrator

A person who sets up an ENOVIA Synchronicity tool, for example DesignSync or ProjectSync for use by others. The set-up can include defining access_controls, note types, command defaults, client defaults etc.

aggregate view data

The candidate set of module members that results when you apply more than one module view to a workspace. DesignSync uses the **initial view data** from each view to build an aggregate view which is then populated. Filters and hrefilters can then be applied to the aggregate view data which results in the data populated into the workspace.

Apache

A public domain web server. The widely used Apache server is developed and maintained by the Apache Server Project. See the Apache web site, www.apache.org, for more information.

authentication

Verification performed by a software product to ensure that a particular user has the right to use the product or components of the product. DesignSync provides the capability of protecting your design data and preventing unauthorized users from accessing your data through the access control commands. See also DesignSync Data Manager User's Guide: Accessing a SyncServer with User Authentication, Access Control Guide: User Authentication Access Controls.

auto branching

A type of branching in which a new branch is created automatically if one does not already exist as part of a checkin or checkout with lock. Auto-branching useful when a developer wants to explore a "what if" scenario.

B

base directory

See module base directory.

bookmark

A placeholder you create in DesignSync to let you quickly visit a working directory, file, or vault you access often. See also DesignSync Data Manager User's Guide: Bookmarks.

branch

A thread of development of a managed design object that diverges from the main development thread of the object. The main branch (Trunk) is designated with branch number 1 and files on this branch have version numbers 1.1, 1.2, 1.3, and so on. The branch number always has an odd number of digits and the version number for a design object always has an even number of digits. For example, if you create the first branch from version 1.2 of a design object, the branch is designated 1.2.1 (a second branch would be designated 1.2.2) and the first version on that branch is 1.2.1.1. In this example, version 1.2 of the design object is referred to as the branch point version. See also DesignSync Data Manager User's Guide: Creating Branches.

branch-point version

A design object version that is the root of a new branch. For example, if you create a new branch off version 1.2 of a design object, that version is the branch-point version. If this is the third branch off this version, it is designated 1.2.3, and the first version on the new branch is 1.2.3.1. See also DesignSync Data Manager User's Guide: Creating Branches.

branch tag

A tag can be applied to either a branch or an object version. A tag applied to a branch is called a branch tag. See also tag, immutable, mutable, version tag, DesignSync Data Manager User's Guide: Tagging Versions and Branches.

C

cache

- An area on a LAN (local area network) where common files used by a team can be stored to avoid redundant local copies. A cache gives users their own view of the design data, while minimizing disk utilization. DesignSync caches are implemented using UNIX hard links and symbolic links, depending on the system setup. For clarity, the documentation refers to these types of caches as file caches or LAN caches. See also LAN,

There is no support on Windows for cached files.

Caches are not intended for use with modules. For information on cached modules, see Module Cache.

- The directory that DesignSync uses for temporary files during operations such as comparing file versions.

cancel

To undo a checkout. Cancel removes the locks on specified checked-out objects.

category

A module category is a virtual path to a module within the module area. All modules are stored in the Modules area. Categories provide a means to separate and organize the modules into related groups. Categories are created implicitly when you specify a module path in at module creation time.

cell

The fundamental element of Milkyway design data on which users operate. A cell always contains at least one file with a name that has the form `<cell_name>:<local_version>`, for example, `route:1`. A cell may contain attached files, whose names have the form `<cell_name>:<local_version>_<number>` (where number is any integer). For example, `route:1_2`.

In the file system, the files that make up a cell are located in a view directory. See also view.

cell view collection

A collection object that represents a Milkyway cell in a Milkyway view directory. DesignSync displays this object as `cell.sync.mw`, where `cell` is the cell name, for example, `route.sync.mw`. A cell view collection includes a cell view file and additional files called cell view attachment files.

Cell View File

A member of the DS MW cell view collection. This file is the primary cell file. The file has a name that has the form `<cell_name>:<local_version>`, for example, `route:1`.

cell view attachment file

A member of the DS MW cell view collection. This file is an attached file of the cell, a file whose name has the form `<cell_name>:<local_version>_<number>` (where number is any integer). For example, `route:1_2`.

cell view non-member

A DS MW cell view file or cell view attachment file that is part of a local version that is not the highest local version of the cell in the workspace. For example, suppose a cell named `route` has local versions `route:1` and `route:2`. The file `route:2` is a cell view file but the file `route:1` is a cell view nonmember. Nonmembers are not part of the current cell collection.

A cell view non-member can also be a cell view attachment file that is no longer attached to the cell. Such a non-member can occur due to an error situation in the

Synopsys Milkyway tools. DesignSync does not consider these "unattached" attached files to be part of a collection; DesignSync identifies them as non-versionable and does not check them into the vault.

cell view lock

A file created by the Milkyway tools. This file is a temporary file and cannot be checked in to the DesignSync vault.

cell view tmp file

A file created by the Milkyway tools. This file is a temporary file and cannot be checked in to the DesignSync vault.

checkin

Process of storing a snapshot of a design object in a vault. The new snapshot is called a version. See DesignSync Data Manager User's Guide: Checking in Design Files.

checkout

Process of making a revision-controlled design object available in your local area. You can check out a locked or unlocked copy of the object, a link to a cache or mirror copy of the object, or a reference to the object. See also populate, DesignSync Data Manager User's Guide: Checking out Design Files.

client

A process communicating with a server process in order to use a service provided by the server. The service provided by the SyncServer process is the management of design objects under revision control. There are 5 DesignSync client applications: DesSync (DesignSync GUI), dss (DesignSync Shell), dssc (concurrent dss), stcl (Synchronicity Tcl), and stclc (concurrent stcl). There are also integrations (or add-ons) into clients to support management of design objects through the software used to create and edit those objects, for example DSDFII provides access to Cadence data, and DSVS provides access to Visual Studio projects. See also server, DesSync, dss, dssc, stcl, stclc, DesignSync Data Manager User's Guide: DesignSync Command-Line Shells.

client defaults

Most of the graphical forms used to run commands in the graphical clients, for example DSDFII, DSVS, and DesSync, allow you to save the options settings as the default value for the commands. When the command form is run again, the form preselects the saved defaults. If you do not wish to use those defaults, you can manually select

different settings. These client defaults are entirely independent of each other and the command defaults allowing you to finely customize your environment.

client trigger

A trigger that is executed on the client. See also trigger, DesignSync Data Manager Administrator's Guide: Triggers Overview.

collection object

A group of files that together define a design object and are revision controlled as a single object. For example, a schematic may consist of dozens or even hundreds of files within any number of folders. You operate on the design object as a single entity while letting your design tools manage the object at the file level. See also DesignSync Data Manager User's Guide: Collections Overview.

command defaults

Default values can be set for the options specified to commands run from the command line. Default values can be set for individual commands, command families (such as all "access" sub-commands), or all commands. This simplifies invocations of commands from the command line, because the user does not need to specify "-[option] <value>" for the saved default value. Command defaults can be set by the administrator for all users on the server, or by individuals for their own commands. See also client defaults, Command Reference: command defaults.

components

Files or other objects that made up a project's hierarchy.

Physical - When using DesignSync, physical components are the files and directories created automatically during the normal process of checking in a design.

Logical - A logical component is an artificial project category set up for the explicit purpose of distinguishing where notes should be attached.

compression

Conversion of a data representation into an equivalent but smaller representation requiring fewer bytes. Compression optimizes storage space and speeds up transmission of data. DesignSync compresses data in its vaults. See also DesignSync Data Manager Administrator's Guide:Turning Off Compression.

configuration

See configuration management.

configuration management

Associated design objects that together form a snapshot of a design project. DesignSync provides two methods of associating design objects: grouping the objects together into a module, or applying a common tag to the design objects. Modules can also be tagged and associated with other modules using tags or hierarchical references.

copy out

See fetch.

CVS

Concurrent Versions System. A software tool for UNIX systems that manages multiple versions of files, tracking changes and controlling shared files. CVS uses a merging work style where a design team does not lock file versions, but instead merges their edits with the latest checked-in version. See also merging work style.

D**data sheet**

The information about an object under DesignSync revision control displayed in an HTML browser or the DesSync client. The information displayed depends on the type of object. For example, the data sheet for a file in your working folder contains its lock status, modification status, version number, and associated tags. See also DesignSync Data Manager User's Guide: What Are Data Sheets?.

DES

Data Encryption Standard. A highly effective data encryption algorithm developed at IBM in 1977. DES uses 72 quadrillion (72,000,000,000,000,000) encryption keys, chosen at random during encryption. Communication between DesignSync and SyncServers uses the DES encryption algorithm for encryption. See also encryption.

DesignSync web interface

The DesignSync web interface, built into ProjectSync, provides a method of configuring your DesignSync environment and users through a Web-page based interface.

design configuration

See configuration_management.

designer role

A user who contributes to the development of block or modules in a SITaR environment.

DesSync

The DesignSync graphical user interface (GUI), one of the DesignSync client applications. To invoke the GUI, enter "DesSync" from your operating-system window (UNIX or Windows). See also client, dssc, stcl, Command Reference: DesSync command description.

domain name

A unique alphanumeric identifier for a computer resource on the Internet. A domain name is a meaningful descriptor for a web resource used in lieu of its IP address, a string of digits. An example of a domain name is host.yourcompany.com. See also IP address, DNS.

DNS

Domain Name System. A system for maintaining a distributed list of Internet domain names and their corresponding IP addresses. DNS locates the domain names and translates these into IP addresses accessible by computers on the Internet. See also Internet, domain name, and IP address.

DSDFII

ENOVIA Synchronicity DesignSync® DFII(TM) is the integration of many DesignSync® design-management (DM) capabilities into the Cadence Design Systems DFII environment. See also DesignSync Data Manager DFII User's Guide

DS MW

ENOVIA Synchronicity DesignSync® MW(TM) (DS MW) is the integration of many DesignSync design management capabilities into the environment of Milkyway-based tools. A Milkyway-based tool reads and writes to a set of files that conform to the Milkyway database, as defined by Synopsys. DesignSync MW provides features to manage large, geographically dispersed projects. See also DesignSync Data Manager MW User's Guide.

dss

The DesignSync shell, one of the DesignSync client applications. To invoke dss, enter "dss" from your operating-system window (UNIX or Windows). You can also enter "dss" followed by a DesignSync command to execute a single command. In most cases, use dssc (concurrent dss) instead of dss to take advantage of the concurrent client capabilities. The dss client uses syncd to communicate with a SyncServer. See also

dssc, client, Command Reference: dss command description, DesignSync Data Manager's User's Guide: DesignSync Command-Line Shells.

dssc

The concurrent version of the DesignSync shell (dss). The dssc client has several advantages over dss, because it's designed to be used concurrently with other client processes. To invoke dssc, enter "dssc" from your operating-system window (UNIX or Windows), optionally specifying a single DesignSync command to execute. The dssc client does not use syncd to communicate with a SyncServer. See also dss, client, Command Reference: dssc command description, DesignSync Data Manager User's Guide: DesignSync Command-Line Shells.

DSVS

ENOVIA Synchronicity DesignSync® VS) is the integration of many DesignSync® design-management (DM) capabilities into the Microsoft Visual Studio® environment. See also DesignSync Data Manager for Visual Studio User's Guide.

dynamic href mode

The selector associated with the hierarchical reference is always evaluated to identify the version of the sub-module to be operated on.

E

EDA

Electronic design automation, the tools and processes that facilitate the design of electronic systems.

encryption

Process of converting data into a secure format that impedes unauthorized use of the data. The data is then "decrypted" (or "deciphered") by the intended receiver of the data using a decryption key -- the algorithm that converts the data back to its original format. A "strong encryption" is an encryption method that is seemingly impossible to break without the use of the decryption key. DesignSync encrypts communications between DesignSync and the SyncServers to protect your design data from unauthorized use using the DES encryption algorithm. See also DES, DesignSync Data Manager Administrator's Guide: Overview of Secure Communications.

end user

A person who uses the system, but performs no system or site wide configuration or maintenance. The end-user benefits from the administrator's set-up. In a SITaR environment, both the designer role and integrator role are end-users.

external module

A link from the DesignSync change management system (CMS) to another CMS. The files within the external CMS are managed inside DesignSync as a module. This allows you to fetch a complete code base into a workspace, even when the files are managed by different change management application, and provides a common application interface for populate operations.

For more information on external modules, see DesignSync Data Manager System Administrator's Guide: Overview of External Modules.

F

fetch

To obtain a version of a design object without locking the object. You fetch an object by performing a populate, checkout, or get operation on objects without a lock. You can fetch an unlocked copy of the object, a link to a cache or mirror copy of the object, or a reference to the object. The integrations do not support fetching object references. See also DesignSync Data Manager System Administrator's Guide: Defining a Default Fetch State, DesignSync Data Manager User's Guide: Checking out Design Files.

file

A file is a collection of data stored in one unit, under a filename. This can be a document, a picture, an audio or video file, a library, an application, or other collection of data.

firewall

A set of programs running on an organization's gateway server that acts as a virtual barrier, protecting the organization's intranet from outside access. See also gateway server, intranet.

fixed tag

A logical tag construct applied once to a single snapshot of a set of object versions and never moved. This tag may be either mutable or immutable. A tag is considered "fixed" if you keep the tag associated with a particular snapshot rather than reapplying the tag to newer versions of the design objects. There are no special attributes for DesignSync tags to indicate whether they are fixed or movable. These terms are used to describe

ways in which you can use design configuration tags. See also tag, movable tag, DesignSync Data Manager User's Guide: Using Tags.

folder

A file system directory.

G**gateway server**

A computer on an organization's intranet through which all communications between the intranet and the WWW are transmitted. Firewall software runs on the gateway server, ensuring that only authorized communications occur. See also firewall, intranet,.

GDM

Generic Data Manager. A Cadence Design Systems API (application programming interface) used to integrate design management tools into the Cadence architecture. DSDFII follows the GDM standard and can therefore be used with GDM-based interfaces, such as Library Manager.

get

Term used for a populate or checkout operation in DSVS. See also fetch.

groupware

Collaborative management tool used by a group of people.

H**Hard link**

A hard link (UNIX only) is an additional name for an existing file. Any number of hard links, and thus any number of names, can be created for any file. Hard links cannot cross file system boundaries or span across file partitions. The operating system does not distinguish between the original file name and any hard links that are subsequently created to that file; other than they are multiple names for the same file because both point to the same inode. The ownership of the hard link is defined by the original file, not the creator of the hard link.

Hard links and symbolic links are used to support the DesignSync file cache feature.

For information on cache usage, see the *ENOVIA System Administration Guide: Introduction to Data Replication*.

Hercules

A Milkyway-based tool that runs layout verification.

hierarchical reference

A dependant link from a module to any of the following: a module branch or version, a legacy module, a DesignSync vault, or an IP Gear deliverable. The hierarchical reference link allows you to build a hierarchical structure to model your project usage.

The SITaR model is one example of how hierarchical references can be used to facilitate controlled code reuse.

href mode

See `hierarchical_reference`, normal href model ; static href mode, dynamic href mode.

http

The portion of a URL address that indicates the protocol used by the software to determine how to access an Internet resource. The "http" protocol supports such resources as HTML files, CGI applications, and Java applets. See also URL `https`.

https

The portion of a URL that indicates the protocol used by the software to determine how to access an internet resource. The "https" protocol supports Secure Sockets Layer (SSL) communication. See also `encryption`, `URL`.

I

immutable

An immutable object, usually a tag, that cannot change. For example, when a module is created, branch 1 is also created with an immutable tag of "Trunk". This tag is always associated with this branch. You can also add immutable tags to a version or branch when you want to uniquely and permanently identify the branch or version. An immutable tag can be used to designate a fixed tag. See also: `mutable`, `movable tags`, `tag`.

integrator role

A use who verifies the stability of blocks or modules, and assembles them into functional packages to provide a qualified and stable integrated code baseline in a SITaR workflow environment.

initial data

The module members that are populated into the workspace when there are no module views and no filter, exclude or hrefilters applied to the data.

initial view data

The contents of a single module view with no filters, excludes, or hrefilters applied to data. See also **aggregate view data**. If you load a single module view into the workspace, this is the set of candidate module members to populate. The data in the workspace may be further refined by the applying filters and hrefilters.

integrations

Collective term for software packages that have plug-in or add-on DesignSync capabilities, for example, DSDFII, or DSVS.

Internet

A world-wide system of computer networks initially developed by the Advanced Research Projects Agency (ARPA) of the US government. The Internet encompasses the networks and protocols (computer communication conventions) coordinating the transfer of data between computers.

intranet

A network of computers internal to a company or other organization protected against unauthorized access.

IP address

Internet protocol address. A unique identifier for the location of a computer resource on the Internet. An IP address is a string of digits, for example, 127.0.01. See also domain name, DNS.

IP Gear deliverable

A IP Gear deliverable is a package of data that has been uploaded and associated with an IP Gear Catalog Component.

J**K****keyword**

Placeholder used as a comment in a revision-controlled object to express information about the design object, for example, the author (\$Author\$), check-in time (\$Date\$), or

revision number (`$Revision$`). The values of keywords are expanded when you check in the object. See *DesignSync Data Manager User's Guide: Revision Control Keywords Overview*.

L

LAN

Local area network. A group of interconnected computers able to access data on other machines on the local network. Using DesignSync, a team can define an area on a LAN as a cache to share access to revision-controlled design objects. See also *cache*.

legacy module

A grouped set of DesignSync objects that represent one level of a design hierarchy. Legacies modules can be grouped into configurations, aliases, or releases, but the individual objects are managed independently. Legacy modules were obsoleted with version 5.0 which introduced a new type of module managed as a single entity in DesignSync. Legacy modules can still be included in a module hierarchy as a sub-module to a module. Legacy modules can also be upgraded to modules. See also hierarchical reference, *DesignSync Data Manager User's Guide: Upgrading Legacy Modules*.

library

A directory tree of Milkyway data. The library has the same name as the top directory. The top directory of a library always includes the library file, a binary file called `lib`.

library file

A binary file containing metadata about a Milkyway library. Each Milkyway library has a library file, called `lib`, which resides in the top directory of the library. The information in the library file includes:

- Details of the views/cells present in the library
- Library technology information
- A list of reference libraries
- Details of library attached files
- General library properties added by various tools
- Information on logical equivalence of cells

The library file is specific to the subset of cells that has been populated to a user's workspace. Two users might be collaborating on the same library, but each may have populated a different subset of cells to the workspace and therefore each may have a different library file. DS MW properly handles this file, allowing multiple users to collaborate on the same library.

library attachment

In Milkyway, A file attached to a library. The file is named in the form `lib_X`.

library lock

The Milkyway lock file, which is a file named `.lock` that resides in the library directory. This file is a temporary file and cannot be checked in to the DesignSync vault.

library shadow collection

A collection object that encompasses the shadow files for the Milkyway `lib` file and its attached files. DesignSync displays this object as `shadow.sync.mw.lib`. A library shadow collection object contains the library file shadow and library attachment shadow files.

library file shadow

A member of the DS MW library shadow collection. This file contains a copy of the contents of the Milkyway library file (`lib`). The file has the name `shadow.lib`.

library attachment shadow

A member of the DS MW library shadow collection. This file contains a copy of the contents of the `lib_X` attached file. The file has the name `shadow.lib.X`.

An export Other Library Information operation copies the library file (`lib`) to the `shadow.lib` file and copies the library attached files (`lib_X`) to `shadow.lib.X`. Together these files form the library shadow collection object (`shadow.sync.mw.lib`). The export Other Library Information operation then checks in the library shadow collection object to the DesignSync vault.

When you import Other Library Information (using the Import Library Information form) DesignSync fetches the library shadow collection object from the vault to your workspace and loads the Other Library Information into your library file.

link

See symbolic link.

Local version

Multiple versions of a cell in Milkyway can exist in a library at the same time. Milkyway-based tools always operate on the version with the highest number in the library. This DS MW documentation refers to this as the local version, to distinguish it from a

DesignSync version, which is created in the DesignSync vault at check-in time (a vault version).

When a Milkyway object is checked in to the DesignSync vault, DesignSync applies a tag that indicates the object's local version number. If the local version is 6, DesignSync applies the tag `MW_6`.

lock

Attribute of a design object indicating that you or another team member is using and most likely editing the object. The owner of the locked object has the exclusive right to check it in to create the next version. See also locking work style.

locking work style

A team work style where each member locks an object while editing it, preventing other team members from checking in local modifications as new versions, until the locker checks in again. See also DesignSync Data Manager User's Guide: Locking or Merging Work Style?.

look & feel

The DesignSync GUI client allows you to customize the look and feel to determine the appearance and behavior of the UI. For example, the appearance of the tool bar, and the accelerator keys for the Cut, Copy, and Paste operations, depend on the look and feel. There are three look and feels available: Java (Metal), Motif, and Windows (not available on UNIX platforms).

M

merge

A merge allows you to combine your locally modified version of a design object with the Latest version of the object, resolving inconsistencies between the versions. In the merging work style, team members check out or populate without locking objects, thus, merging is required to reconcile the streams. Team members can "merge to" a branch to align the local metadata with the target branch, or "merge from" a branch to leave the metadata aligned with the current branch. See also merging work style.

merge edge

A merge edge indicates an extra "derived from" version for a version. The merge edge information is stored in the vault. When you perform a subsequent merge on the vault object, the merge edge is used to determine what changes have already been made, improving the efficiency of the merge.

merging work style

A team work style where team members check out objects without locking them. Multiple team members can edit the same object at one time; the first checked-in version becomes the Latest version and team members have to merge their edits in with this Latest version. See also DesignSync Data Manager User's Guide: Locking or Merging Work Style?.

metadata

Information about the objects that DesignSync manages, including the object's vault, branches (including current branch), versions and version-last-retrieved, its status (locked, unlocked, retired), its state as a local object (copy, locked copy, reference, link to cache or mirror directory), configuration tags applied to the object, and timestamps of operations on the object. Metadata is stored both on the server and locally. Local metadata is stored in `.SYNC` folders on your local file system. Note: DesignSync manages these `.SYNC` metadata folders; do not directly manipulate these folders or their contents. See also DesignSync Data Manager User's Guide: Metadata Overview.

Milkyway

The name of a database format, data model, and a directory structure that tools can adopt.

mirror directory

A directory that is automatically updated to contain the data set defined by a project leader for your project vault. For example, your team may always want access to the Latest version of files on the main Trunk branch. Another team may always want access to the file versions with a specific tag that are on a development branch. DesignSync creates links from users' work areas to the design objects in the mirror directory, ensuring that you have the most up-to-date configuration for your project.

A mirror directory saves team members from performing populate or check-out operations each time they want to access design changes made by team members. Because DesignSync mirrors are implemented using UNIX symbolic links, there is no support for mirrors on Windows. See DesignSync Data Manager Administrator's Guide: Mirroring Overview.

module

A module is a collection of managed objects that together make up a single entity. For example, DesignSync is a module composed of several sub-modules, such as ProjectSync and DSDFII, which contain all the code, examples, and documentation necessary to develop the applications.

The data included in a module includes its own objects and references to other modules, but not the contents of the referenced modules. See *DesignSync Data Manager User's Guide: What is a Module?* for more information.

module base directory

The top workspace directory of a module.

You specify the location of the base directory with the **populate** command using the **Working directory** option (`populate -dir`) when you get the module to your work area. (If you do not specify the **Working directory** option, the operation uses your current work area directory.)

module cache

A module cache is as a shareable workspace used for storing static modules to share with users across the network. Users link to the shareable workspace instead of to the module on the SyncServer. DesignSync Administrators or Project Managers can create module caches for modules containing common tools, libraries, or other modules needed for reference. Instead of populating the full contents of a module, UNIX users can populate a module cache link that provides access to the data without copying all the files locally. This reduces the populate time for the user as well as load on the server.

A project leader fetches modules into a module cache, preferably in "share" mode, to utilize DesignSync's file caching. When users populate module data, they can specify whether to link to modules in the module cache, or fetch modules from the server. How module data is retrieved is referred to as the module cache mode. The default module cache path, and the default module cache mode, can be defined in the registry by the team leader or by an individual user.

Note: Legacy modules can be linked to, or copied from, a module cache. The module cache mode of "copy" only applies to legacy modules. If you attempt to copy a non-legacy module from a module cache, DesignSync fetches the module from the server.

module cache link

A managed link to a module cache (mcache). The mcache link provides access to the module contents without requiring the user to load the contents of the module into her workspace by using UNIX symbolic links.

module category

A module category is a logical folder on the DesignSync server used to group modules into related groups.

For instance, tools developed and maintained for internal use, can be in a DesignSync category "Tools" which differentiate them immediately from applications being developed for external use.

module delta

Module delta mode shows only the files, folders, and hierarchical references that have been changed or added in the current version when you expand the module version on the server.

module member

A module member is any individual file, directory, or collection object added to a module.

module root

The module root of a module is the top level directory into which a module is fetched when users populate the module to their work areas.

module view

A module view is a filtered sub-group of module members. A module view definition is loaded onto the server so that the filtered sub-group is available to users who have access to the module but only need a specific set of files. For example, a project may contain source code, documentation, and compiled code. The development team may require only the source code. The release engineering team may require both the source code and the compiled code. The quality assurance team may require only the compiled code. The documentation team may require only the documentation. Using views, the team members do not have to populate the entire module or customize their own filtering at populate time.

module view definition

A module view definition is a TCL list that defines the module view. The module view definition is created in a local file on the system and then loaded or removed from the server with the `view` command set. The module view definition contains the name of the view, an optional description, and the filters and hrefilters that define the module view.

movable tag

A logical tag construct applied to a progression of snapshots of a set of object versions. A tag is considered "movable" if you reapply it to new versions of the objects as development progresses. A moveable tag must be mutable. If a tag is never meant to

be moved, it is considered a fixed tag. See also tag, DesignSync Data Manager User's Guide: Fixed Tags and Movable Tags.

mutable

A mutable object has the ability to change. In DesignSync, this term is usually used in relationship to tags. For example, the creation of module branch, it is assigned an immutable module branch tag of Trunk which can not be altered. However, you can add mutable tags to versions of this module branch such as Test, Gold, Silver, Release, December. You can move these tags as needed. Mutable and immutable tags can be used to enforce logical tag constructs like movable tags and fixed tags. See also tag.

N

natural path

The natural path is the path where that object is placed under the module base directory.

normal href mode

The selector associated with the href is examined. If it represents a static version, which means that it is a version numeric, or is a version tag, or a list of such items, then that module is fetched using the selector and the mode switches to static for sub-modules. Otherwise, the module is still fetched using the selector, but the mode remains normal for subsequent levels of hierarchy.

netlist

See RTL Team, Place and Route.

note

A note is a record of information that you attach to a project or a project configuration. You can attach notes to multiple objects, including design files. Each note is of a particular note type with specific fields, or properties, you use to enter data for ProjectSync to track. See also note type, property.

note type

A note type is a database schema of fields (properties) for users to enter data for ProjectSync to track. ProjectSync provides standard note types for the ProjectSync server administrator to install. The administrator can modify the standard note types, as well as creating custom note types. See also note, property, property type.

O

object

A DesignSync object, commonly referred to as an "object" in the DesignSync documentation, is a file, folder, module, collection, symbolic link, or other object that can be managed by DesignSync.

other library information

Information contained in the Milkyway library file other than technology information, reference libraries, and the catalog of cells for the local directory. Examples of Other Library Information are:

- Logical equivalence of cells, such as those used by Astro during optimization
- Cosmos pcells
- Properties stored by Cosmos during schematic driven layout
- A reference control file

Note: The Other Library Information can include the technology file. If your DesignSync administrator has disabled the "Manage technology file separately" option on the SyncAdmin Third Party Integration form, DS MW manages the technology file as part of the Other Library Information. See DesignSync Data Manager MW User's Guide: Managing the Technology File for information.

overlapping modules

A workspace folder can be populated with files from more than one module. These modules are considered overlapping because a single workspace directory contains members of more than one module. When you have overlapping modules, you must explicitly add any new members to the correct module before checkin. You can check in or populate the modules either independently or in a single operation.

overlay

Fetching a file specified by the selector-list, and placing it in the working directory without changing the metadata. In other words, the user remains working on the same branch as before.

P**panel**

A ProjectSync window or subwindow containing properties. Most panels behave like a graphical user interface (GUI) form element in which you enter data in fields and other GUI widgets. See also property.

persistent selector list

Specify what branch or version a command operates a version or branch is not explicitly specified. Checkin, checkout, populate, and import operations support persistent selector lists. Commands that do not use the persistent selector list typically operate on the current version or current branch of the object in your workspace. An object's persistent selector list is stored in local metadata or is inherited from the parent folder. See also selector, selector list.

place and route

One of the Milkyway-based tools (Astro) performs this step, where a netlist is used to describe which components need to be placed and which wires or 'nets' need to be routed among the components.

populate

Process of making a revision-controlled design object available in your workspace. You can populate with locked or unlocked copies of objects, links to cache or mirror copies of objects, or references to objects. The objects populated reflect those objects existing in the vault and not those objects currently residing in your local area, unlike a recursive checkout where only the objects currently residing in your local area are checked out. Populate and checkout can be used interchangeably for all DesignSync objects except modules and module members which can only be populated into your workspace. See also DesignSync Data Manager User's Guide: Populating Your Work Area.

port number

Identifier for a specific server process through which a client process sends and receives data. The registered port number for SyncServer (Synchronicity server) processes is 2647.

project

A group of revision-controlled design objects stored together as a single design effort. You can manage caches on a per project basis -- in creating a project, you associate a vault with a cache to create a local view of the design data. See also DesignSync Data Manager's User's Guide: What Is a Project?, DesignSync Data Manager's User's Guide: Design Reuse.

ProjectSync

ENOVIA Synchronicity ProjectSync is a web-based system for managing engineering projects. ProjectSync helps engineering teams track bug reports, change orders, and continuing dialogue on engineering projects.

property

A property in ProjectSync is an element of a panel, like a field or other widget on a form in the graphical user interface (GUI). A property on a note type is characterized by 4 attributes - a name, a prompt string (optional), a type, and a default value (optional). See also note, note type, panel, ProjectSync User's Guide: What Are Properties?.

property type

A data type corresponding to a property on a note type. Examples of predefined property types include Boolean, Date, Integer, Float, and String. See also property, note type.

protocol

Portion of a URL address that indicates how to access the Internet resource. The "http" protocol, for example, supports such resources as HTML files, CGI applications, and Java applets. The DesignSync "sync" protocol is used to communicate with a SyncServer, while the "file" protocol can be used to navigate your local file system. See also DesignSync Data Manager User's Guide: DesignSync URLs.

proxy

A program residing on a firewall host that intercepts communications between a company's intranet and the Internet, only forwarding a request if the requesting party can be authenticated. Proxies can also perform tasks such as caching data to improve transmission speeds -- for example, the proxy might store a previously requested web page in cached memory to improve response time for subsequent requests for the data. DesignSync provides a registry setting and environment variable so that you can specify a proxy IP address and port number. See also authentication, cache, firewall, intranet.

prune

To remove unneeded versions from a vault to free up disk space. See DesignSync Help: Deleting Versions from a Vault.

Q**R****RCS**

Revision Control System. A software tool for UNIX systems that manages multiple versions of files, tracking changes and controlling shared files. RCS uses a locking work style where a user locks a file version and thus has the exclusive right to check in the next version of that file. See locking work style.

reference

1. An object that is not physically present, but instead points to another object. The object has local metadata. When an object is checked in with the option to leave a reference to the object, the local copy is deleted, and a reference pointing to the vault replaces it. The integrations do not support references. See also Object States, regenerate.
2. REFERENCE also refers to a pointer you create in a ProjectSync project (in a `sync_project.txt` file); use REFERENCES to import modules from other design projects into your design. See also DesignSync Data Manager User's Guide: Using Vault REFERENCES for Design Reuse.
3. An ASCII file created and modified by DS MW to list the reference libraries for DesignSync maintenance.

When you set up a library (using the Library Setup Options form) or export reference libraries (using the Export Library Information form), DS MW captures the path to each reference library in your design. For each reference library in your design, the `refs.txt` file contains a line with the structure: REFERENCE path_to_library, for example:

```
REFERENCE /home/takeda/mde_lab/lab3/defoutlib
```

The `pathname_to_library` can be an absolute or relative path. The `pathname_to_library` can also take the form `$MW_REFLIB/path`, where `MW_REFLIB` specifies an environment variable and `path` specifies a path relative to the directory defined by `$MW_REFLIB`.

When you populate or when you check out a library to a new workspace, DesignSync fetches the `refs.txt` file from the vault to your workspace and loads the reference libraries the file identifies into your library file.

reference Library

A Milkyway cell in a library under DesignSync control may contain a reference to a cell in another library. The other library is called a reference library. In order to instantiate cells into your cell, you must first have declared the reference libraries with a Milkyway command, which causes a pointer to the reference library to be injected into the library file.

When you populate a new workspace, that operation cannot be completed until the library file in the new workspace is made aware of the reference libraries.

reference control file mode

Commonly, the library reference information for a Milkyway library resides in its `lib` file. In reference control file mode, library reference information resides in one of the library's attached files. This attached file is similar in form to the DS MW `refs.txt` file, except that it also contains the path to the library itself.

When a library is in reference control file mode, DS MW does not export the library references to a separate file. Instead, DS MW maintains the attached file as part of the Other Library Information.

To put a library into reference control file mode, use Scheme calls. See also DesignSync Data Manager MW User's Guide: Putting a Library into Reference Control Files Mode.

Note:

- When a library is in reference control file mode, the Reference Library field of the Library Setup and Export Library Information forms cannot be edited.
- If you import a library from the vault to a different library name in your workspace (for example, if you populate a library into a new workspace with a different name), DS MW cannot determine the name of the library and therefore cannot update the library reference information in the attached file. You must edit the attached file and correct the library references manually.

regenerate

DSDFII locked reference mode used when regenerating the object from a different source rather than editing it directly. This mode does not transfer the object from the vault. This is a more efficient mode for that use model since the user does not require the latest version of the generated file as long as the source files are correct. See also reference.

registry

The key-based repository used by DesignSync to store DesignSync settings. It is designed to mirror in functionality and layout the Windows Operating System registry. The registry supports customizing settings for the entire site, a specific server, a project, and an individual user. This allows a user to personalize the DesignSync environment.

For more information, see registry files, DesignSync Data Manager Administrator's Guide: Overview of Registry Files.

registry file

A file containing setup and tool configuration information for DesignSync tools. DesignSync uses different registry files for different functions. Some of these registry files are only for use by the DesignSync administrator while others can be customized by individual users or project managers. See also registry.

relative path

The relative path indicates the path from the upper-level module to the object it is creating the connection to. The object can be a module, legacy module, configuration, IP Gear deliverable, or DesignSync vault.

remove

An operation to remove a module member from a module. This removes the member from all future operations on the module. You can add the object back to the module later, if needed.

replica

A copy of a design object created when you fetch (check out without a lock) a version from the vault. A replica contains relational information about the original replicated object, so that DesignSync can determine, for instance, whether the original object has changed since you fetched it. See also DesignSync Data Manager User's Guide: Object States.

repository

See vault.

retire

Prevents a design object from being automatically included in populate or recursive checkout operations. Retiring an object's branch rather than deleting the object itself gives you the opportunity to reinstate the object for future populate and recursive checkouts. Module objects are never retired, they are removed from the module. See also DesignSync Data Manager User's Guide: Retiring Branches.

RevisionControl note

A ProjectSync note created when a DesignSync revision control operation takes place. The note has a built-in type, RevisionControl, that contains information about the operation, such as the name of the user, command used to invoke the operation, and execution time. See also DesignSync Data Manager Administrator's Guide: RevisionControl Notes Overview.

root directory

See workspace root directory.

RSA

Rivest, Sharmir, and Adleman Internet encryption and authentication system. RSA is a commonly used encryption and authentication algorithm embedded in most web

browsers. The algorithm uses separate keys -- a public key a sender uses for encrypting and a private key used by the owner to decrypt messages. The private key is never transmitted across the Internet, thus protecting the encryption from being deciphered. See also encryption.

RTL team

A group of people who create RTL source code, typically in Verilog or VHDL format. These people put the RTL code through synthesis to produce a netlist. A netlist serves as an entry point to Place and Route.

S

save settings

See client defaults.

SCC (Source Code Control)

DSVS features Windows-based integration that uses MSSCCI standard and is SCC compliant meaning that in addition to be integrated with Visual Studio, DSVS can be used with any other Windows application that supports an SCC plug-in.

selector

An expression that identifies a branch and version of a managed object. For example, the version tag "gold", the branch selector "Rel2:Latest", the version number "1.4", and the reserved keyword "Latest" are all selectors. Selectors are specified with -version and -branch options to various commands or through a persistent selector list. See also selector list, persistent selector list.

selector list

A comma-separated list of selectors. DesignSync uses the selector list as a search order to select a version on which to operate. If the first selector in the list results in no match, DesignSync retries the command using the next selector in the list. The command fails only if all selectors in the list produce no match. An example of a selector list is "gold,silver,Rel3.0:Latest,Trunk". See also selector, persistent selector list.

server

A computer process that provides services to client processes. The service provided by the SyncServer process is the management of the revision control and configuration tasks on the data in the DesignSync vaults. See also client.

server-side trigger

A trigger that runs on the server. See also trigger.

SITaR (Submit, Integrate, Test, and Release)

SITaR (Submit, Integrate, Test, and Release) is a simple module-based workflow for projects that consist of multiple blocks, or modules, developed by several contributors. SITaR uses hierarchical references to create a controlled integration environment for testing the interaction between the contents of modules in the hierarchy. See also, DesignSync Data Manager User's Guide: Overview of SITaR Workflow

skip

The action of skipping interim versions when checking in a new version of a design object which you did not lock, and replacing the Latest version. Skipping differs from a standard checkin in that interim versions can not be seen by team members checking out the Latest version; if you want the changes from the interim versions, you must request the versions explicitly by version number and enter their changes manually, or, if the file is not a binary file, you may be able to merge the changes into the file. Note: Skipping is an advanced operation and should be used with caution because it retires versions from the vault. You can only access the skipped versions if you enter their version numbers explicitly. See also DesignSync Data Manager User's Guide: Checking in Design Files.

SSL

A protocol to provide encrypted communications across a network. SSL is indicated by the https designated in a URL.

staging area

Intermediate folder used while relocating or converting vault folders (repositories). For example, if you move a vault folder to a new SyncServer, you first export the vault folder to an intermediate folder, then import the intermediate directory into a new SyncServer.

stcl

ENOVIA Synchronicity's customized Tcl (Tool Command Language) interpreter, one of the DesignSync client applications. The stcl client combines the text commands available in dss (the DesignSync shell) with the general scripting capabilities of Tcl. To invoke stcl, enter "stcl" from your operation-system window (UNIX or Windows). You can also enter "stcl" followed by a script name to execute an stcl script, or enter "stcl -exp" followed by a DesignSync or Tcl command in quotes to execute a single command. In most cases, use stclc to take advantage of the concurrent client capabilities. See also Command Reference: stcl command description, DesignSync Data Manager User's Guide: DesignSync Command-Line Shells.

stcl

The concurrent version of stcl. The stcl client has several advantages over stcl, because it's designed to be used concurrently with other client processes. To invoke stcl, enter "stcl" from your operation-system window (UNIX or Windows). You can also enter "stcl" followed by a script name to execute an stcl script, or enter "stcl -exp" followed by a DesignSync or Tcl command in quotes to execute a single command. See also Command Reference: stcl command description, DesignSync Data Manager User's Guide: DesignSync Command-Line Shells.

static href mode

Resolves hierarchical references to the static version of the sub-module recorded with the href at the time the hierarchical reference was created is always used. See also hierarchical_reference, dynamic reference.

substitution tag

ProjectSync extension to HTML that lets you embed placeholders for GUI widgets within a panel's HTML template. Substitution tags are coded as HTML comments in the template in the following format:

```
<!-- SYNC <substname> -->
```

where <substname> is the substitution tag name. You can define the behavior for the substitution widget by modifying the installation (.ini) file corresponding to the panel's HTML template (or by creating an initialization file for the template if one does not exist.)

symbolic link

(UNIX symbolic link) A special UNIX file that points to another file. A link takes up less space than a copy of the original file. Use the UNIX **ln** command to create symbolic links. See the UNIX **ln** documentation for more information. See also DesignSync Data Manager User's Guide: Object States, DesignSync Data Manager Administrator's Guide: How DesignSync Handles Symbolic Links.

Symbolic links are also used with mirror directories , caches, and module caches. Also see hard link.

sync

The portion of a URL address that indicates the protocol used by the software to determine how to access an Internet resource. The "sync" protocol is used to communicate with a SyncServer. See also protocol, URL.

syncd

The Synchronicity daemon process. The syncd process manages communication between dss/stcl sessions and SyncServers. Note that dssc and stlc do not use syncd; they communicate directly with a SyncServer. See also Command Reference: syncdadmin.

SyncAdmin

The Synchronicity Administrator tool. LAN administrators can use SyncAdmin to set site-wide preferences, such as the default fetch preference, how symbolic links are managed, and which cache users should access. Users can use SyncAdmin to set user preferences such as their default editor and HTML browser. SyncAdmin is a standalone tool located in <SYNC_DIR>/bin. See also DesignSync Data Manager Administrator's Guide.

sync_project.txt file

A file created when you create a project in ProjectSync. The sync_project.txt file includes the REFERENCE and CONFIG mapping statements that let you import modules from other projects into your designs. See also ProjectSync User's Guide.

SyncServer

A DesignSync server process that manages shared information, controls access to design files, and performs administrative functions such as user privilege validation. A SyncServer is an http server, with a special "sync" protocol layered on the http protocol, for example, sync://host:port/Projects/Sportster. See also server, protocol, DesignSync Data Manager User's Guide: What Is a SyncServer?.

sync_servers.txt file

A file created by a user (My Servers), project leader (Site Servers), or administrator (Enterprise Servers) that lists SyncServer or vault definitions, which facilitates entering URLs in various locations in the DesignSync graphical interface and integrations; such as the DesignSync workspace wizard and the DSDFII Join Library wizard.. See DesignSync Data Manager Administrator's Guide: SyncServer List Files.

sync_server_list File

The DesignSync installation script, sync_install, automatically generates the sync_server_list file. Do not edit the sync_server_list file.

T

tag

An identifier you attach to a group of design object versions that you want to identify as a group. You can then perform operations on the group of objects as a whole. For example, you might tag all versions that went into a release "Rel2_0" so that at a later date another user could fetch all versions tagged "Rel2_0" thereby recreating that release. Tags can also be applied to branches. See also design configuration, mutable, immutable, DesignSync Data Manager User's Guide: Tagging Versions and Branches.

Tcl

Tool Command Language, a scripting language and interpreter developed at University of California, Berkeley by Dr. John Ousterhout. Tcl interpreters are embedded in many EDA applications, including DesignSync. See also stcl, stclc.

TCP/IP

Transmission Control Protocol/Internet Protocol, the communication conventions, or protocol, for transferring data on the Internet. Computers accessing the Internet have copies of the TCP/IP program which manages data transmissions, decoding addresses and transmitting the data to the specified IP address. See also IP address, domain name, DNS.

technology file

An ASCII file (`tech/tech.tf`) created and modified by DS MW with the purpose of making DesignSync aware of the library technology data for a design.

When you populate or when you check out a library to a new workspace, DesignSync fetches the `tech.tf` file from the vault to your workspace and loads the technology data the file identifies into your library file.

Note: If your DesignSync administrator has disabled the "Manage technology file separately" option on the SyncAdmin Third Party Integration form, DS MW manages the technology file as part of the Other Library Information. See also DesignSync Data Manager MW User's Guide: Managing the Technology File.

template

ProjectSync implements note panels with HTML templates. Administrators can generate an HTML template from an existing note type, then modify the HTML code to change its appearance.

trigger

A watchpoint (a Tcl script) that causes an action, such as sending an email, to occur automatically in response to some other action, such as checking in a file. A **client**

trigger is a trigger that is executed on the client. A server-side trigger is a trigger that runs on the server.

Trigger arguments

Trigger arguments are passed to any triggers that have been set up for the operation. Consult your project leader for information about any triggers that are in use and how they use arguments.

U

unique identifier

Every object and folder in the vault of a module is assigned a unique identifier. These unique identifiers are used within the module's versions. There is a mapping from the unique identifier to a natural path. The natural path can change from one module version to another, however, changing the natural path in this manner does not change the object's unique identifier.

unlock

Remove a lock from a design object's branch. You can unlock branches you have locked or those locked by team members. Access controls can be set up to limit unlocking operations. See also *DesignSync Data Manager User's Guide: Undoing a Check Out*, *DesignSync Data Manager User's Guide: Unlocking Branches*.

URL

Uniform resource locator, an address for a resource. A URL consists of the access protocol, followed optionally by a machine name and/or the path to the object. DesignSync uses a special "sync" protocol, for example, `sync://host:port/Projects/Sportster`. Use the "file" protocol to access files on a local network. See also `http`, `https`, `sync`, IP address, domain name.

user

Any individual authorized to work with DesignSync or ProjectSync.

user group

A collection of users defined as a single unit and, as a unit, granted or restricted access to DesignSync commands or objects with the access control system. User groups are defined in **Access Administrator**.

user profile

Attributes defined in ProjectSync by an administrator to identify a user. The administrator enters the username, password, and contact information for each team member. ProjectSync can be configured to allow access only to registered users.

V

vault

The repository of the versions, including branched versions, checked in for a particular design object. Note that a vault stores versions of a single design object. A "project vault" refers to the top-level folder of a project and thus contains a vault for each revision-controlled object in the folder. See also DesignSync Data Manager User's Guide: Vaults, Versions, and Branches.

Each DesignSync object has its own vault repository to maintain the branches. DesignSync modules functionality creates an abstraction layer rather than providing direct access to the vault objects. This allows the module to function as an atomic object, rather individual objects.

DesignSync file-based functionality directly manipulates vault objects. This requires each object to be processed individually and enumerated explicitly in such operations as populate, checkout, tag, and checkin.

version

A fixed snapshot of a design object, such as a file or collection, stored in a vault with a numeric identifier. Version numbers are composed of an even number of numeric fields, separated by periods. For example the following are valid version numbers: 1.3, 1.3.1.4, and 1.2.5.1.1.1. See also DesignSync Data Manager User's Guide: Vaults, Versions, and Branches.

version-extended naming

When invoking Advanced Diff from DesignSync, you can specify the files you want to compare using version-extended filenames, which consist of the filename, followed by a semicolon (;), followed by a version number or tag. For example, alu.v;1.2 is version 1.2 of alu.v, and alu.v;golden is the version that is tagged 'golden'. You can also use the following reserved tags:

- **Orig** The version in the vault from which your local version originated; for example, alu.v;Orig.
- **Latest** The most recent version in the vault; for example, alu.v;Latest. Often Latest and Orig are the same version. For example, you fetch alu.v;1.4, which is the most recent version in the vault. Version 1.4 is both Latest and Orig. If your teammate then checks in version 1.5, Orig is still version 1.4, but Latest is now version 1.5.

DesignSync interprets these version-extended filenames and fetches files from the vault as needed, placing them in your DesignSync cache. These cached files may be used by the defined graphical Diff utility ; DesignSync does not use these files and users will not get links to these files if they populate from the cache.

version tag

A tag can be applied to either an object version or a branch. A tag applied to an object version is called a version tag. See DesignSync Data Manager User's Guide: Tagging Versions and Branches.

view

A way to categorize cells. Conceptually, a cell may have many different views--a place and route view, an abstract view, and so on.

In the Milkyway file system, the view is a directory beneath the library directory. The view directory contains all of the cells of that view. (Cells are made up of files in the view directory.) The file system hierarchy has this structure:

Library directory => View directory => Cell view files

W

workspace

A DesignSync workspace is a folder on your local system allows a user to gather various objects, generally under revision control, and work with them as a cohesive unit. Revision Controlled objects can be checked out or populated into the workspace, edited, and then checked in to create the next version on the server vault.

workspace root directory

See workspace root path.

workspace root path

The top folder of your workspace, where information about all modules with base directories at or below that point is stored. In DSDFII, you can manually enter a workspace root path for any modules containing Cadence libraries outside the workspace root directories already known to Cadence. See also module base directory, DesignSync Data Manager DFII User's Guide: Setting the Workspace Root Path.

workspace wizard

A series of dialog boxes, launched by **Revision Control =>Workspace Wizard**, that guide you through the process of creating your own project or joining an existing project. See also [Workspace Wizard Overview](#).

X

Y

Z

Index

.		
.avntrc file	81	
C		
Cancel	43, 45	
Cell		
cancellation of the checkout of	45	
checkin of	29	
checkout of	38	
datasheet for	51	
dependencies between views of	25	
purging versions of	75	
tagging of	60	
version history of	53	
D		
Datasheet	51	
dbRebuildLibDS command	81	
DS MW	3	
online help for	87	
overview of	3	
release information for	1	
setup of	19	
using	7	
L		
Library		
cancellation of the checkout of	43	
checkin of	27	
checkout of	35	
export of other information for	67	
export of references for	67	
import of other information for	70	
import of references for	70	
rebuild of	81	
setup of	21	
tagging of	55	
Local version	85	
checkout handling of	85	
M		
Milkyway objects	3	
checking in	27, 29	
deleting from the vault	75	
fetching from the vault	35, 38, 49	
handling of	77	
Mirror mode	77	

O

Other library information

- export of..... 67
- import of..... 70

P

Populate..... 49

purge..... 75

- command for..... 75

R

Rebuild library command..... 81

Reference control file mode 83

refs.txt

- Exporting Library Information..... 67
- Setting Up a Library 21

S

Scheme 81

stcl command 81

T

Technology file 67

- export of..... 67
- import of..... 70
- specifying DS MW treatment of 20

V

Versions 53

- history of 53
- purging..... 75

View

- specification of dependencies
between 25