Contents

1 Introduction to the VersionSync Design Management System ........................................... 5
   Overview ............................................................................................................................. 6
   VersionSync and DesignSync DFII: a Comparison ........................................................... 6
   How VersionSync Works with GDM ............................................................................... 7
   The VersionSync Data Model ............................................................................................ 8
       Workareas ..................................................................................................................... 8
       Repositories .................................................................................................................. 9
       Co-managed Sets ......................................................................................................... 10

2 Using VersionSync ........................................................................................................... 11
   Overview ........................................................................................................................... 12
   Configuring Your Environment ......................................................................................... 12
   Specifying VersionSync as Your Design Management System ........................................ 13
   Creating a VersionSync Workarea ..................................................................................... 13
       Using the Default Repository Location ....................................................................... 14
       Specifying the Repository Location ............................................................................ 15
       Sharing Workareas and Repositories ........................................................................... 15
   Performing Design Management Operations ..................................................................... 17
       Using VersionSync from Library Manager ................................................................... 17
       Using VersionSync from Project Manager .................................................................... 18
       Using VersionSync from GDM Shell Commands ......................................................... 18
       Using RCS Keywords .................................................................................................. 22
   Example of Using VersionSync ......................................................................................... 23

3 System Administration ...................................................................................................... 26
   Overview ............................................................................................................................ 27
   Controlling Access to a Repository ................................................................................... 27
   Reorganizing a Repository ................................................................................................. 27
VersionSync Guide

Renaming Vaults ................................................................. 27
Renaming Vault Directories .................................................. 28
Understanding Performance Issues ....................................... 28
Providing Adequate Disk Space ........................................... 30
Backing Up Data ................................................................. 31

4
Command Reference ............................................................ 32

vsinfo .................................................................................. 33
vssetvault ............................................................................ 35
Introduction to the VersionSync Design Management System

This chapter contains the following sections:

- “Overview” on page 6
- “VersionSync and DesignSync DFII: a Comparison” on page 6
- “How VersionSync Works with GDM” on page 7
- “The VersionSync Data Model” on page 8
Overview

VersionSync™ is the name of the Cadence® basic design management (DM) system provided by Synchronicity Software, Inc. You use VersionSync with Cadence applications that support Cadence’s Generic Design Management (GDM) facility, such as Cadence Design Framework II (DFII), to manage your design data—Cadence library data (libraries, cells, cell views) and UNIX files.

This document describes user topics such as creating a VersionSync workarea and performing DM operations, and system administrator topics such as controlling access to repositories and backing up data.

This document assumes you are familiar with GDM, DFII, and Library Manager. For more information on these topics, refer to the following documents in the Cadence online documentation system:

- *Cadence Application Infrastructure User Guide*
- *Design Framework II Help*
- *Library Manager Help*

VersionSync and DesignSync DFII: a Comparison

A team sharing Cadence data has three design management (DM) options available:

- **No DM**
  
  Not using DM is impractical. Trying to manage data created by multiple designers without a DM system is cumbersome and can result in lost changes or even corruption of Cadence data.

- **VersionSync**
  
  VersionSync provides basic DM capabilities aimed at small design teams. VersionSync’s support for version control and workspace management lets small teams manage and coordinate their work. VersionSync is lightweight, easy to install, and easy to use. However, VersionSync does not scale to large teams and does not support cross-geographic teams.

  VersionSync is free to all Cadence users and can be obtained from Cadence. Support for VersionSync is provided by Cadence.

- **DesignSync® DFII**
DesignSync DFII is a sophisticated DM system that is targeted at large design teams and teams with more advanced DM requirements. DesignSync DFII's web-based architecture lets remote teams easily access and manage project data.

DesignSync DFII provides superior DM integration into the Cadence DFII environment. In addition to Library Manager integration through Cadence's GDM facility, DesignSync DFII provides Synchronicity pull-down menus from many DFII user interfaces. These Synchronicity menus make DesignSync DFII easy to use and also provide more sophisticated DM features than those available through GDM.

In addition, DesignSync DFII
- Manages all the files in your design flow, from initial requirements to the final layout
- Manages cell view files as atomic collections to ensure data integrity
- Provides security through user authentication, access controls, and SSL encryption
- Provides file caching methodologies that reduce disk space requirements

DesignSync DFII can be purchased from Synchronicity, Inc.

How VersionSync Works with GDM

The Cadence generic design management (GDM) facility is the interface through which Cadence applications interact with a design management (DM) system. GDM supports the DM operations that are commonly used by designers, such as check in, check out, cancel, and delete. These DM operations are available in several ways:

- As GDM shell commands
- Through interfaces such as Library Manager and Project Manager
- Through other GDM-based operations, such as Auto Checkin and Auto Checkout

GDM specifies a set of operations that an underlying wrap (or wrapper)—a GDM integration such as VersionSync—supports. These operations may be implemented either as shell commands or as part of a file information server. The VersionSync file information server is vsfisrv. The vsfisrv process, which runs as a background process, is started automatically by GDM when you request a DM operation. The process ends when you exit a tool such as Library Manager, or after a short period of inactivity (typically five minutes) if you are using GDM shell commands.

Regardless of whether VersionSync implements a given operation as a shell command or through vsfisrv, you do not call VersionSync commands directly—GDM calls VersionSync.
You execute your DM operations using GDM shell commands or through any application that uses GDM (such as Library Manager), which in turn calls VersionSync.

For more information about GDM, see the Cadence Application Infrastructure User Guide.

The VersionSync Data Model

The following sections describe the elements that make up the VersionSync data model. Many of the same terms and concepts are used by GDM and are described in the Cadence Application Infrastructure User Guide.

Workareas

A workarea is the directory hierarchy where you create and modify your designs. A workarea contains one or more Cadence libraries. The workarea can be associated with a single VersionSync repository, or each library can have its own repository. The top level of a workarea is called the workarea root directory.

Files that are version controlled by VersionSync are said to be managed. Files in the workarea can be but do not have to be managed by VersionSync.

When you check out a file, it has write permissions in the workarea and its repository counterpart is locked. These permissions enable you to edit the file and prevent other users from checking out that version of the file. Only one user can have a lock on a version of a file.

When you check in a file, VersionSync

- Sets the file in your workarea to read only, which prevents unintended changes to the file
- Releases the lock, allowing other users to check out and modify the file

The workarea root directory also contains the following elements, which provide VersionSync and GDM with information about the contents of the workarea and the location of the repository:

- .VSVAULT
  Can be a file that contains the path to the repository directory or the repository directory itself. The .VSVAULT file or directory is created by VersionSync when you set up your workarea (see “Creating a VersionSync Workarea” on page 13).
- cdsinfo.tag
This file is used by GDM to determine the design management (DM) system being used to manage the files within the workarea. When VersionSync is the DM system, the \texttt{cdsinfo.tag} file contains the following line:

\begin{verbatim}
DMTYPE vs
\end{verbatim}

See “Specifying VersionSync as Your Design Management System” on page 13 for more details. The \texttt{cdsinfo.tag} file typically contains information other than the DM system. For a full description of the \texttt{cdsinfo.tag} file, see the \textit{Cadence Application Infrastructure User Guide}.

## Repositories

A \textbf{repository} is a VersionSync-managed directory hierarchy that stores file and history information for managed files. VersionSync creates a new repository or associates an existing repository with a local directory when you create a workarea. The repository can be shared simultaneously by a number of users, assuming the UNIX permissions allow it.

\begin{important}
VersionSync manages the repository. Do not directly modify files in the repository.
\end{important}

Each managed file in a repository has a vault. The \textbf{vault} stores the versions and version history of the file. A \textbf{version} (sometime called \textit{revision}) is an immutable instance of a file—a fixed snapshot at a given point in time. Each time a user checks out, modifies, and checks in a file, a new version is created. All versions have a corresponding version number. The initial version is 1.1. Subsequent versions are 1.2, 1.3, and so on. The leading “1” in each of these version numbers identifies the branch that the version is on. Branch 1 is the main or trunk branch.

\textbf{Note}: VersionSync only supports development on the main branch (branch “1”).

Vaults have the same name as the files in your workarea, but with a ,\texttt{vs} suffix. For example, a file named \texttt{alu.v} has a vault named \texttt{alu.v,vs}. Vaults are read-only so as to prevent unintended modifications. VersionSync commands update the vaults.

The repository structure parallels that of your workarea. \textbf{Figure 1-1} shows a typical workarea containing a library with its repository located in the library directory.
Note: When a .VSVAULT is located in a library directory, there is no corresponding library directory in the repository.

Co-managed Sets

A co-managed set, as described in the Cadence Application Infrastructure User Guide, is a group of files that are version controlled as a single object. All files in the set always have the same version number, although not all files may exist for a given version of the set. For example, a prop.xx file may not exist in early versions of a cell view but may get added later.

GDM identifies co-managed sets and groups the appropriate files for VersionSync. VersionSync then manages each of the individual files at the vault level. When operating on a co-managed set, VersionSync outputs status messages for each file in the set.

In the Cadence library structure, cellviews are co-managed sets.
Using VersionSync

This chapter contains the following sections:

- “Overview” on page 12
- “Configuring Your Environment” on page 12
- “Specifying VersionSync as Your Design Management System” on page 13
- “Creating a VersionSync Workarea” on page 13
- “Performing Design Management Operations” on page 17
- “Example of Using VersionSync” on page 23
Overview

This chapter discusses the tasks associated with using VersionSync™. The following diagram illustrates the typical flow:

Configure Your Environment
Ensure that you are set up to use VersionSync.

Specify VersionSync as Your DM
Ensure that your `cdsinfo.tag` defines `DMTYPE vs.`

Create Your VersionSync Workarea
Let VersionSync do this for you, or use `vssetvault`.

Import Existing Designs
Use `gdmimport`.

Create New Design Data
Use Cadence design tools.

Check In
Check files into the repository with `Library Manager` or `gdmci`.

Check Out
Check out files for editing with `Library Manager` or `gdmco`.

Configuring Your Environment

VersionSync can be installed within the Cadence® installation hierarchy or separately. You or your VersionSync administrator should follow the installation instructions provided with the VersionSync package.

To verify that VersionSync is configured, type
vsinfo

The vsinfo command should display the VersionSync version number, and workarea and repository information. If vsinfo does not execute, your environment has not been configured to use VersionSync. Typically, your PATH variable has not been set properly. Refer to the VersionSync installation instructions.

You must also be set up to use Cadence tools. If you are not, review the Cadence Installation Guide or contact your Cadence tools administrator.

See “vsinfo” on page 33 for details on the vsinfo command.

Specifying VersionSync as Your Design Management System

Cadence cdsinfo.tag files contain entries for various properties, including what design management (DM) system to use. The DMTYPE property, when defined in a cdsinfo.tag file, specifies the DM system. An entry of DMTYPE none means you are not using DM. No DMTYPE entry means another cdsinfo.tag file will determine the DM system.

➤ To make VersionSync your DM system, you or your project leader must add the following line to the appropriate cdsinfo.tag file:

    DMTYPE vs

There is an installation-wide cdsinfo.tag. Entries in this file can be overridden by any other cdsinfo.tag file found using a search mechanism. For example, there may be a site cdsinfo.tag file, a cdsinfo.tag file in your home directory, or a library-specific cdsinfo.tag file. For details on the cdsinfo.tag file and this search mechanism, see the Cadence Application Infrastructure User Guide.

➤ To verify that VersionSync is defined as the DM system for a given object, go to the object’s directory and type the following command:

    cdsinfo -lookup DMTYPE

Creating a VersionSync Workarea

A VersionSync workarea is a local directory hierarchy, typically containing one or more Cadence libraries, that has a VersionSync repository associated with it. The root (top-level) directory of a VersionSync workarea contains a .VSVAULT file or directory. When .VSVAULT is a directory, then .VSVAULT is itself the repository directory. When .VSVAULT is a file, it contains the full path specification to the repository directory. Whether .VSVAULT is a
directory or file depends on whether VersionSync created the workarea automatically or whether you explicitly specified the repository using the `vssetvault` command.

**Note:** VersionSync does not obey a user’s umask when setting permissions on repository and workarea files and directories. VersionSync uses default permissions of 664 for files and 775 for directories. Use the `VSDIRPERMS` and `VSFILEPERMS` environment variables to control repository and workarea permissions. You may want to define these variables prior to setting up a workarea. See “Sharing Workareas and Repositories” on page 15 for details.

### Using the Default Repository Location

VersionSync will create a repository automatically for you within a library directory if needed. This feature is designed to make using VersionSync through Library Manager transparent. If you are using GDM shell commands directly, do one of the following:

- Use Library Manager to create a new library before calling any GDM shell commands. When you create a library from Library Manager, the `cdsinfo.tag` file is created for you and will contain the `CDSLIBRARY` entry.
- Use the `vssetvault` command to explicitly define the repository for the workarea (see “Specifying the Repository Location” on page 15).

VersionSync’s scheme for locating and creating a repository is as follows:

1. First, look in the directory containing the object to be managed and look for a `.VSVVAULT` file or directory. If `.VSVVAULT` is not found in this directory, look in the parent directory, and continue looking up the directory structure all the way to the root of the file system. The first `.VSVVAULT` located during the search is the repository for the object.

2. If no `.VSVVAULT` is found, determine the library directory for the object. A library directory is one that has a `cdsinfo.tag` file containing a `CDSLIBRARY` entry. If the library directory is identified, create the repository (`.VSVVAULT` directory) here.

3. If no library directory is located, VersionSync is unable to create a repository and returns an error. You must then properly define the library directory for the object, or explicitly specify the location of the repository using the `vssetvault` command (see “Specifying the Repository Location” on page 15).

Note that this algorithm restricts the creation of a repository to a library directory and no lower. There will never be `.VSVVAULT` directories in cell or cellview directories unless you use the `vssetvault` command, and even then it is recommended that you do not create a repository at the cell or cell-view level.
Specifying the Repository Location

You can override the default behavior of VersionSync creating repositories in library directories when no workarea repository already exists (see “Using the Default Repository Location” on page 14) by using the vssetvault command. This command explicitly associates a workarea with a repository directory.

➤ To create a VersionSync workarea with the vssetvault command, type:

```
vssetvault repositorydir workareadir
```

where repositorydir is the path to the directory you want to use as the repository and workareadir is the path to the directory that you are setting up as a workarea. If you want other users to be able to access the repository (a shared repository), ensure that the repositorydir path is accessible to them.

The vssetvault command:

- Creates the repositorydir and workareadir directories if they do not already exist, file permissions permitting.
- If the workarea already contains a cdsinfo.tag file, appends a DMTYPE vs line to the file. Otherwise, creates a cdsinfo.tag file with a DMTYPE vs line.
- If repositorydir and workareadir are the same directory, creates a .VSVAULT directory in the workarea directory. Otherwise, creates a .VSVAULT file that contains the absolute path of the specified repository directory.

Note: Defining a repository at the cell or cell-view level is not recommended. Define the repository at the library level or above.

For an example of setting up a VersionSync workarea, see “Example of Using VersionSync” on page 23.

For the more information on the vssetvault command, see “vssetvault” on page 35.

Sharing Workareas and Repositories

A shared workarea, sometimes called a sandbox, is a workarea that is used by multiple team members. Sharing a workarea requires less disk space than each user having a private workarea, each containing the same design files.

A shared repository is one that is referenced by more than one workarea. It is likely that you will be accessing shared repositories, as most design data needs to be shared by a design team.
VersionSync supports sharing workareas and repositories.

**How VersionSync Shares Workareas and Repositories**

The ability to share workareas and repositories is determined as follows:

- Workarea and repository directories must be visible to the users who intend to access them. NFS-mounting or similar network mechanisms can be used. For example, a repository directory of `/net/apollo/usr1/projects/AsicLib` will be visible to all users on the same LAN as `apollo`.

- UNIX permissions must grant access to the users of the workarea or repository.

  VersionSync does not obey a user's umask. VersionSync creates files with 664 permissions and directories with 775 permissions by default. These default permissions facilitate typical sandbox sharing models.

  Users can define the **VSFILEPERMS** and **VSDIRPERMS** environment variables to change VersionSync's default permissions. Each team member should define these variables as appropriate. For example, if you do not want other users within the same UNIX group to be able to edit files that you have checked out, define **VSFILEPERMS** to be 644.

  **Note:** The following exceptions apply to the permissions of files created by VersionSync:

  - Checked out files always have owner read and write permissions.
  - Files that are not checked out never have write permission for any user.
  - Following a check-in, files always have owner read permission.
  - `.VSVAULT` and `cdsinfo.tag` files always have user read permission.

**Common Sharing Models**

The following scenarios describe three common sharing models that are supported by VersionSync, listed from least to most permissive. These scenarios assume that all team members are in the same UNIX group and that access by people not in the group is always read only.

1. Users have read-only access to other users' check outs and read-only access to the repository (except for the repository owner, who has read/write permission).

   To implement this model, set **VSFILEPERMS** to 644 and **VSDIRPERMS** to 755.

2. Users have read-only access to other users’ check outs but read/write access to the repository.
To implement this model, set `VSFILEPERMS` to 644 and leave `VSDIRPERMS` undefined (or set to 775).

3. Users have read/write access to other users’ check outs and read/write access to the repository.

To implement this model, leave both `VSFILEPERMS` and `VSDIRPERMS` undefined (or set `VSFILEPERMS` to 664 and `VSDIRPERMS` to 775). This sharing model is the default for VersionSync.

**Performing Design Management Operations**

As explained in “How VersionSync Works with GDM” on page 7, VersionSync is a GDM wrap. You can therefore use any GDM-based application to perform your design management operations, including

- Library Manager (see “Using VersionSync from Library Manager” on page 17)
- Project Manager (see “Using VersionSync from Project Manager” on page 18)
- GDM shell commands (see “Using VersionSync from GDM Shell Commands” on page 18)

**Using VersionSync from Library Manager**

**Note:** Library Manager is described in *Library Manager Help* from the Cadence online documentation system.

When used from Library Manager, VersionSync can run transparently—you do not need to do anything other than ensure Library Manager recognizes VersionSync as your design management system (see “Specifying VersionSync as Your Design Management System” on page 13). VersionSync locates and, when necessary, creates repositories for your designs automatically. You can, however, optionally control the location and creation of repositories. See “Creating a VersionSync Workarea” on page 13 for information on how VersionSync sets up workareas.

**Important**

VersionSync supports most Library Manager capabilities as documented in *Library Manager Help*. However, review the following restrictions and exceptions before performing design management operations from Library Manager.

- VersionSync does not support `Submit`. 
When using **Delete** with the **Delete Local And Inactivate From DM System** option, VersionSync deletes the object’s vault (as opposed to inactivating it). You cannot recover the versioned data once you have deleted the vault. If you do not want to delete the versioned data, select the **Delete Local Only** option.

When using **Rename**, Library Manager does not preserve the version history, and the renamed object is unmanaged.

When using **Rollback**, VersionSync changes the default version but does not update your workarea. Use **Update** after **Rollback** to update your workarea.

When using **Rollback**, the **Version Rollback** confirmation box indicates that checking out a non-latest version causes a branch to be created. VersionSync does not support branching. When you check out, modify, and check in a non-latest version with VersionSync, the new version becomes the latest version on the main branch. For example, a cellview has versions 1.1 and 1.2. You select 1.1 as the default version. You then check out version 1.1, modify it, and check it in. The new version is 1.3. Note that the derivation history is maintained; Library Manager’s **Version Info** shows that version 1.3 is derived from version 1.1.

Unless you have a design object checked out, VersionSync does not track what versions of design objects are in your workarea. **Version Info** will not indicate which versions are in your workarea. Therefore, use **Update** any time you are uncertain about the contents of your workarea.

For details on the behavior of VersionSync operations when invoked from Library Manager, see the descriptions for the corresponding GDM shell commands in “Using VersionSync from GDM Shell Commands” on page 18.

### Using VersionSync from Project Manager

VersionSync is currently not qualified for use with Project Manager.

### Using VersionSync from GDM Shell Commands

**Note:** GDM shell commands are described in the *Cadence Application Infrastructure User Guide*.

You typically perform design management (DM) operations through tools such as Library Manager and Project Manager. These tools invoke GDM, which in turn calls VersionSync commands. However, GDM also provides shell commands that you can invoke directly from your UNIX shell.
GDM Shell Commands Supported by VersionSync

VersionSync supports the GDM shell commands listed in Table 2-1. VersionSync supports all of the GDM command options, and in some cases additional VersionSync-specific options that you specify using the GDM \(-xtra\) option (see “The -xtra Argument” on page 21 for details).

Table 2-1  GDM Shell Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>-xtra Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>gdmcancel</td>
<td>Cancels a previous check-out.</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td><strong>Note:</strong> Any modifications remain in your workarea but files become read only. Subsequent check-outs or updates will overwrite your local modifications.</td>
<td>None.</td>
</tr>
<tr>
<td>gdmci</td>
<td>Checks in unmanaged files or creates new versions of managed files. After the check in, files are read only.</td>
<td>-force: Creates a new version of a file even if the file has not changed. -kkv, -kkvl, -kk, -ko, -kb, -kv: Controls the expansion of RCS keywords (see “Using RCS Keywords” on page 22 for details).</td>
</tr>
<tr>
<td>gdmco</td>
<td>Checks out files from the repository for editing in your workarea. After the check out, files are read/write.</td>
<td>-force: Allows the check out to overwrite write-enabled (possibly modified) files. Use with caution. -name name: Lets you specify a name (tag) to identify the version to be checked out. -kkv, -kkvl, -kk, -ko, -kb: Control the expansion of RCS keywords (see “Using RCS Keywords” on page 22 for details).</td>
</tr>
</tbody>
</table>
### VersionSync Guide

#### Using VersionSync

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdmdelete</td>
<td>Removes workarea files and optionally deletes the corresponding vaults.</td>
<td>None.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> VersionSync deletes vaults as opposed to inactivating them. Once deleted, the versioned data is lost.</td>
<td></td>
</tr>
<tr>
<td>gdmexport</td>
<td>Exports files from the repository to the specified destination.</td>
<td>-kkv, -kkvl, -kk, -ko, -kb, -kv: Control the expansion of RCS keywords (see &quot;Using RCS Keywords&quot; on page 22 for details).</td>
</tr>
<tr>
<td>gdmhistory</td>
<td>Provides a version history of a file.</td>
<td>None.</td>
</tr>
<tr>
<td>gdmimport</td>
<td>Copies a Cadence library to the specified location and checks in the library.</td>
<td>None.</td>
</tr>
<tr>
<td>gdmsetdefver</td>
<td>Sets the default version for one or more files.</td>
<td>None.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> VersionSync does not support the -name option. You cannot use a name (tag) to specify the default version.</td>
<td></td>
</tr>
<tr>
<td>gdmsetname</td>
<td>Associates a name (tag) with the specified workarea files.</td>
<td>None.</td>
</tr>
</tbody>
</table>
GDM Commands Not Supported by VersionSync

VersionSync does not support the gdmsubmit command.

The -xtra Argument

GDM provides a -xtra option for all GDM shell commands as a means of passing additional options to the underlying design management (DM) system commands. The -xtra option allows access to capabilities of the DM system that are not natively supported by GDM. Note that Library Manager implements the -xtra option through the Use Options field.

VersionSync supports GDM’s -xtra option (and the Library Manager Use Options field). For example, the VersionSync check-out command accepts a -force option, which allows a check-out to overwrite local modifications of a workarea file. You can specify the -force behavior as follows:

```
  gdmco -xtra -force filename
```

You must include the hyphen (-) when specifying -xtra arguments that are VersionSync options. For example, you must specify -xtra -force, not -xtra force. VersionSync interprets a -xtra argument without an initial hyphen as a filename on which to operate.

If the -xtra argument contains whitespace, you must surround the argument with quotation marks. For example, you must specify -xtra "-name golden", not -xtra -name golden.
VersionSync Guide
Using VersionSync

Table 2-1 on page 19 lists the GDM commands supported by VersionSync, including supported -xtra values.

Using RCS Keywords

VersionSync supports RCS keywords, such as $Id$, $Header$, and $Author$. VersionSync supports the following options to control keyword expansion: -kkv, -kkvl, -kk, -ko, -kb, and -kv. See the UNIX manpage for the RCS co command for details on keywords, including a list of the available keywords and descriptions of the -kxxx options.

Setting the Default Keyword Expansion Mode

VersionSync sets the default keyword expansion mode for a file at initial check-in. VersionSync then applies this default mode to subsequent check-out, update, and export operations. By default, VersionSync's default keyword expansion mode is -kb, which disables keyword expansion (effectively treating all files as binaries). To set a different default mode, specify a keyword expansion option (-kkv, -kkvl, -kk, -ko, -kb, -kv) during the initial check-in of a file, either as a command-line option (as specified using the -xtra option to the gdmci command or using the Use Options field from Library Manager) or through the VSINIT environment variable.

Note: Once a file is initially checked in, VersionSync cannot change the default keyword expansion mode. VersionSync has no equivalent to the RCS command rcs -kxxx.

Specifying the Keyword Expansion Mode for VersionSync Operations

VersionSync applies the default keyword expansion mode (as determined during the initial check-in of a file) to check-out, update, and export operations. You can override this default on a per-operation basis by specifying a keyword expansion option (-kkv, -kkvl, -kk, -ko, -kb, -kv), either as a command-line option (as specified using the -xtra option to GDM commands or using the Use Options field from Library Manager) or through the VSINIT environment variable.

Note: The -kv option generates keyword values but removes the keywords themselves, thereby preventing future keyword expansions. Because of the potential danger of losing keywords, VersionSync's check-out operation does not support the -kv option. If you explicitly specify -kv to a check-out operation, VersionSync errors. If VSINIT is defined as -kv, VersionSync silently ignores the option.
Using VSINIT to Pass Options to VersionSync Operations

When defined, VersionSync uses the VSINIT environment variable as a list of command-line options to be applied to VersionSync operations (as specified using the -xtra option to GDM commands, or using the Use Options field from Library Manager).

If you typically use a keyword expansion mode other than the default of -kb, you can define VSINIT to eliminate the need to specify a keyword expansion option each time you initially check in a file. For example, you can define VSINIT as -kkv (expand and keep keywords) if you typically work with ASCII files that contain keywords.

You can override VSINIT by explicitly specifying keyword expansion options on a per-operation basis.

**Note:** VersionSync currently supports only keyword expansion options in the VSINIT variable.

Example of Using VersionSync

The following sample session demonstrates typical VersionSync operations.

1. Verify that VersionSync is properly configured.
   
   `% vsinfo
   
   VersionSync Information
   
   Version:  1.0.2
   Workarea:  Unknown
   Repository:  Unknown
   
   See “Configuring Your Environment” on page 12 for more information.

2. Create the workarea.
   
   Create a workarea with the vssetvault command. If the directories you specify do not exist, VersionSync creates them for you. The files cdsinfo.tag and .VSVAULT are also created.

   For example, if you type
   
   `% vssetvault /net/myhost/projects/Repository ~/Workarea
   
   a workarea named Workarea is created and is associated with the repository directory /net/myhost/projects/Repository.

   See “Creating a VersionSync Workarea” on page 13 for more information.
3. Check repository directory permissions and ownership.

   If you created the repository as opposed to referencing an existing repository, check that the directory access permissions are correct for your needs. The directory access permissions must be enabled for read, write, and examine permissions (rwX) for yourself, and if desired, group or world. Also, make sure the owner and UNIX group for the directory are set appropriately.

   For example, if you set the permissions as follows:

```bash
% ls -ld /net/myhost/projects/Repository
```

```
drwxrwx--- 3 dmadmin dmgroup 106 Feb 29 12:01 /net/myhost/projects/Repository
```

   the owner dmadmin and anyone in the group dmgroup will be able to check files in and out of the repository.

   Also, define the VSFILEPERMS and VSDIRPERMS environment variables if you want new files and directories created by VersionSync to have permissions other than the defaults (664 for files, 775 for directories). See “Sharing Workareas and Repositories” on page 15 for more information.

4. Change your working directory to the workarea directory.

   % cd ~/Workarea

   You should now see the files cdsinfo.tag and .VSVault in your workarea:

   % ls

   .VSVault      cdsinfo.tag

5. Create some design data in your workarea.

   You can create data in variety of ways, such as copying existing files, using the Cadence design framework II Library Manager to create new data, or converting 4.3X libraries to 4.4 libraries. For example, use Library Manager to create the library TestLib, cell and2, and cellview symbol.

6. Check in your design data.

   Choose the symbol cellview from Library Manager, and choose Design Manager – Check In to check in the cellview. Version 1.1 of each of the files comprising the symbol cell view are checked into the repository.

   You could also use the GDM shell command equivalent:

   % gdmci -initial -lib TestLib.and2:symbol

   **Note:** If you use RCS keywords, read “Using RCS Keywords” on page 22 before checking in your design files.
7. Set a symbolic name for the base-line release.

To name (also known as tag) the cell for future reference, use the `gdmsetname` command. (Library Manager does not support defining symbolic names). Naming allows future check-out or update operations by name. Set the name `Beta1` to version 1.1 of the cell view `symbol`.

```
gdmsetname -version 1.1 -name Beta1 -lib TestLib.and2:symbol
```

8. Create a new workarea to use as a test workarea.

```
vssetvault /net/myhost/projects/Repository ~/TestWorkarea
```

The new workarea uses the repository directory you created earlier.

9. Change your working directory to the test workarea directory.

```
cd ~/TestWorkarea
```

10. Run the `gdmupdate` command to update the test workarea with the version `Beta1` of the cell view `symbol`.

```
gdmupdate -name Beta1 -lib TestLib.and2:symbol
```

You can now verify that the library directory `TestLib` was created and within it, the cell directory `and2` and the cellview directory `symbol`, containing the files you created with Library Manager earlier.

The cellview `symbol` is ready for check-out and modification.
System Administration

This chapter contains the following sections:

- “Overview” on page 27
- “Controlling Access to a Repository” on page 27
- “Reorganizing a Repository” on page 27
- “Understanding Performance Issues” on page 28
- “Providing Adequate Disk Space” on page 30
- “Backing Up Data” on page 31
Overview

This chapter describes concepts and tasks of interest to someone responsible for setting up and maintaining a VersionSync™ repository.

Controlling Access to a Repository

You can take several measures to help prevent unintended or unauthorized reading or writing of repository vaults. The most restrictive security measures available are those provided by the system and are instituted as file, directory, and host access permissions on the machine hosting the repository. You can restrict access to the repository to a single user or groups of users with directory permissions and to specific hosts with network file system export mechanisms. These measures should be sufficient for most installations.

See “Sharing Workareas and Repositories” on page 5 for more information on repository security.

Reorganizing a Repository

Generic design management (GDM) and VersionSync do not support renaming or moving vaults or vault directories within a repository. There is a Rename command available from Library Manager, but it does not preserve a file's version history information. Although not recommended, you can reorganize a repository manually.

While renaming or moving a vault file or directory, you must consider a few limitations. The directory hierarchy for a workarea is identical to that of the repository. Thus, any change in the repository must be reflected in each existing workarea for the workarea to continue to function properly. VersionSync does not fix inconsistencies caused by repository changes. Follow the guidelines described in this section to minimize errors when you rename vaults or vault directories.

Renaming Vaults

Renaming a vault, .vs file, as long as it continues to end with the required .vs suffix. Follow these guidelines:

- Do not rename a vault that has a locked version. If you do, you will not be able to check in the checked-out workarea file until its name has been changed as well. Therefore, before changing the name of a vault, users should cancel all outstanding locks.
Do not rename files that are part of a co-managed set, as doing so may destroy the integrity of a co-managed set. For example, the files master.tag, pc.db, sch.cdb, and prop.xx are the four files of a co-managed set—they comprise a view. Renaming pc.db would destroy this view, because pc.db would no longer be available by the required name. Therefore, rename only those files that are not part of a co-managed set.

Renaming Vault Directories

It is usually safe to change the name of a vault directory when it does not represent a library, cell, or cellview and when it is not referenced by any other data. However, be careful when the directory is a library, cell, or cellview.

Follow these guidelines:

- If you rename a library, cell, or cellview, ensure that all library data that references the old name is also renamed.
- If you rename a library directory, update Cadence application infrastructure files—cds.lib, project.lib, and cdsinfo.tag. For information about these files, see the Cadence Application Infrastructure User Guide.

In addition, the guidelines for renaming vaults also apply to renaming vault directories.

Understanding Performance Issues

Several factors affect the performance of VersionSync. Host platform, network file system performance, and RAID configuration are external factors that impact performance. In addition, the performance of individual VersionSync operations is based on several factors.

Factors That Affect VersionSync Operation Performance

The following factors characterize the performance of VersionSync operations:

- The command being performed
  
  All VersionSync commands can be broken into one or more smaller operations: copy, difference analysis, and metadata update.

  **Copy** moves a file from a workarea to the repository or from the repository to the workarea. The check-in, check-out, update, and export commands perform a copy.

  **Difference analysis** compares the working version of a file with the latest version in the vault. Only the check-in command requires a difference analysis.
**Metadata update** updates the metadata used to manage the files in the repository. Most commands, including the cancel, delete, set name, and set default version commands perform metadata updates. Metadata updates impact performance far less than do copy and difference analysis.

- The number of files affected by the operation
  Performance is affected by the number of files that are part of a library, cell, or view
- The size of the files
- The file type (ASCII or binary)
  Performance is much better with ASCII files than with files containing binary data. Difference analysis on files containing binary data is not as efficient as with ASCII files. Also, some characters in the binary data must be encoded before being deposited into the vault’s data section. Therefore, vaults for binary files are larger than those for ASCII files, affecting performance considerably. This performance degradation is noticeable when the size of a file is on the order of 5 megabytes or larger.
- The file system configuration (NFS or local disk). NFS is much slower than a local disk, but NFS may be required for shared workareas or repositories.

**Performance of VersionSync Operations**

This section discusses the performance of the VersionSync operations that typically have the greatest performance overhead.

**Update**

The update operation does a copy. Because VersionSync does not store metadata in workareas about the state of the files in the repository, update is the most expensive operation. Update requires the file server to examine every file in the repository to determine its status. Because it would be prohibitively expensive to compare the working files to those in the repository, an update operation copies each file specified by the request. Thus, the performance is the same when every file requested needs to be updated as when only one of the files needs to be updated.

Update requests are typically on a library, cell, or view, and the time the request takes increases based on the number of files specified by the request.
Check In

The check-in operation does a copy and a difference analysis. The performance of check-in operations depends directly on the size of the file and the nature of the differences between the working file and the latest version in the vault. As the difference between the two versions increases, the time to complete the difference analysis increases. The most time-consuming case is where the file's contents have been completely changed. The performance of a check-in with few changes is comparable to that of check-out operations.

A check-in operation on previously unmanaged data only does a copy because there are no previous versions against which to do a difference analysis. Therefore, the check-in time for previously unmanaged data is the same as that for update operations for the same number of files, and the time taken to perform the operation increases with the number of files in a request. For example, the time to check in 50 new files is the same as the time to update the same 50 files.

Check Out

The check-out operation does a copy. The performance characteristics are identical to that of a check-in operation for previously managed data, where there are slight differences between file versions.

Export

The export operation does a copy and has the same performance characteristics as check-out.

Providing Adequate Disk Space

Repositories grow over time as more vaults and more versions within vaults are created. You should locate repositories where there will be sufficient disk space to support their growth. The type of files that are stored in the repository also affects the rate at which the repository grows. VersionSync stores versions using reverse deltas so that only the changes between versions are stored in the vault. However, when the difference between versions is great (as is common for binary data), a new version can use disk space equivalent to a full copy of the file.

For private workareas, each user has copies of the managed files in his workarea. Shared workareas can be employed to reduce disk space requirements. Unlike the repository, a workarea has only one version of a file at a time, so disk usage does not grow significantly over time unless the number of managed files in the workarea increases.
Back Up Data

A VersionSync repository is a directory hierarchy. Backing up the repository directory tree from its root is sufficient to capture all managed files. No user work files or unimportant files are located in the repository.

Within a given workarea, there may be more than one repository. All .VSVAULT directories and all other repository directories should be backed up. To simplify the backup requirements, consider using the vssetvault command to set up a workarea with a single repository (see “Specifying the Repository Location” on page 5).

When establishing backup strategies, remember that the repository is “live” and may be updated at any time by users. A good backup policy should ensure that the vault is not being used while it is being backed up.

Each user’s workarea data should be backed up according to the methodologies and practices you use for other volatile user data. Files that have write access in a user’s workarea are either unmanaged or checked out and possibly modified. Your backup policy should take into account the risk associated with having to recreate the data if it is lost.

It may be the case that read-only files in a user’s workarea can be excluded from backups, because they can generally be derived from the VersionSync repository with the gdmupdate command. However, if you have read-only volatile data, you may want to back it up. User practices and policies should dictate which files should be backed up by default.
Most VersionSync™ capabilities are accessed through generic design management (GDM) and are not called directly by a user. This chapter documents the VersionSync commands that are callable directly by a user:

- “vsinfo” on page 33
- “vssetvault” on page 35
vsinfo

vsinfo [-help] [-usage]

Description

Displays the following information:

- VersionSync version number.
- Workarea root directory and its associated repository directory. A VersionSync workarea root directory contains a .VSVAULT file or directory. If the directory from which vsinfo was invoked is not part of a workarea, the workarea and repository are listed as Unknown.

Arguments

- -help Displays the vsinfo help page.
- -usage Displays the vsinfo syntax.

Examples

This example shows vsinfo invoked from a user's home directory, which is not a workarea:

% vsinfo

    VersionSync Information

    Version: 1.0.2
    Workarea: Unknown
    Repository: Unknown

This example shows vsinfo invoked from a user's workarea:

% vsinfo

    VersionSync Information

    Version: 1.0.2
    Workarea: /usr1/goss/asic1Workarea
    Repository: /net/apollo/usr1/repositories/asic1
This example shows `vsinfo` invoked from a shared workarea where the repository is located in the workarea:

```bash
% vsinfo

VersionSync Information

Version: 1.0.2
Workarea: /usr1/projects/asic2
Repository: /usr1/projects/asic2/.VSVAULT (in workarea)
**vsetvault**

vsetvault [-help] [-usage] repositorydir workareadir

**Description**

Associates a workarea directory with a repository directory. The `vsetvault` command

- Creates the `repositorydir` and `workareadir` directories if they do not already exist, UNIX file permissions permitting.
- If the workarea already contains a `cdsinfo.tag` file, appends a `DMTYPE vs` line to the file. Otherwise, creates a `cdsinfo.tag` file with a `DMTYPE vs` line.
- If `repositorydir` and `workareadir` are the same directory, creates a `.VSVAULT` directory in the workarea directory. Otherwise, creates a `.VSVAULT` file that contains the absolute path of the specified repository directory.

**Note:** It is not recommended to define a repository at the cell or cellview level. Define the repository at the library level or above.

You cannot use `vsetvault` on an existing VersionSync workarea (a directory containing a `.VSVAULT` file or directory). If attempted, `vsetvault` returns an error. This restriction safeguards against unintentionally changing the repository that is associated with a workarea, which in the case of `.VSVAULT` being the repository directory itself, could result in lost data. If you need to reassign a workarea’s repository, you must delete or rename the existing `.VSVAULT` before calling `vsetvault`.

**Arguments**

- **repositorydir**
  Specifies the repository directory to be associated with the workarea. If `repositorydir` does not already exist, it is created. You can specify an absolute or relative path. If you want other users on your local area network (LAN) to be able to access the repository, specify a path that is accessible to them.

- **workareadir**
  Specifies the directory that you are setting up as a workarea. If `workareadir` does not already exist, it is created. You can specify an absolute or relative path.

- **-help**
  Displays the `vsetvault` help page.

- **-usage**
  Displays the `vsetvault` syntax.
Examples

This example defines a workarea with a repository located on a different host (~/
asic1Workarea/.VSVAULT is a file that points to the repository directory):

    % vssetvault /net/apollo/usr1/repositories/asic1 ~/asic1Workarea

This example defines a workarea with a repository located in the workarea itself (/usr1/
projects/asic2/.VSVAULT is the repository directory):

    % vssetvault /usr1/projects/asic2 /usr1/projects/asic2