SERENA®

Dimensions® CM 14.2.0.2

Migrating to Dimensions CM

Serena Proprietary and Confidential Information

Publication date: March 2016

Table of Contents

Welcome to Serena Dimensions CM

Serena® Dimensions® CM is a powerful process management and change control system. Dimensions CM helps you organize, manage, and protect your software development projects on every level—from storing and tracking changes to individual files, to managing and monitoring an entire development cycle.

Before you Begin

The Dimensions CM readme contains the following information:

- What's new
- Fixed issues
- Software compatibility requirements
- Installation notes
- Known issues

The readme is available online at:

http://help.serena.com/doc_center/doc_center.html#dcmDoc

Contacting Serena Technical Support

Serena provides technical support for all registered users of this product, including limited installation support for the first 30 days. If you need support after that time, contact Serena Support at the following web site and follow the instructions:

http://www.serena.com/support

Language-specific technical support is available during local business hours. For all other hours, technical support is provided in English.

You can use the Serena Support web page to:

- Report problems and ask questions.
- Obtain up-to-date technical support information, including that shared by our customers via the web, automatic e-mail notification, newsgroups, and regional user groups.
- Access a knowledge base, which contains how-to information and allows you to search on keywords for technical bulletins.
- Download updates and fix releases for your Serena products.

Videos

Videos of Dimensions CM features can be viewed online at:

http://help.serena.com/doc_center/doc_center.html#dcmVid

License and Copyright Information for Third-Party Software

License and copyright information for third-party software included in this release can be found as part of the software download available at:

http://www.serena.com/support

Part 1

Migrating from PVCS Version Manager

Chapter 1

About the PVCS Version Manager Migration

Introduction

You can use the extended migration tools in Dimensions CM to migrate Version Manager objects to Dimensions.

The Migration Console is a graphical front end to a set of XML-based utilities that move data from supported sources into Dimensions products, allowing you to define how the data is mapped into Dimensions.

Using the Migration Console, you create, design, and execute one or more migrations. The Migration Console facilitates migrations from all PVCS Version Manager platforms to all Dimensions CM platforms. For example, a user running the Migration Console (on Windows) can migrate data from a Version Manager File Server on Solaris to a Dimensions CM server on AIX.

The Migration Console makes the following features available:

- You can migrate all, part, or none of the Version Manager project structure into a design part hierarchy and/or into a Dimensions project hierarchy.
- Migration preserves your Version Manager project structure.
- You can exclude any subset of Version Manager projects.
- You can target any subset of Version Manager files to specific Dimensions design parts, projects or streams, and item types.
- You can migrate Version Manager labels and promotion groups to Dimensions multi-valued attributes, projects or streams, and baselines.

Migration automatically creates process model data in Dimensions, including products, users, groups, privileges, item types, design parts, projects, baselines, and relationships.

Migration Preparation Overview

Version Manager and Dimensions are conceptually similar in many ways, however they are quite different in many respects as well. Features that you are accustomed to using on a daily basis in Version Manager may not map directly to a comparable feature in Dimensions. You should plan for a number of technical and organizational challenges, including:

- Dimensions is a sophisticated, feature-rich product. This guide and other materials from Serena can help you begin to understand how it compares to Version Manager and where you might start with your implementation, however take the time to read the Dimensions documentation as well and, if necessary, participate in product training.
- The owners of the migration process, perhaps your IT or rollout administrators, must decide what Dimensions features to use and how. This guide can help with this, by explaining how many commonly used Version Manager features compare to features in Dimensions.
- Furthermore, the administrators must decide how to migrate key data from Version Manager to Dimensions. This guide can help with this as well, by suggesting migration paths based on different scenarios.

- Migration of large data sets takes time and patience. Planning, executing, and validating a large-scale migration may be an iterative process, with some undesirable results along the way. Again, this guide can help by helping you make the right decisions about your migration strategy.
- Finally, once migration is complete, Serena highly recommends that your users be trained on the Dimensions clients that they will use. Within Serena Software, we deliver in depth internal training to all new users with a specific focus on just those clients that different types of user are interested in.

What Interfaces Can I Use with Dimensions?

With PVCS Version Manager, you are accustomed to working with a broad array of clients, including the desktop GUI, the Web client, the command-line client, the Developer's Toolkit, and a wide selection of IDE integrations. Dimensions is no different, and has also been designed and built to support application developers in whatever environment they are most comfortable with, and with many options for customization and extension.

Dimensions interfaces and clients include:

- A fully featured end user desktop client, with rich file and project management features, as well as powerful collaborate tools for larger teams.
- A Windows Explorer client, from which you can upload and download files, as well as refactor local changes.
- Many IDE clients, including Visual Studio and Eclipse IDEs. Most developers can complete all of their Dimensions tasks without having to leave their favorite environment.
- A Web client that extends core version functionality to distributed and casual users who do not need a heavier local client.
- A complete command-line interface that exposes all core product functionality.
- A variety of APIs, including Web services and Java APIs.

Using the Sample Migration

When you first open the Migration Console, you see one predefined migration in the migrations list. This predefined migration comes with the product and defines a migration of the VM Sample Project database into Dimensions CM using migration defaults. You can examine, edit, and execute the predefined migration to get an idea of how to use the Migration Console. (Be sure to modify the server information to match your environment.)

IMPORTANT! Using the sample migration does not take the place of prototyping and testing the actual migration using your actual data! Please see "Preparing to Migrate from Version Manager" on page 33.

Chapter 2

Planning Migration from Version Manager

What Version Manager Features are Migrated?

This topic summarizes the types of Version Manager data that are and are not migrated. To better understand how features in Version Manager that you are accustomed to using compare to similar features in Dimensions, continue on to "Comparing Features" on page 19.

What Is Migrated

The Migration Console supports migration of the following data from PVCS Version Manager to Dimensions CM:

- Users are migrated and merged from all source Version Manager projects into one user list in Dimensions.
- Version Manager groups are migrated to Dimensions roles. By default, roles in Dimensions have the same names as the source Version Manager groups.
- You can migrate all projects and files from a particular Version Manager project database, or you can migrate specific projects and subprojects. You can also migrate empty folders from Version Manager.
- Version Manager revisions are migrated to item revisions.
- You can migrate Version Manager archives intact to a Dimensions delta-library (Archive mode), or you can migrate the individual revisions one at a time to a non-delta item library (File mode).

  Though migration in Archive mode is faster, you will normally want to use File mode, because it does not require the Dimensions delta engine. If the project database being migrated is managed by a Version Manager file server, you must either use File mode or make sure that the client path specified by the path map in your Version Manager installation is the path to an actual directory rather than a secure path.
- Version Manager labels and promotion groups migrate to multi-value attributes, projects or streams, and baselines in any combination that you specify. By default, labels move to the VCS_TAGS multi-value attribute.

What Is Not Migrated

The Migration Console does not support migration of certain types of data:

- Shared archives are not migrated. Version Manager allows you to reference a common archive from multiple locations, for example if you want to share a readme.txt file across multiple projects. Dimensions does not support archive sharing. As a result, any versioned files in Version Manager that share an archive will be imported into separate files in Dimensions. For example, if a shared archive is referenced from two locations in Version Manager, two files will be created in Dimensions. Those files will have no relationship to each other, and must be managed and versioned independently of one another.
- Version Manager administrative data including but not limited to permission / access control, event triggers, and Version Manager configuration files (vcs.cfg).
- Version Manager locks are not migrated.
- Version Manager workspaces are not migrated.

- Version Manager relationships to items in external issue tracking systems (such as Serena Business Manager issues linked via SourceBridge or TrackerLink) are not migrated.

Comparing Features

The following topics compare familiar features in Version Manager to comparable features in Dimensions. In some cases, the features are very similar. However, Version Manager and Dimensions differ in a number of important ways. By understanding how Version Manager and Dimensions compare, you can better prepare yourself and your colleagues for the new ways of working once migration is complete.

If you are responsible for choosing the migration options for your Version Manager projects, review "Migration Recommendations" on page 29.

Project Databases

What are project databases in Version Manager?

Version Manager organizes projects in project databases, which are hierarchical collections of projects and sub-projects. Project databases reside on the file system.

How does this compare to Dimensions?

Dimensions stores all data in a back-end database using a supported database platform such as Oracle or SQL Server. Dimensions organizes projects and streams into entities called products. A product is the top-level organizer for development groups using Dimensions. A product might correspond to a specific product line or business unit. For example, the commercial software part of a business might have its own Dimensions product.

All work within a product is further organized into design parts. Design parts represent one large structural component of the product. For example, separate design parts might be created for product client code, user documentation, installers, and customer support applications.

Migration Recommendations

You can migrate any combination of projects, subprojects, and files to any Dimensions product, design part, project, or stream. The Migration Console is designed to provide you with as much flexibility as possible.

If your goal is to maintain your current working practices as much as possible and minimize end-user confusion, consider a one-to-one migration of your project databases to Dimensions products. Conceptually, project databases and products are comparable.

For More Information

To learn more about products, design parts, projects, and streams in Dimensions, see *Getting Started with Dimensions CM*. For details on product, part, and project configuration, see the *Process Configuration Guide*.

Projects

What are projects in Version Manager?

Version Manager organizes all work into a hierarchical collection of projects and sub-projects. Projects logically group related work based on such criteria as product lines or modules, or different releases.

How does this compare to Dimensions?

Within Dimensions, you can also organize work for different modules and releases into projects. Dimensions projects store all files into a hierarchical folder structure that may correspond to the physical organization of files on a disk.

For detailed information on file and project organization in Dimensions, see *Getting Started with Dimensions CM*.

Migration Recommendations

You can migrate any combination of projects, subprojects, and files to any Dimensions product, design part, project, and stream. The Migration Console is designed to provide you with as much flexibility as possible.

If your goal is to maintain your current working practices as much as possible and minimize end-user confusion, consider a one-to-one migration of your Version Manager projects to Dimensions projects. Conceptually, projects are comparable in Version Manager and Dimensions.

If you want to start with new projects or streams in Dimensions that better reflect how your organization plans and executes work, you may also choose to migrate specific version labels or promotion groups to Dimensions projects. See "Migrating Version Labels" on page 29 and "Migrating Promotion Groups" on page 30.

For More Information

To learn more about Dimensions projects and streams, see *Getting Started with Dimensions*.

File Checkout / Check-In

How do you check files in / out in Version Manager?

In Version Manager, when you check out a file you also typically place a lock on that file. You also have the option of getting files without locking them, as well as locking and unlocking them without checking them in or out.

How does this compare to Dimensions?

Dimensions checkout and check-in features are essentially equivalent, in that you can check a file out, make changes, and check it back in. However, there are some key differences:

- Depending on how Dimensions is configured, you may be able to work optimistically. This means that you can get files, work on them, and check them in without having to first check them out and lock them.
- Dimensions provides a rich synchronization tool that allows you to quickly compare the contents of your work area with the contents of the repository and choose how to resolve differences all at once.
- You cannot lock or unlock a file independently of checkout / check-in.

For More Information

To learn more about file versioning, check out, and check in, see *Getting Started with Dimensions*.

Getting All Files

How do you get all files in Version Manager?

Version Manager allows you to get all files under a project at once, simply by selecting the project and choosing to get all subprojects. You can also use the PCLI to get all files needed for a build.

How does this compare to Dimensions?

You can do the same in Dimensions, either by getting the files or using the download or update features. For example, in the desktop client, you can right click a project or stream and choose to download or update the project.

Dimensions also provides a command-line interface that you can use to script gets of all files needed for a build.

For More Information

For more information on synchronizing and updating projects and streams, see *Getting Started with Dimensions*.

To learn more about the Dimensions command-line interface, please see the *Command-Line Reference*.

Version Labels

What are version labels?

In Version Manager, you can apply either a fixed or floating label to revisions of files that identify those revisions as belonging to particular milestones, builds, or other criteria as defined by your organization.

How do version labels compare to features in Dimensions?

Dimensions does not support version labels. However, it does provide comparable features that allow you to accomplish many of the same goals:

- **Baselines**. A baseline is a collection of specific revisions of specific files. Baselines allow you to recreate builds and recover content associated with key milestones. In this way, baselines are comparable to the use of version labels to tag content belonging to a specific build or milestone, a common usage scenario in Version Manager.
- **Custom attributes**. Custom attributes are data fields that can store any sort of metadata about files and revisions, just as you can use version labels in Version Manager to store text information about particular revisions. You can configure attribute values to automatically apply to new revisions of a file, much as you would do with a floating version label. You can also configure attribute values to apply to specific revisions only.

Migration Options

You can migrate labels to baselines, attributes, or projects. See "Migrating Version Labels" on page 29 for recommendations for your scenario.

For More Information

To learn more about Dimensions baselines, attributes, and requests, please see the *Dimensions User's Guide*.

Promotion Groups

What are promotion groups?

Promotion groups in Version Manager represent milestones in a development lifecycle as defined by a promotion model. Typical examples include Development, Quality Assurance, and Production. You can use promotion groups to find and apply actions to collections of revisions of files, much as you can with version labels. In this way, you can use promotion group assignments to gather revisions into a build, such as builds for specific project milestones.

How do promotion groups compare to Dimensions features?

Dimensions provides a number of features that provide options comparable to common use cases for promotion groups. These include:

- *Collecting all files associated with a development milestone*. If you use promotion groups in Version Manager to identify and collect files associated with a development milestone phase or build, you can achieve the same result in Dimensions CM by creating a baseline of all files that belong to a particular state in the lifecycle. Dimensions also provides a global lifecycle that you can use to progress files into a build-ready state once they have reached the appropriate level of approval. By using the global stage lifecycle in this way, you can approximate the use of promotion groups in Version Manager to identify and collect files associated with a particular milestone.

Migration Options

You can migrate promotion groups to baselines, attributes, or projects. See "Migrating Promotion Groups" on page 30 for recommendations for your scenario.

For More Information

To learn more about lifecycles and baselines, see *Getting Started with Dimensions* and the *Dimensions User's Guide*.

Event Triggers

What are event triggers?

With Version Manager, you can configure specific actions to occur automatically when a specific event occurs. For example, an event such as a file check-in will trigger an event such as starting an external program or running a script. Common uses include the triggering of notification systems, as well as running external applications to copy or deploy files.

What Dimensions features compare?

With Dimensions, you can use the events callout interface in the Developer's Toolkit (DTK) to define events that are triggered in response to specific commands. You can define validation events, events that run before the command is executed (pre-events), and events that run after the command is executed (post-events). The events include creating new Dimensions objects, checking files in or out, sending a notification, and more. For a detailed explanation of DTK events using the events callout interface, please see chapter 5 in the *Serena Dimensions Developer's Toolkit Reference*.

Can I send event notifications with Dimensions?

If you use event triggers in Version Manager to trigger email notifications of events, you may consider using notification features in Dimensions. You can configure email notifications in response to a variety of events in Dimensions, including:

- An item arrives in a user's inbox
- An attribute on an object in a user's inbox is updated
- An item is added to or removed from a project
- A user is added or removed
- A product is added or removed

Migration Options

Event triggers are not migrated to Dimensions. You can set up DTK events, notifications, and deployment triggers in Dimensions, once migration is complete.

For More Information

To learn more about the events callout interface, see the *Serena Dimensions Developer's Toolkit Reference*.

To learn how to configure email notifications and to learn more about deployment features in Dimensions CM, see the *Serena Dimensions Process Configuration Guide*.

Shared Files

What are shared files?

Version Manager allows you to store references to shared files from multiple folders and projects. For example, if there is a particular class file required by multiple applications, each project that requires it can store a reference to one shared instance of the file. In this way, you do not need to duplicate common files into all of the projects that need them.

Can I share files in Dimensions?

Dimensions does not support file sharing in this way. You must do one of the following:

- Duplicate the shared files into each of the projects that need them and take care to update all locations when the files are updated.
- Use the local item replication feature to automatically copy new revisions of the shared files into all of the projects that use them. Please see the *Dimensions System Administration Guide* for more information.

Migration Options

Shared files are not migrated to Dimensions, because Dimensions does not support file sharing.

Branches

What is branching in Version Manager?

A branch is a separate line of development consisting of one or more revisions that diverge from a revision on the trunk or on another branch. Branching lets you develop alternate variations of a file in parallel with the continued development of the revision from which it was branched. In Version Manager, a branch is created on a file by file basis, when you check in a non-tip revision, check in a revision with multiple locks, or choose to force a branch on a file. The resulting branched revision uses a branched numbering

scheme, for example if you branch off of revision 1.3, the first branch revision might be 1.3.1.0.

How does Dimensions support parallel development?

In Dimensions, you can branch revisions for files within a project, or use projects to store entire branches.

- To branch revisions for files within a project, you can enable *named branching* within the project. Named branches allow you to number new revisions of files appropriately for a branch, such as patch#1 or branch#1. This allows you to store branch revisions of files within the files' complete revision history, much as you can do with Version Manager.
- Alternatively, to branch entire projects, you create an entirely new project, or stream of development, based on an existing project or baseline. All of the files from the existing project or baseline are duplicated into the new stream, which should be named accordingly to identify it as a branch. You can optionally assign a named branch revision numbering scheme to each project as well. If necessary, you can then merge the branch back into the original project using the powerful parallel development and project merge features in Dimensions.

Migration Options

Revisions and revision numbers are migrated to Dimensions. However, if you want to create projects in Dimensions that mirror your branches in Version Manager, you may choose to migrate only files relevant to a particular branch to one project at a time in Dimensions. You may also need to refactor your projects once migration is complete.

For More Information

To learn more about branching concepts in Dimensions, see the *Getting Started with Dimensions*.

To learn about branch configuration, see the *Process Configuration Guide*.

Locking

How does locking work in Version Manager?

In Version Manager, you can lock and unlock files without checking them in or out. This allows you to prevent other users from checking new revisions in, even if you have not checked the files out.

How does this compare to Dimensions?

Dimensions does not provide this feature. Files are only locked when they are checked out. Depending on how it is configured, Dimensions also allows you to work on files and check them in without locking them. In this scenario, you can upload files without locking them first.

Migration Options

The lock status of files is not migrated. If any files are locked in Version Manager at the time of migration and you want to carry this forward, you must lock the files again after migration is complete.

Workspaces

What are workspaces?

Workspaces in Version Manager store a number of work settings for groups and individuals, including default working directory and default version.

Does Dimensions have any comparable features?

Dimensions has no workspace or default revision concept. However, you can achieve similar results with the **default work area**. You can define a default working location (*work area*) for every project and stream. Every individual can also store a unique work area for each project and stream.

Migration Options

Workspace settings are not migrated to Dimensions. You must define the work area for the project or stream after migration, and each user can specify unique work areas.

For More Information

For more information on work areas, see the *Dimensions User's Guide*.

Issue Relationships

How can you relate issues in Version Manager?

In Version Manager, using Sourcebridge, you can relate issues stored in Serena Business Manager to versioned files. This allows you to track all work in response to a defect or enhancement in Serena Business Manager.

Can you relate issues in Dimensions?

Dimensions also integrates with Serena Business Manager, via Serena Development Manager. This solution tightly integrates Dimensions CM and SBM, allowing development teams to track work requests in Dimensions that originated in SBM.

Migration Options

Issue relationships using Sourcebridge or Trackerlink are not migrated to Dimensions. You must recreate any issues as requests in Dimensions or, if you are using SBM, implement the integration to SBM.

For More Information

For more information on Dimensions issue management features, please see:

- The *Serena Dimensions CM Process Configuration Guide* for information on configuring issue management features and enable the integration to SBM.
- The *Dimensions CM User's Guide* for more information on using issues.
- The *Dimensions CM for Visual Studio User's Guide* and *Dimensions for Eclipse User's Guide* for information on using issue management features from within IDEs.

Users, Groups, and Permissions

How do you manage users in Version Manager?

Version Manager allows administrators to manually create users and groups and then assign permissions to them. Version Manager also supports external login sources such as LDAP.

How do you manage users in Dimensions?

Dimensions also allows administrators to manually create users and groups and then assign privileges to both users and groups. Dimensions also allows administrators to define and assign organization roles to users and groups.

For detailed information on Dimensions user and group management, see the *Process Configuration Guide*. For information on configuring external login sources such as LDAP, please Appendix A, "Configuring Centralized Network Authentication," of the *System Administration Guide*.

Migration Options

You can choose to migrate Version Manager users and groups to Dimensions users and roles, respectively. However you will need to re-establish privileges once migration is complete.

For More Information

To learn more about users, groups, and permissions in Dimensions please see the *Process Configuration Guide*.

Keyword Expansion

What is keyword expansion in Version Manager?

In Version Manager, you can place keyword strings in workfiles that expand to display specific information about the archive, such as the path to the archive, the check-in date, the revision number, and the like.

Does Dimensions support keyword expansion?

You can use the item header substitution feature in Dimensions to achieve functionality similar to keyword expansion. Dimensions allows you to define item format templates that describe content that should be expanded in the header for different types of files. This may include the values of certain user-defined attributes, the current lifecycle status, the item name, and the last update date.

Migration Options

Keywords that are already expanded are migrated. Keywords that are not expanded can be lost during migration, but can be restored using the header substitution feature in Dimensions. Version Manager style keywords such as %ARCHIVE% can be expanded using the header substitution feature. Paths in the variables are expanded to include forward slashes (**/**), similar to behavior in Version Manager.

For More Information

For details on defining item format templates and using item header substitution, please see Appendix A, Item and Request Templates, in the *Process Configuration Guide*.

End of Line Characters

How does Version Manager handle EOL characters?

For archives that have end-of-line translation enabled, Version Manager revision content with Windows end-of-line markers, specifically carriage return plus line feed (CRLF), and then converts the characters to just line feed when checking out to UNIX platforms.

How does Dimensions handle EOL characters?

Dimensions stores files as they are, when they are checked in. By default, end of line characters are preserved when they are checked out, meaning that files created on Windows include CRLF end-of-line characters when checked out on Windows or UNIX platforms. You can also set either CR / LF or LF as the default for all files, or set end-of-line characters to remain unchanged regardless what platform the files are extracted to.

If necessary, you can also set the end-of-line characters temporarily for your session by using the SET EOL command. This is useful if, for example, you must check files out to UNIX that were originally created and checked in on Windows, as you can change the end-of-line characters to LF during your session. This prevents any file corruption that may result from checking files out and opening them with incorrect end-of-line characters for your current system's platform.

Migration Options

End-of-line characters are migrated as they are in the source files in Version Manager. However, you may choose to change the default options for end-of-line characters once migration is complete.

IDE Integrations

Dimensions supports a broad range of IDEs with rich integrations to the most popular development tools.

Visual Studio and Eclipse

The Dimensions integrations to Visual Studio and Eclipse are generally comparable in functionality to the Version Manager integrations, with rich synchronization and file management features. There are some functional differences based on core differences between Version Manager and Dimensions. For example, baselining features are available instead of version labeling features.

As in Version Manager, you can integrate to SBM to track all work in SBM issues. Because the check out / check in model is somewhat different between Version Manager and Dimensions, the information that appears in SBM issues is different. Specifically, a new revision is created in Dimensions when you check out a file, and the new revision is related as "in response-to" the SBM issue. The notes in associated SBM issues reflect this.

For more information, please see the *Visual Studio for Dimensions CM User's Guide* or the *Eclipse for Dimensions CM User's Guide*.

Migration Options

IDE projects are migrated along with any other files. If you are working with SCC projects in Visual Studio, there are some additional steps you must complete; please see "Migrating Version Manager Visual Studio Projects" on page 51.

For More Information

Once migration is complete, you will need to open your IDE projects from source control, from within your IDE. For details on this and using the IDE integrations in general, please see:

- The *IDE User's Guide* for information on working with projects in SCC and COM integrations
- The *Eclipse for Dimensions User's Guide*
- The *Visual Studio for Dimensions User's Guide*

Reports

How does reporting work in Version Manager?

Version Manager provides a wide variety of reports, including:

- Journal reports, that contain information about actions that modify an archive
- History reports, that contain information about versioned file history and properties
- Security reports, that report on security settings in the ACDB
- And difference reports, that illustrate the differences between different versions and instances of files.

How do reports in Dimensions compare?

Dimensions provides very rich reporting capabilities that in some ways surpass what is available in Version Manager. These include the following:

- You can use the Reports Builder in the Dimensions desktop client to generate list, table, and graphical pie and bar charts on any type of object in the Dimensions repository (such as file items). You can report on data such as title, ID, type, status, update date, and more. A broad number of listing and graphical reports are available. You can use the PVCS Merge Tool to compare files and versions and generate difference reports.
- Standard reports on a broad variety of Dimensions metadata that you can generate in ASCII text from the command-line interface.
- Published database views that you can use to build your own applications to extract database from the Dimensions database.

Migration Options

Reports are not migrated to Dimensions.

For More Information

- To learn more about the Reports Builder, see the *Dimensions User's Guide*.
- To learn more about the command-line interface, see the *Command-Line Reference*.
- To learn more about the published views, see the *Dimensions Reports Guide*.

Login

How do you log in to Version Manager?

To log in to a Version Manager project database, you must browse or supply the path to the project database and provide your login credentials.

How does this compare to Dimensions?

Dimensions runs on a database platform such as Oracle or SQL Server, and therefore requires you to supply database connection information when logging in. For example, for Oracle, you must supply the system identifier (SID) as well as the host and database names. The administrator responsible for Dimensions installation must note this information and supply it to you, so that you can then provide the information when configuring migration and logging in to Dimensions.

Archives

What are archives in Version Manager?

In Version Manager, revision history is stored in an archive. By default, Version Manager stores revisions of text files as a series of deltas, or changes. Only the latest revision is stored in its entirety. This method conserves disk space.

Does Dimensions support Archives?

With Dimensions, you can choose to store every revision as an entire file in the Dimensions item library, which is the location where file revisions are stored, or you can store the revision in archives using Version Manager archive technology. In Dimensions, archive storage is called delta storage. Serena does not recommend using delta storage, as doing so doing so can negatively affect day-to-day performance after migration, as well as cause problems with end-of-line (EOL) characters.

Migration Options

When choosing the export options, you can choose whether to migrate the archives intact or migrate the archives to non-delta item libraries. Preserving the archives may improve migration performance, however Serena recommends using non-delta item libraries for reasons noted above.

For More Information

To learn more about item libraries and delta storage in Dimensions, see the *Process Configuration Guide*.

Migration Recommendations

Consulting the following topics for recommendations on migrating different types of Version Manager objects.

- "Migrating Version Labels" on page 29
- "Migrating Promotion Groups" on page 30
- "Migrating Branches" on page 31
- "Migrating Users" on page 32
- "Migrating Archives" on page 32
- "Migrating IDE Projects" on page 32

Migrating Version Labels

See "Version Labels" on page 21 for more information on how version labels compare to features in Dimensions.

You can migrate version labels to any of the following. You can choose different options for every label in the source Version Manager project database or project that you are migrating.

- **Baselines**. We recommend migrating version labels to baselines if you use *fixed version labels* to identify revisions associated with milestones or builds that you want to be able to recreate at any time. Baselines allow you to collect all files associated with specific builds so that you can recreate that build at any time. When you migrate

to a baseline, all revisions associated with the original version labels in Version Manager are collected into that baseline.

- **Projects**. You may choose to migrate version labels to projects if you use labels to identify files that belong to unique development efforts or branches. For example, if you have used a particular label to track revisions that are only relevant to a specific branch of a product and then migrate that label to a project, a new project is created in Dimensions that contains all of those revisions. You can assign a named branch to the target project in Dimensions.

  Migrating version labels to projects will typically result in a very different project structure in Dimensions than you have already established in Version Manager. Only choose this option if you want to reorganize your files and projects to reflect your labeling scheme. Do not choose this option if you are happy with the current project structure in Version Manager.

- **Attributes**. Migrate labels to attributes if you want to store all label history after migration. Only use this option for labels that serve a purely informational purpose, but that do not tag revisions that belong to particular builds, milestones, or branches.

Frequently Asked Questions

- *Without floating labels, how do I track the tip revision of ongoing work?*

  From the Dimensions desktop client, you can easily filter a list of items to display only tip revisions, by clicking the **Expand Revisions** button. You can also collect all of of the latest revisions in a project into a baseline by creating a *project baseline*. Finally, you can always display item history, or pedigree, to see the full revision history.

- *Without floating labels, how do I identify which item revisions to deploy or include in a build?*

  If a build should include the latest versions of files, you can create a project baseline that gathers all of the tip revisions in a project.

- *Without fixed version labels, how do I identify item revisions in Dimensions?*

  You can use named branches to identify revisions associated with a particular line of development. With Dimensions CM, you can also relate requests to specific revisions, allowing you to quickly identify work that is relevant to a particular issue.

Migrating Promotion Groups

See "Promotion Groups" on page 22 for more information on how promotion groups compare to features in Dimensions.

You can choose to migrate promotion groups to any of the following. You can choose different options for every group in the Version Manager project database or project that you are migrating.

- **Projects**. Migrate promotion groups to projects if you use groups to identify files that belong to unique development efforts or projects, or if you want to create projects in Dimensions related to specific stages of development. For example, if you have used a particular group in the promotion model to identify revisions that are only relevant to a specific release of a product and then migrate that group to a project, a new project is created in Dimensions that contains all of those revisions.

- **Baselines**. Migrate promotion groups to baselines if you use groups to identify revisions associated with milestones or builds that you want to be able to recreate at

any time. Baselines allow you to collect all files associated with specific builds so that you can recreate that build at any time. When you migrate to a baseline, all revisions associated with the original promotion group in Version Manager are collected into that baseline.

- **Attributes**. You can also migrate promotion groups to attributes to keep a record of promotion group assignments and history.

Frequently Asked Questions

- *How do we represent an item revision making progress to the next higher group in Dimensions?*

  Item revisions can be transitioned, or *actioned,* from one stage of the development lifecycle to the next. For example, if an item is ready for testing and the lifecycle includes a stage called Test, then the latest revision can be transitioned to this stage.

  Depending on how the process model is set up, you may also be able transition requests associated with item revisions from one lifecycle stage to another.

- *How do I generate a build using revisions at a specific state of development*

  You can compile all files in a specific development state into a baseline, that you can then deploy to include in a build.

- *Can I use lifecycles to manage builds?*

  You can automatically copy all files to a build location by deploying baselines. You can tie baseline deployment to lifecycle stages, so that the files are all deployed to build areas as the baselines are moved from one lifecycle stage to another.

Migrating Branches

How you choose to migrate branches depends on how you managed branches in Version Manager. Consider the following recommendations.

- If you use revision numbering to manage branches on a file by file basis, you can choose to migrate complete revision history. In this case, the branch history is preserved, and you can continue to work on revision branches in Dimensions as you have in Version Manager.
- If you use version labels to identify files that belong to different branches of development, you can selectively migrate revisions based on label to unique projects in Dimensions. For example, if all revisions belonging to patch 12 are labeled with "Patch 12," then as part of one migration configuration you can choose to get only those revisions and put them into a Dimensions project called "Patch 12."
- If you have branched by duplicating entire projects or even project databases and storing the branch versions in those duplicates, then you can migrate the entire duplicate project or project database to its own project in Dimensions. This is the simplest scenario, as there is essentially a one to one.

Frequently Asked Questions

- *How do I perform parallel development in Dimensions?*

  Dimensions provides a rich set of features that allow teams to easily branch and merge code streams. You can create projects and streams to store code branches,

and name all revisions in these branches appropriately. Please see "Branches" on page 23.

- *How do I perform concurrent development in Dimensions?*

  Dimensions provides rich synchronization features that allow multiple users to collaborate with ease on common projects. Using the desktop client, Windows Explorer integration, or the Visual Studio and Eclipse integrations, users can quickly synchronize their local workspaces with the repository. This enables you to compare local files with the repository, identify and choose how to resolve conflicts, upload new revisions, and download other users' new revisions and files. This also enables users to share folder and file refactoring with each other, such as moved or deleted files and folders.

- *How do I setup branch numbering so that it is the same as VM numbering?*

  Dimensions projects can be set up to support unnamed branches. By default, if item revision number is set up to follow the standard Dimensions revision numbering scheme (to increase revision numbers by 1 every time a new revision is checked in), unnamed branches within projects assign revision numbers such as 1.1.2, 1.1.3, and 1.1.4.

Migrating Users

You can choose to migrate Version Manager users and groups to Dimensions users and roles, respectively.

Migrating Archives

When choosing the export options, you can choose whether to migrate the archives intact or migrate the archives to non-delta item libraries. Preserving the archives may improve migration performance. To learn more about archive options in Dimensions, see "Archives" on page 29.

Migrating IDE Projects

IDE projects are migrated along with any other files. If you are working with SCC projects in Visual Studio, there are some additional steps you must complete; please see "Migrating Version Manager Visual Studio Projects" on page 51.

Once migration is complete, you will need to open your IDE projects from source control, from within your IDE. For details on doing this, please see:

- The *IDE User's Guide* for information on working with projects in SCC and COM integrations
- The *Eclipse for Dimensions User's Guide*
- The *Visual Studio for Dimensions User's Guide*

Chapter 3

Preparing to Migrate from Version Manager

System Requirements

Dimensions Requirements

To use the version of the Migration Console documented here, you must:

- Install Dimensions 2009 R2 or later. For details, see the *Serena Dimensions Installation Guide*.
- Install the Migration Console and the Dimensions client components to a Windows system.
- Install the Microsoft .NET 2.0 framework to this system.
- If you will log in to your system as a different user than your Dimensions administration user (such as dmsys), add your system user to Dimensions with administrative privileges.
- If Single Sign-On (SSO) is configured for Dimensions, you must ensure that your operating system username match the SSO username and password.
- A DAT item type is required in the target Dimensions product to ensure successful migration. You can define item types using the Dimensions Administration Console. Please see the *Dimensions Process Configuration Guide* for detailed instructions on configurating item types.
- A project type called Project must exist in Dimensions before you can migrate. See the *Process Configuration Guide* for more on creating projects.
- To successfully launch the Migration Console, you must be logged into the system where it is installed as a user that belongs to the Windows Administrator group.

Version Manager Requirements

In addition, the Migration Console requires a full installation of an English version of PVCS Version Manager 8.1.4 or later. If you have an earlier version that is at least 6.8, you can upgrade to Version Manager 8.1.4 prior to migrating to Dimensions CM.

The Migration Console is Windows-based; however, it facilitates migrations from all PVCS Version Manager platforms to all Dimensions CM platforms and databases.

- The Migration Console uses the HOST login ID to run a series of Version Manager commands to validate the success of the migration; you must assign a Version Manager license to that user before running the migration.
- Your Version Manager installation must include at least the following components:
  - Desktop Client
  - Sample Project Database
  - Command-Line Interface

IMPORTANT!

- Windows imposes a limit of 255 characters in a file path. If your migration directory is deeply nested, you could more easily run into this limit during a File mode migration and receive an error message. To help avoid this, you can specify the top-level directory where you will store your migration data by running the Migration Console from the command line using the `-workdir` option:

```
MigrationConsole -workdir:C:\Migrations
```

CAUTION: Although a short work-directory path can help, **it is still possible to exceed the limit if you have a long enough combined Version Manager project path and file name.**

- Migrations must be performed on a local drive. This is the default configuration. If the work directory is on a network drive, the import phase of the migration will fail.

Optimizing Migration Performance for WAN Scenarios

In situations where Dimensions and Version Manager reside on computers located in geographically separate sites, such as offices in different cities or countries, consider the following suggestions to improve migration performance. Because Version Manager does not need to run over the network, these steps may dramatically improve performance.

1. Install and configure the migration console on two systems, one that is geographically near by the Version Manager server, and another that is geographically near by the Dimensions server.
2. Define and run the Version Manager migration from the system that is located near by the Version Manager server (such as another system in the same office building). However, from the Execute Migration screen, choose the **Run Stages** option and select only the **Version Manager Export** and **Transform** checkboxes. This generates a migration output directory, and will not yet attempt to import the data into Dimensions. By default, the migration output directory appears under the following location:

   C:\Documents and Settings\All Users\Application Data\Serena\Migrations
3. Compress this migration directory and the miglist.xml file, located in the \Migrations folder, into a Zip file and transfer this to the system that is located near by the Dimensions server (such as another system in the same office building).
4. Unzip the migration directory and definition to the C:\Documents and Settings\All Users\Application Data\Serena\Migrations directory. You must also ensure that the miglist.xml file is located directly under the \Migrations folder.
5. Run the migration again from this system. Again choose the **Run Stages** option, however select only the **Dimensions Import** checkbox.

Licensing Requirements

To run the Migration Console, you must have a valid Dimensions CM license and a valid Version Manager license.

Preparing PVCS Version Manager Files

Before you migrate, perform the following steps:

- If you will perform an Archive mode migration, make sure that there are no locked revisions in the Version Manager projects to be migrated.
- Disable any get or check out event triggers. If any of these triggers are enabled, migration may fail.
- Make sure that any work in progress is completed in the projects to be migrated.
- Restrict access to the Version Manager archives to be exported.
- Remove all access lists from the Version Manger archives to be exported. To remove the access lists, make sure that all the files are selected, select the Admin | Archive Attributes menu option, select the Advanced tab, and then select **Delete** from the **Access List** drop-down list. Click **OK**.
- Make sure that all desired labels are in place.

NOTE Filenames that include commas may generate errors during migration, however migration will not fail. To avoid these errors, remove commas from filenames in Version Manager.

Team Preparation Overview

When planning a migration, consider the following:

- **Plan and run a prototype.** Your test data should be representative of your production data in quantity and structure. If time permits, a trial conversion of your production data could be worthwhile. You can do this by creating a Dimensions base database just for testing the migration. For details on creating a base database using the create database (CRDB) command with the Dimensions DBA utility (dmdba), please see the *Serena Dimensions System Administration Guide*.
- **Prevent interruptions.** If the migration will run for a long time, take steps to ensure that no backups, planned network outages, software updates, or other interruptions occur.
- **Prevent log-ins.** If migrating into a live database with users, ask them not to log in during the migration. Anything they do will take resources away from the migration and could cause performance problems or other undesirable side effects.
- **Configure notification.** Be aware that if you migrate into a destination database with users, newly migrated objects might generate e-mail notifications. Test this behavior ahead of time, and take appropriate steps (for example, adjusting e-mail settings).

Migrating to UNIX and Linux Servers

Review this topic very carefully if you will migrate from Version Manager to a Dimensions CM Server running on a non-Windows platform, such as supported UNIX and Linux

platforms. Because the migration console is only supported on Windows, you must complete these steps in order to migrate to a remote Dimensions CM server from a Windows system.

Summary of Remote Migration Stages

Stage	Description
1	If you have not already done so, install Dimensions and its repository to the target UNIX or Linux server. For details on installation and setting up the Dimensions CM server, please see the *Serena Dimensions CM Installation Guide for UNIX* and the *Serena Dimensions CM System Administration Guide*.
2	Identify a Windows system in the network that you can use to run the Migration Console. This system must meet the minimum system requirements for the Dimensions CM Server.
3	Install a remote DBMS client, such as Oracle SQL Net, to this Windows system. This is necessary so that you can then connect to the remote Dimensions CM database. For example, if you have purchased it, you can install the Serena Runtime to this system.
4	Install the Dimensions CM server, with the Migration Console and Common Tools options, to this Windows system.
5	Install and configure the correct version of Version Manager to this Windows system.
6	Once installation to the Windows system is complete, you must complete the following steps from the Administration Console, to set up and configure network nodes. Please see the *Serena Dimensions CM System Administration Guide* for detailed information on managing network nodes. a Log into the Distributed Development | Network Administration view. b Select the **Network Nodes** tab. c Click the **New** button and choose **Physical Network Node**. The New / Edit Node dialog box appears. d Complete the dialog box to define this node as the remote Dimensions CM server to which you will migrate, including choosing the correct operating system from the **Operating System** list. e After you have defined the node, select it and then click the + icon under Base Databases to add a new base database to the network node. Configure this base database to point to the base database on the remote server. To define this, you must know the base database name and database connection string.

Stage	Description
7	On Oracle, you must now add the database connection as a local net service name. To do this: a If you installed the Serena Runtime, select Oracle - Dimensions | Configuration and Migration Tools | Net Configuration Assistant from the Windows Start menu. b On the wizard that appears, select the **Local Net Service Name configuration** option and click **Next**. c On the next page of the wizard, select the **Add** option and click **Next**. d Enter the database service name, such as DIM10, in the **Service Name** field and click **Next**. The service name is typically the same as the global database name. To determine the global database name on the Dimensions Oracle server, contact your database administrator. e On the next page of the wizard, select **TCP** and click **Next**. f Enter the host name in the **Host name** field and choose the **Use the standard port number of 1521** option (unless the Oracle port number on the target server has been changed). Click **Next**. g When prompted to test the connection select the **Yes, perform a test** option and click **Next**. h When prompted, enter or confirm the net service name that the local database should use to interact with the remote database. In most cases, this should be the service name that you entered earlier, such as DIM10. i When you are asked to configure another net service name, choose **No**. Choose **Finish** when possible. j Reopen the Net Configuration Assistant. k Select the **Local Net Service Name configuration** option and click **Next**. l Select **Rename**. m Verify that the new connection appears in the drop-down list, and click **Cancel**.

Stage	Description
8	Before you can connect to the remote database from the Windows server, you must store your login information to a password storage file (registry.dat) on the Windows server. To do this, you must use the `dmpasswd` utility on the Windows server. The `dmpasswd` utility ensures that all Dimensions base database names, connection strings, and passwords are encrypted before they are written to disk. If the remote database server is Oracle, then the `dmpasswd` command will look something like the following: `dmpasswd <Base_DB>@<SID> -add -pwd <Password>` Where `Base_DB` is the base database name, `SID` is the Oracle system identifier, and `Password` is your login password. The base database name typically corresponds to your login name for oracle databases. For detailed information on the `dmpasswd` utility, including the correct command format for other DBMS platforms as well as additional options, please consult the *Serena Dimensions CM Command Line Reference*.

9	Test the database connection by making sure that the `pdiff` utility can connect to the base database from the Windows server. `pdiff` is a tool for importing and exporting data to and from a Dimensions CM Product, using PDIFF files.
	Before you can test this, you must set the DMDB environment variable to the following value:
	`<Base_DB_name>@<DB_connection>`
	For example, if the base DB is named qlarius_cm and the DB connection is dim10, use the following command to set the environment variable:
	`set DMDB=qlarius_cm@dim10`
	For details on the DMDB environment variable, please see the *Serena Dimensions CM Adminstrator's Guide*.
	Then, to test the connection, you will run a command like the following from the command line, that will dump a partial process model from a product into a PDIFF file:
	`pdiff <product_id> -dump_cpl`
	Where `product_id` is the ID of the product from which you will dump the process model. For example, if you installed the Qlarius sample process model to the Dimensions CM server, the command would appear as follows:
	`pdiff qlarius_cm -dump_cpl`
	The command succeeds if the connection is made.
	For details on the `pdiff` utility and the `dump_cpl` function, please see the *Serena Dimensions CM System Administration Guide*.

Having completed the above steps, you can now set up the migration as you would for any other scenario.

Chapter 4

Migrating from Version Manager

Configuring a Migration

To configure a migration, you must complete the following procedures:

Creating a New Migration

To create a migration, perform the following steps:

1 Open the Migration Console. You can open it from the Windows start menu by selecting Programs | Serena | Dimensions <*version*> | Migration Console.

IMPORTANT! To successfully launch the migration console, you must be logged into Windows as a user who belongs to the Windows administrator group.

2 Click the **New Migration** button to define a new migration, or select an existing migration and click the **Edit Migration** button.

3 Enter a name for the migration definition, and choose the VM to Dimensions option.

Setting Target Dimensions Server Options

On the **Dimensions Destination** screen, you must set the necessary Dimensions server connection information. You can test connectivity to the Dimensions server from here, and choose the default destination product, project, stream, and design part for this connection.

Set the Dimensions server options as follows:

1 If you have already defined a connection profile using the Dimensions desktop client, you can choose this profile from the **Profile** list.

2 Under **System**, enter the user name and password that the Migration Console should use to log in and retrieve data from Dimensions CM. This may be the administrative user that was associated with Dimensions during Dimensions installation, such the default user dmsys.

3 Under **Dimensions**, enter the Dimensions CM server name, database name, and database connection string, if required. With Oracle databases, the connection string is equivalent to the system identifier (SID). The database connection information is

determined during database and Dimensions installation. The system administrator responsible for database and Dimensions installation should supply this to you.

IMPORTANT! Even if Dimensions is installed to the local machine, do not enter *localhost* as the server name. You must use the network node name as defined in the Dimensions Administration Console. You can use *localhost* only if *localhost* is defined as a network node.

4 Click the **Test Connectio**n button to verify that you can connect to the server with the information you have supplied.

5 Under **Default Dimensions Destinations**, specify the target product, project or stream, and part for the migrated data. You do not need to enter an existing product or project; if these do not already exist, they will be created for you during migration.

6 Click the **Advanced** button to display and set a number of additional migration options. These options are not necessary for successful migration:

Field	Description
Item Library Path	The path to the location where the item library files should be stored for the destination project. This field only applies if you are migrating to a new product. If you are migrating to an existing product, the existing item library path is used.
Parent Part	The parent design part for the destination design part.
Variant	The design part variant to which the files will be migrated. By default, this is **A**.
Product Manager	Additional user or group to whom the PRODUCT MANAGER role is assigned for the destination product, once the destination product is created.
Product Manager Role	The user or group to whom the $PRODUCT-MANAGER role is assigned for the destination product.
Part Category	An attribute of the design part that indicates its general purpose.
Item type for comments (>1978)	When migrating files from Version Manager with change descriptions that exceed the maximum number of characters that Dimensions CM allows by default, the change descriptions are migrated to items that are then related to the migrated revision. Define the *item type* here.
Relationship for comments (>1978)	When migrating files from Version Manager with change descriptions that exceed the maximum number of characters that Dimensions CM allows by default, the change descriptions are migrated to items that are then related to the migrated revision. Define the *relationship type* here.

7 Click the **Apply** button to save the Dimensions connection settings, or **OK** if you are done setting migration options.

Choosing Version Manager Project Databases

On the the **Version Manager PDBs** screen, you must set the options for the Version Manager project database that you want to migrate.

To choose Version Manager project databases:

1. To choose project databases to migrate, click **Add** to browse for a project database to add to the list. To remove a PDB from the list, select the PDB and then click **Remove**.
2. On the Input Version Manager Project Database Information dialog box, browse to select the root project database folder. Then, enter the user name and password that you will use to log in to the project database.

Choosing Version Manager Projects

On the **Version Manager Projects** screen, you must choose the specific project databases and projects that you want to migrate.

To choose Version Manager projects:

1. Select the **Migrate All PDBs and Projects** option to migrate all project databases and projects contained within them.
2. Select the **Select PDBs and Projects to Migrate** option to select specific project databases and projects to migrate. The check boxes have three possible states:
   - **Unchecked** – Project and all subprojects are not selected for migration.
   - **Checked** – Project and all subprojects are selected for migration.
   - **"Half" checked (grey)** – Project, files under the project, and some (but not all) subprojects are selected for migration.

NOTE Because there is no selection of individual files (only folders), you cannot skip migration of the files within a folder if any subprojects in that folder are selected for migration.

Setting Migration Options for Labels and Promotion Groups

On the **Map Labels** screen you can automatically migrate all labels to baselines and groups to projects or streams, or manually define the appropriate mapping from Version Manager labels and promotion groups to Dimensions attributes, projects, streams, and baselines.

To automatically migrate all version labels to baselines and promotion groups to projects or streams, select the Migrate All Labels and Groups option.

To manually map labels and groups, select **Migrate Selected Labels and Groups** and complete the following options.

NOTE – Resolving baseline name conflicts: If a baseline name conflict arises during execution of the migration (this can be caused by truncation or case-insensitivity), a modified version of the baseline name is used (a number is appended to the end of the baseline name). If this would would result in too many characters, the baseline name is truncated sufficiently to accommodate the appended digits.

To create a new mapping, click **Insert**, and then set the fields as described below:

Field name	Description
Label/Group	Enter the Version Manager label or promotion group that you want to map.
Map Type	Select one of the following options in the drop-down list: **Label-to-Attribute** **Label-to-Project** **Label-to-Stream** **Label-to-Baseline** **Group-to-Project** **Group-to-Stream** **Group-to-Baseline** **Group-to-Attribute**
Dimensions Attribute/ Workset/ Baseline	Enter the name of the attribute, project, stream, or baseline.
Advanced Baseline button	When you choose to map a label or group to a Dimensions baseline, you can click this button to display the Baseline Settings dialog box. From this dialog box you have the following additional options: ▪ **Baseline Type**: Specify the type of baseline that the migration will create. The baseline type must already exist in the target Dimensions database. ▪ **Baseline Template**: The template to be used for the baseline that the migration will create. This template must already exist in the target Dimensions CM database. If you do not specify a baseline type or template, the migration will assign the default type and template in the process model.

Revisions Options

From the Revisions Options screen, choose whether to migrate all revisions from Version Manager, or migrate just a specific subset of revisions.

Choose from the following options:

- **Migrate all revisions**: To migrate all revisions from the Version Manager archives to Dimensions.
- **Filter revisions**: To limit the revisions you will migrate. You can then choose either or both of the following options:

- **Specify number of trunk revisions to migrate for each file** to limit the number of trunk revisions you will migrate for each file. Enter the number.
- **Migrate revisions after** to only migrate trunk revisions created after a specific date. Click the date, month, and year in sequence to type a new value for each. For example, click the date and type *a* to select April.

Migrating Users and Groups

On the Groups and Users Options screen, select from the following options:

- **Migrate VM groups to DM roles** to migrate Version Manager groups to Dimensions roles
- **Migrate VM users** to migrate Version Manager users to Dimensions users

Remember that you must set up privileges appropriately in Dimensions after migration is complete.

Copying a Migration Configuration

To create a new migration definition based on an existing one, select the migration that you want to copy, and then click **Copy Migration**.

Enter a name for the new migration. You can then optionally change any of the settings as described in the previous topics. Only key configuration files are copied when you copy a migration. This excludes any migrated files that you might already have exported from Version Manager.

Migrating Incrementally

The migration console migrates files incrementally. Incremental migration allows you to:

- Spread the migration out over a longer period to avoid having to re-migrate if network or server issues force migration to stop.
- Stop the import phase of a migration at any time, and then restart it later without having to re-import the files that have already been migrated. For example, if network issues force migration to stop before the import is complete, you can simply restart the import phase later on. For details on starting migrations and specifically about the **Dimensions Import** phase, see "Choosing Migration Stages" on page 47.

Running a Migration

When you are ready to run a migration, complete the following steps:

1 From the Migration Console, select the migration that you want to run and click **Run Migration**. The Execute Migration dialog box appears.

2 Before starting the migration, set any migration options correctly. These include:

- Choosing which of the migration stages you want to run. This includes running all migration stages, or running only the *export* stage, *transform* stage, or *Dimensions import* stage. Please see "Choosing Migration Stages" on page 47.
- Set the Dimensions destination options. See "Setting Dimensions Destination Options" on page 48.
- Choosing the export mode. See "Choosing the Export Mode" on page 48.

3 Once you have set all of the options correctly, start the migration. See "Starting Migration" on page 48.

You can collapse or expand the execution options, progress details, and error/warning messages using the arrow buttons.

Choosing Migration Stages

When setting options from the Execute Migration dialog box, you can choose which migration stages you will run from the **Execution Options** area. Consider the following:

Enable this option...	If you want to...
Run All	Complete all migration stages at once.
Version Manager Export	To export the data from Version Manager. If you are migrating from a Version Manager server that is located a great distance away from the Dimensions server, consider running just the Version Manager Export and Transform options, then transfering the resulting migration directory to the location of the Dimensions server, and then completing the Dimensions Import option from there. See "Optimizing Migration Performance for WAN Scenarios" on page 35.
Transform	Transform the data into the correct import format.
Dimensions Import	Import data into Dimensions. If you have both a Dimensions test server and a product server, you might re-run the transform and import stages after changing the destination server from test to production. If **Load Baselines** or **Run All** is selected, the Migration Console generates and executes the baseline migration script (baselines.cmd) that migrates selected Version Manager labels and promotion groups to Dimensions baselines. If **Run Stages** and **Dimensions Import** are selected but **Load Baselines** is not, the Migration Console generates the scripts but does not execute them. If **Validate Import** or **Run All** is selected, the Migration Console generates and executes an import validation script (dmimport_validate.cmd). If **Run Stages** and **Dimensions Import** are checked but **Validate Import** is not checked, this validation script is generated but not executed. If you wish to disable import validation (to speed up processing, for example), you can clear the Validate Import check box before starting the migration.

Setting Dimensions Destination Options

The fields under **Dimensions Destination** are filled in with the target product, project or stream, and part from the migration configuration. Optionally change these options here before running the migration.

Choosing the Export Mode

Under **Export Mode**, you can choose to migrate Version Manager archives intact to a Dimensions delta-library (the **Archive Mode** option), or you can migrate the individual revisions one at a time to a non-delta item library (the **File Mode** option).

Serena does not recommend using delta storage, as doing so doing so can negatively affect day-to-day performance after migration, as well as cause problems with end-of-line (EOL) characters.

Starting Migration

Once you have set all of the options, click the **Start** button to start the migration.

Clicking the **Cancel** button may not immediately stop the migration process. Certain operations (specifically the import phase as well as trigger commands and baseline creation scripts) may continue until completed.

Using the Events Pane

This pane contains error, warning, and informational messages. You can double-click an error or warning message to open a help viewer.

Reviewing Migration Results

When a migration is completed, review the information in the Execute Migration window, review any warnings or errors during any of the migration phases (export, transform, import), and make sure that the expected data is in Dimensions.

You can re-run the migration after fixing any errors, however you must complete some steps before doing this. Please continue to "Iterating Migration" on page 48.

Iterating Migration

You may need to re-run your migration to address errors that occured during previous attempts, for example after cleaning up files in Version Manager or choosing new options in the Migration Console. You must complete the following steps before running migration again, or migration may fail:

1 From the Cross Project Catalog window in the Dimensions desktop client, select **All Baselines**. Choose the product and open the baselines.

2. From the All Baselines window, select all of the baselines and select Baseline | Delete.
3. Once the Baselines are deleted, close the All Baselines window.
4. Select Edit | Find | Items.
5. On the Find Item dialog box, select the product from the **Product** list, then click the **Open** button.
6. Expand the revisions by clicking on the **Expand items display** button.
7. Select all items, and select Item | delete to delete all item revisions.
8. Select Project | Open Project.
9. On the Open Project dialog box, select the **global project** and click Open.
10. Once the global project has opened, select Project | Delete.
11. In the dialog boxes, select the Product and the project into which the VM data was migrated, then click **OK**.
12. Verify that the items were deleted. From the $Generic:$Global project, select Edit | Find | Items and search for the product in question. This should return 0 items.
13. Delete the following directory:
    C:\Documents and Settings\All Users\Application Data\Serena\MigrationConsole
14. From the Console window of the Desktop Client (View | Console) or from a DMCLI prompt, run the following command to rename the Product. This must be a unique name for the migration. If a successful Migration has been done, use a new, unique name.

    Rename /product=<OLD_NAME> /new_id=<NEW_NAME>
15. Log in to the Administration Console.
16. Open the **Items** module and change to the new product. Verify that the item types shown are correct.
17. Close the Administration Console.
18. Open the Migration Console and edit the migration.
19. Change the default target product to the new product. Please see "Setting Target Dimensions Server Options" on page 42.
20. Save the changes.
21. Re-run the export and transform stages. Please see "Choosing Migration Stages" on page 47.
22. Close the Migration Utility.
23. Now, re-run the Dimensions import stage of the Migration again.
24. When migration is complete, you can rename the product back to the correct name using the rename command.

Refactoring Branched Projects

If you migrated a Version Manager project database that includes branched code that is organized into sub-projects, you may need to refactor your projects in Dimensions once migration is complete. Branches in Version Manager and Dimensions are managed in fundamentally different ways:

- In Version Manager, branched code may be stored in separate sub-projects within a common project database. Each branch may refer to different revisions of shared archives. For example, the main line of development (or trunk) may coexist alongside a separate line of development that was created to support a small maintenance or patch release of the code. The trunk and the branch may reside within their own sub-folders within the shared project database, and each of which may contain references to many common files that are shared between them.
- Dimensions CM does not support archive sharing. Code branches are typically stored in their own projects, rather than in sub-folders alongside each other in one project. Files that are common to multiple projects are copied into each project, rather than shared. Then, when work is complete on a given branch, the changes to the copied files may be merged back into the main line of development, or into another branch.

Once migration is complete, determine whether any code branches have been imported to Dimensions CM as separate sub-folders alongside other branches or the main line of development. If this has occured, consider moving the branch code to new projects.

Chapter 5

Migrating Version Manager Visual Studio Projects

Overview of Visual Studio SCC Migration

You can use the migrateIDE utility to complete migration of Visual Studio SCC projects from Version Manager to Dimensions CM. This utility completes the following steps when you run it:

1. Locates the Visual Studio project and solution files in Dimensions, and checks them out. This is the location in Dimensions to which you have already migrated the files from Version Manager.
2. Updates the source control bindings in the Visual Studio project and solution files so that they are correctly bound to Dimensions CM, and to the appropriate product and project in Dimensions CM.
3. Checks the updated files back in to Dimensions CM.

Supported Migrations

This section summarizes prerequisites, and supported Visual Studio project types.

System Requirements

The Visual Studio SCC project migration is supported on Windows, and requires the Microsoft .NET Framework, version 2.0. If you do not already have the .NET Framework, you can install it through Windows Update.

Supported Visual Studio IDE Project Types

You can migrate SCC projects from the following Visual Studio project types:

- Visual Studio 2003/2005 Visual C#
- Visual Studio 2003/2005 Visual C++
- Visual Studio 2003/2005 Visual Basic
- Visual Studio 6 C++
- Visual Studio 6 Visual BAsic

Visual Studio Files Updated by Migration

The migration utility updates the following Visual Studio project and solution files when the projects are migrated, to change the source control binding from Version Manager to Dimensions CM:

- Microsoft Visual Studio .NET solutions (.sln)
- Microsoft Visual Studio .NET C# projects (.csproj)
- Microsoft Visual Studio .NET Visual Basic projects (.vbproj)
- Microsoft Visual Studio .NET C++ projects (.vcproj)

- Microsoft Visual Studio 6 workspace (.dsw)
- Microsoft Visual Studio 6 C++ projects (.dsp)
- Microsoft Visual Studio 6 Visual Basic projects (.vbp)

Supported Dimensions IDE Client

After migration is complete, you can access the migrated projects using the Dimensions CM SCC integration to Visual Studio. See the *Serena Dimensions CM IDE User's Guide* for details.

Repairing SCC Bindings

Before you migrate your Visual Studio projects, you must first make sure that the project structure as defined within the Visual Studio solution matches the actual folder structure in Dimensions. If it does not, then you must manually set this up.

What Are SCC Bindings?

SCC bindings are markers stored within Visual Studio solution and project files that determine the relationships between the IDE projects and their counterpart version control projects, such as Version Manager or Dimensions CM projects and folders. Ideally, the structure of the Visual Studio projects within the solution maps to the project structure in the version control repository.

Why is it Necessary to Repair SCC Bindings?

Using the Version Manager integration to Visual Studio, it is possible to map individual Visual Studio projects to individual Version Manager projects, regardless of the actual project structure in Visual Studio. This means that the project structure in Visual Studio may be different from the project structure in Version Manager. The Version Manager project structure is migrated as is to Dimensions CM.

To work successfully with the Dimensions CM integration to Visual Studio, it is critical that the Visual Studio project structure match the corresponding folder structure in Dimensions CM. If the original project structure in Version Manager did not match the project structure in Visual Studio, then you must manually update the SCC bindings, or references, within the Visual Studio solution to match the folder structure in Dimensions.

How to Repair SCC Bindings

If necessary, to correct the SCC bindings:

1. Once the files have been migrated to Dimensions but before you run the IDE migration utility, check out the Visual Studio solution and project structure from Dimensions CM to a working directory.
2. In turn, open each Visual Studio solution and project file (such .sln, .vbproj) in a text editor (such as Windows Notepad) and complete the following steps:

- Look for any SCC bindings. These appear something like the following:

  `SccProjectName<#> = /<path to project files>`

  For example:

  `SccProjectName10 = /code/path`

- Ensure that the path defined here is correct. For example, if the relative path should be changed from the example above to include a root /src directory, then add this to the path:

  `SccProjectName10 = /src/code/path`

- Repeat as necessary for every solution and project file. If you need to make the same change to the path in every file, consider creating a script to automate the changes.

3 Check the files back in to Dimensions and continue on to using the IDE migration utility. Now that you have made these changes, the SCC marker files will be created in the correct project folders.

Using the SCC IDE Migration

The following procedures will help you migrate your Version Manager SCC projects to Dimensions.

- "Preparation" on page 54
- "Running the Migration Command" on page 54

Preparation

Complete the following steps before you begin migration of IDE projects:

1 You must first make sure that the SCC bindings are correct before using this utility. See "Repairing SCC Bindings" on page 53.

1 Make sure that the Dimensions user that you will use to run this utility has the necessary privileges to check out, check in, and add.

2 Create a temporary directory on your local system to hold files during the update process. This is the work area that you will specify on the command line when you run the utility. The utility then uses this location to store all files that it will update. To avoid any potential for file conflicts, this directory should be empty.

3 Use the migration console to migrate your project database, specifying the same product, workset (project), and design part that you will specify on the command line when you run this SCC IDE migration utility. This is necessary because the IDE migration updates Visual Studio project files that are stored in Dimensions. See "Migrating from Version Manager" on page 41.

Running the Migration Command

To run the IDE project migration:

1. Launch a Windows command prompt.
2. Navigate to the VM Migration Windows bin directory. By default, this is:
   `C:\Program Files\Serena\Dimensions <version>\CM\Migration\bin\w32`
3. From this directory, run the migrateIDE command in the following format:
   `migrateIDE serverName dbName dbDsn userId password product workset part workArea`

   Refer to the following table for detailed information on each parameter.

Parameter	Description
`serverName`	The Dimensions host database
`dbName`	The Dimensions database instance
`dbDsn`	The Dimensions database SID (system identifier)
`userId`	The Dimensions database user name
`password`	The Dimensions database password
`product`	The Dimensions product that you are updating
`workset (project)`	The Dimensions project (workset) that you are updating
`part`	The Dimensions design part that you are updating
`workArea (root path)`	The path to the work directory that you set up for the migration utility to use

Example Command

```
migrateIDE OR04735 intermediate DIM10 dmsys dmsysdmsys PROD1 PROJECT
    DESPART C:\temp\IdeSccTmp
```

Chapter 6

Version Manager Post-Migration Checklist

Perform the following steps after you run a migration:

1. Verify that there were no unexpected warnings or errors during the migration by studying the Errors/Warnings log in the Execute Migration window.

2. Review the `AuditReport.html` file, which the migration utility generates and which you can display by clicking the **View Migration Audit Report** button in the Execute Migration dialog box. (The report is located in the directory containing the migration; for example, `$DM_ROOT\cm\Migration\Console\Migrations\TestMigration`.) This very useful report contains the migration configuration details, timing metrics for migration execution, validation information, and links to specific log files for errors and warnings encountered during the migration.

3. Clean up shared archives.

   Version Manager allows multiple versioned files to reference the same physical archive. If the same Version Manager archive is migrated to two separate locations within a single Dimensions project, multiple copies of the archive will exist in the Dimensions item library (Archive mode). Review the warning messages to identify archives shared within the Version Manager PDB, shared across PDBs, and shared outside the current migration.

4. Set file size. Newly imported items in Dimensions will show a file length of -1 until the first operation is performed on them. Perform a Get of the global project or item catalog to automatically set the file length values.

5. Tune Oracle. For best results, perform tuning on the database whenever large amounts of data are added to the Dimensions system.

   For details, consult the *Serena Dimensions CM System Administration Guide*.

Part 2

Migrating from CVS

Chapter 7

About the CVS Migration

Introduction

You can use the Dimensions Migration Console to migrate CVS directories and objects to Dimensions. The Migration Console is a graphical front end to a set of XML-based utilities that move data from supported sources into Dimensions products, allowing you to define how the data is mapped into Dimensions. The Migration Console includes the following features:

- You can migrate all or part of a CVS repository into a Dimensions CM project hierarchy.
- Migration preserves your CVS repository structure.
- You can exclude any subset of CVS folders.
- You can target subsets of CVS files to specific Dimensions design parts, projects, named branches, and item types.
- You can migrate CVS tags and branches groups to Dimensions multi-valued attributes, projects, and baselines.

Migration automatically creates process model data in Dimensions, including users, groups, privileges, item types, design parts, projects, baselines, and relationships.

Data Mapping

The following table shows how the migration process transforms CVS data into Dimensions data:

CVS	Dimensions CM
Repository	Product
Users	Users
Files and revisions	Items, revisions, named branches
Directories	Folders
Tags and branches	Projects, baselines, attributes, named branches

What Is Migrated

The Migration Console supports migration of the following data from CVS to Dimensions CM:

- You can optionally migrate all CVS users to Dimensions CM.
- You can migrate everything within a CVS repository, or you can specify the specific directories to migrate. CVS directories are migrated to folders in Dimensions CM.
- CVS revisions become item revisions.
- You can migrate individual revisions to a non-delta item library (File mode).
- CVS trunk and branch items migrate to attributes, projects, named branches, and baselines in any combination that you specify.

- Empty folders

What Is Not Migrated

The Migration Console does not support migration of certain types of data:

- User permissions and groups.
- Multi-byte characters.
- Local workspaces.

Chapter 8

Preparing to Migrate from CVS

System Requirements

Dimensions Requirements

To use the Migration Console with CVS, you must:

- Install Dimensions 2009 R2 or later. For details, see the *Serena Dimensions CM Installation Guide*.
- If you will log in to your system as a different user than your Dimensions administration user (such as dmsys), add your system user to Dimensions with administrative privileges.
- If Single Sign-On (SSO) is configured for Dimensions, you must ensure that your operating system username match the SSO username and password.
- Run the Migration Console from an installation of the Dimensions Server. The CVS client must also be installed to this system.

CVS Requirements

You can use the Migration Console with all versions of CVS that support the `rls`, `rlog`, `co`, `login`, `version`, and `logout` commands.

The Migration Console supports the pserver protocol for communicating with CVS servers.

IMPORTANT!

- Migrations must be performed on a local drive. This is the default configuration. If the download directory is on a network drive, migration will fail.

Storage Requirements

To ensure successful migration, follow these guidelines for disk storage:

- Make sure the location where temporary migration files are stored has at least twice the storage capacity of the size of the originating CVS repository. For example, if you are migrating a repository that is 1 GB is size, make sure that the location for temporary files has at least 2 GB free. By default, this location is: c:\Documents and Settings\All Users\Application Data\Serena\Migrations
- Make sure that the location to which CVS revisions are downloaded has space available that is at least equivalent to the size of the originating CVS repository. By default, this is:
  C:\Documents and Settings\All Users\Application Data\Serena\Migrations\<*migration name*>\cvsTemp\<*CVS server*>\<*CVS repository*>
  However you can customize this location during migration.
- Make sure that the Dimensions item library location has enough space to store all revisions to be migrated. If the originating CVS repository is 1 GB and you are migration all revisions, make sure that at least 1 GB is available on the item library location.

Licensing Requirements

To run the Migration Console, you must have a valid Dimensions CM license.

Preparing CVS Files

Before you migrate, perform the following steps:

- Make sure that any work in progress is completed in the projects to be migrated.
- Restrict access to the CVS files to be exported, so that only the CVS users that the Migration Console users have access to them.

Team Preparation Overview

When planning a migration, consider the following:

- **Plan and run a prototype.** Your test data should be representative of your production data in quantity and structure. If time permits, a trial conversion of your production data could be worthwhile. You can do this by creating a Dimensions base database just for testing the migration. For details on creating a base database using the create database (CRDB) command with the Dimensions DBA utility (dmdba), please see the *Serena Dimensions CM System Administration Guide*.
- **Prevent interruptions.** If the migration will run for a long time, take steps to ensure that no backups, planned network outages, software updates, or other interruptions occur.
- **Prevent log-ins.** If migrating into a live database with users, ask them not to log in during the migration. Anything they do will take resources away from the migration and could cause performance problems or other undesirable side effects.
- **Configure notification.** Be aware that if you migrate into a destination database with users, newly migrated objects might generate e-mail notifications. Test this behavior ahead of time, and take appropriate steps (for example, adjusting e-mail settings).

Migrating to UNIX and Linux Servers

Review this topic very carefully if you will migrate from CVS to a Dimensions CM Server running on a non-Windows platform, such as supported UNIX and Linux platforms. Because the migration console is only supported on Windows, you must complete these steps in order to migrate to a remote Dimensions CM server from a Windows system.

Summary of Remote Migration Process

Stage	Description
1	If you have not already done so, install Dimensions and its repository to the target UNIX or Linux server. For details on installation and setting up the Dimensions CM server, see the *Serena Dimensions CM Installation Guide for UNIX* and *Serena Dimensions CM System Administration Guide*.
2	Identify a Windows system in the network that you can use to run the Migration Console. This system must meet the minimum system requirements for the Dimensions CM Server.
3	Install a remote DBMS client, such as Oracle SQL Net, to this Windows system. This is necessary so that you can then connect to the remote Dimensions CM database. For example, if you have purchased it, you can install the Serena Runtime to this system.
4	Install the Dimensions CM server, with the Migration Console and Common Tools options.
5	Install and configure the correct version of CVS to this Windows system. See "System Requirements" on page 66 for detailed requirements.
6	Once installation to the Windows system is complete, you must complete the following steps from the Administration Console, to set up and configure network nodes. See the *Serena Dimensions CM System Administration Guide* for detailed information on managing network nodes. a Log into the Distributed Development | Network Administration view. b Select the **Network Nodes** tab. c Click the **New** button and choose **Physical Network Node**. The New / Edit Node dialog box appears. d Complete the dialog box to define this node as the remote Dimensions CM server to which to migrate, including choosing the correct operating system from the **Operating System** list. e After you have defined the node, select it and then click the **+** icon under Base Databases to add a new base database to the network node. Configure this base database to point to the base database on the remote server. To define this, you must know the base database name and database connection string.

Stage	Description
7	On Oracle, you must now add the database connection as a local net service name. To do this: a If you installed the Serena Runtime, select Oracle - Dimensions | Configuration and Migration Tools | Net Configuration Assistant from the Windows Start menu. b On the wizard that appears, select the **Local Net Service Name configuration** option and click **Next**. c On the next page of the wizard, select the **Add** option and click **Next**. d Enter the database service name, such as DIM10, in the **Service Name** field and click **Next**. The service name is typically the same as the global database name. To determine the global database name on the Oracle server, contact your database administrator. e On the next page of the wizard, select **TCP** and click **Next**. f Enter the host name in the **Host name** field and choose the **Use the standard port number of 1521** option (unless the Oracle port number on the target server has been changed). Click Next. g When prompted to test the connection select the **Yes, perform a test** option and click **Next**. h When prompted, enter or confirm the net service name that the local database should use to interact with the remote database. In most cases, this should be the service name that you entered earlier, such as DIM10. i When you are asked to configure another net service name, choose **No**. Choose **Finish** when possible. j Reopen the Net Configuration Assistant. k Select the **Local Net Service Name configuration** option and click **Next**. l Select **Rename**. m Verify that the new connection appears in the drop-down list, and click **Cancel**.

Stage	Description
8	Before you can connect to the remote database from the Windows server, you must store your login information to a password storage file (registry.dat) on the Windows server. To do this, you must use the `dmpasswd` utility on the Windows server. The `dmpasswd` utility ensures that all Dimensions base database names, connection strings, and passwords are encrypted before they are written to disk. If the remote database server is Oracle, then the `dmpasswd` command will look something like the following: `dmpasswd <Base_DB>@<SID> -add -pwd <Password>` Where *Base_DB* is the base database name, *SID* is the Oracle system identifier, and *Password* is your login password. The base database name typically corresponds to your login name for oracle databases. For detailed information on the `dmpasswd` utility, consult the *Serena Dimensions CM Command Line Reference*.

9	Test the database connection by making sure that the `pdiff` utility can connect to the base database from the Windows server. `pdiff` is a tool for importing and exporting data to and from a Dimensions CM Product, using PDIFF files.
	Before you can test this, you must set the DMDB environment variable to the following value:
	`<Base_DB_name>@<DB_connection>`
	For example, if the base DB is named qlarius_cm and the DB connection is dim10, use the following command to set the environment variable:
	`set DMDB=qlarius_cm@dim10`
	For details on the DMDB environment variable, please see the *Serena Dimensions CM System Administration Guide*.
	Then, to test the connection, you will run a command like the following from the command line, that will dump a partial process model from a product into a PDIFF file:
	`pdiff <product_id> -dump_cpl`
	Where *product_id* is the ID of the product from which you will dump the process model. For example, if you installed the Qlarius sample process model to the Dimensions CM server, the command would appear as follows:
	`pdiff qlarius_cm -dump_cpl`
	The command succeeds if the connection is made.
	For details on the `pdiff` utility and the `dump_cpl` function, see the *Serena Dimensions CM System Administration Guide*.

Having completed the above steps, you can now set up the migration as you would for any other scenario.

Chapter 9

Migrating from CVS

Using the Migration Console

When you first run the Migration Console, you see a list of predefined migrations. You can sort this list by clicking on a column heading.

Each migration has a status, indicated in the **Execution Status** field.

To define a new migration, click **New Migration**. To copy, edit, or run an existing migration, select the migration in the list, and then click the appropriate button.

NOTES

- You can click the **Delete Migration** button to remove a migration from the list, however this does not actually delete the migration definition. Doing so could delete migration data that you actually need.
- You can also optionally delete all migration data produced during the migration process.
- At any time when you are creating or editing a migration, you can click **OK** to save all changes and close the designer, **Cancel** to close the designer and receive a prompt asking whether you want to save changes, or **Apply** to save changes while leaving the designer open.

Configuring a Migration

To configure a migration, perform the following steps:

1. Click the **New Migration** button to define a new migration, or select an existing migration and click the **Edit Migration** button.
2. Enter a name for the migration in the **Migration ID** field.
3. Select the **CVS to Dimensions** option.
4. Click **Next**.

Setting Target Dimensions Server Options

5. On the **Dimensions Connection** screen, you must set the necessary Dimensions server connection information. You can test connectivity to the Dimensions server from here, and choose the default destination product, project or stream, and design part for this connection.

IMPORTANT! Remember that the Dimensions Server must be installed locally, or if it is on a UNIX or Linux server, please see "Summary of Remote Migration Process" on page 68.

6. Under **System**, enter the user name and password that the Migration Console should use to log in and retrieve data from Dimensions CM. **It is important to make sure that this operating system user also exists in Dimensions.**
7. Under **Dimensions**, enter the Dimensions CM server name, database name, and database connection string. Click the **Test Connection** button to ensure that you can connect to Dimensions using the connection information.
8. Click **Next**.

9 Under **Specify Dimensions destination to migrate into**, specify the target product and part for the migrated data. You must select an existing product, however you can create a new design part. If you choose to create a new design part, you can also assign a design category and parent part.

10 Under **Specify where the trunk revisions are to be migrated to**, choose whether to migrate the CVS trunk directory to a stream or a project. Serena recommends migrating to Dimensions streams, which support more agile collaborative development practices. Enter a name for the stream or project. If you will migrate the CVS trunk to a project, choose the project type from the **Dimensions project type list**.

11 Under **Specify what should we do with long comments**, choose to either truncate (cut short) comments over 1978 characters, or store the entire comments in items in Dimensions. If you choose to store the comments in items, you must also choose the item type. This can be any type of item. Also, you must specify the name of the relationship between the comment items and the items that they are associated with. You can choose an existing relationship type, or enter a new one.

12 Click **Next**.

Defining the CVS Connection Options

13 Under **Authentication**, enter the user name and password for the user account that the Migration Console will use to log in to CVS and extract data.

The Migration Console supports the pserver protocol for communicating with CVS servers.

14 Under **Connection**, enter the CVS host name, as well as the path to the repository that you will migrate, and the path to the local CVS download directory.

15 To log in to CVS using the default port of 2401 select the **Use default port** option. If the port is set to something other than the default, select the **Use port** option and enter the port number.

16 Click the **Test connection** button to verify that you can connect to CVS using the connection settings you entered. A red exclamation point indicates an error. You can hover over the error to display more information.

17 Click **Next**.

18 To log in to CVS using the default port of 2401 select the **Use default port** option. If the port is set to something other than the default, select the **Use port** option and enter the port number.

19 Click the **Test connection** button to verify that you can connect to CVS using the connection settings you entered. A red exclamation point indicates an error. You can hover over the error to display more information.

20 Click **Next**.

Choose CVS directories to migrate

21 Choose the **Migrate all CVS directories** option to migrate all directories in the CVS repository, or choose the **Select CVS directories to migrate** option and select the specific directories you want to migrate. Files and subdirectories within the selected directories are migrated according to the following rules:

- All files within a selected directory will be migrated. To select and de-select specific directories, you must expand the directory tree and change your selections. A directory name in bold black text indicates that all subdirectories are selected. A directory name in bold gray text indicates that some subdirectories are selected.

- No files within an unselected directory will be migrated, however you can select subdirectories under an unselected directory.

22 Enter the path or click **Choose Dir** to select a temporary directory to use during migration.

23 Click **Next**.

Setting migration options for CVS tags and branches

24 If you want to migrate all tags and branches from CVS to baselines, projects, or attributes in Dimensions, select the **Migrate all tags and branches** option. Otherwise, to migrate a selection of tags and branches, select the **Select tags and branches to migrate** option and complete the following steps:

a To load a list of tags and branches from the CVS repository to the Migration Console, click the first empty cell under the Tag/Branch columan and select **click to load tags/branches** from the list. You are then prompted to choose whether to list all tags and branches from the repository, or just for selected projects. You should click this button every time you edit the migration configuration, before you complete the mapping options. This will ensure that all of the latest tag and branch information is loaded into the Migration Console. Keep in mind that loading all tags and branches from the entire repository can take a very long time, however doing so ensures that no tags or branches are missed.

b If you are migrating specific tags and branches, click the **Add Row** button to add an entry to the table for every tag and branch that you want to migrate. For each row, you can choose the tag or branch under the **Tag/Branch** column.

c For each tag or branch listed under **Tag/Branch**, choose whether to migrate to a baseline, project, stream, or attribute from the **Map Type** column. The names that will be given to the resulting baselines, projects, streams, and attributes display in the **Dimensions Destination** column.

25 Click **Next**.

26 From the Dimensions Named Branches screen, enter named branch names in Dimensions for migrated items. You can enter named branch names for:

- Items in the CVS trunk. The named branch will have the following format: *<ItemID>*.A-*<ItemType>*;*<NamedBranch>*#*<Revision>* where *NamedBranch* is the branch name you enter here.
- Items in CVS branches. The named branch will have the following format: *<ItemID>*.A-*<ItemType>*;*<NamedDimBranch><CVSBranch>*#*<Revision>* where *NamedDimBranch* is the branch name you enter here, and *CVSBranch* is the name of the originating CVS branch.

27 Optionally, to automatically create named branch names that do not conflict with any existing named branch names in Dimensions, click the **Generate unique named branches** button. Unique branch names are entered into the fields.

28 Click **Next**.

29 Choose whether to migrate all file revisions from CVS (**Migrate all revisions**), or filter the revisions to migrate (**Filter revisions**). If you chose to select to filter revisions, you can select a maximum number of revisions to migrate for each file, as well as limit migration to revisions created after a particular date.

30 Click **Finish**.

Copying a Migration

To create a new migration definition based on an existing one, select the migration that you want to copy, and then click **Copy Migration**.

Enter a name for the new migration. Only key configuration files are copied when you copy a migration.

Running a Migration

When you are ready to run a migration, complete the following steps:

1 From the Migration Console, select the migration that you want to run and click Run Migration. The Execute Migration dialog box appears.

2 Before starting the migration, set any migration options correctly. These include:

- Choosing which of the migration stages you want to run. This includes the *transfer* stage, *generation* stage, *execution* stage, and *validation* stage. Please see "Choosing Migration Stages" on page 75.
- Options for the transfer stage, which is the portion of the migration during which files are transferred out of CVS and into a temporary folder location. Please see "Choosing Transfer Options" on page 76.
- Options for the execution stage, which is the portion of migration during which the CVS files and data are loaded into Dimensions. Please see "Choosing Execution Options" on page 76.
- Options for the validation stage, which is the portion of migration during which the migrated data in Dimensions is validated. "Choosing Validation Options" on page 77.

Choosing Migration Stages

When setting options from the Execute Migration dialog box, you can choose which migration stages you will run from the General tab. Consider the following:

Enable this stage...	If you want to...
Transfer	Copy all data out of the CVS repository and into the download directory that Dimensions will collect the files from. You only need to enable this stage if you have not already done so. If you have run this stage previously, then the files are already in the temporary area.
Generation	Generate the XML data files that control the Dimensions import. The generation stage is automatically enabled when either the execution or validation stage is enabled.

Enable this stage...	If you want to...
Execution	Move the data into Dimensions. This includes creating any baselines, projects, streams, and attributes that you defined when setting the migration options, as well as adding all files to Dimensions. You cannot complete this stage until you have completed the Transfer and Generation stages. Consider running this stage on its own if you have already completed the transfer stage but still need to import the data into Dimensions.
Validation	Verify that the data was all migrated correctly.

Choosing Transfer Options

When setting options for the Transfer stage from the Execute Migration dialog box, you can review the CVS server and repository information, and choose one of the following transfer types:

- **Extract All**: Choose this to transfer all information about the CVS data, including directories, files, tags, branches, and revisions. This option does not override file and revision content that has already been transfered.
- **Skip Existing**: Choose this to only transfer data out of CVS that has not been transferred previously. Choose this option if you are confident that the repository has not changed since the previous transfers, and you do not need to re-transfer any data about directories, files, tags, branches, or revisions.

Choosing Execution Options

From the Execution stage tab, optionally select the following options:

- **Create projects/streams**: Select this option to create projects and streams in Dimensions during the execution stage. Loading projects and streams can take some time; you can optionally de-select this option if you wish to save project and stream loading for later. In this case, only the mainline trunk project is created, however no branch projects or streams will be created. To load the projects or streams later, you can run Dimensions command-line scripts from the following directory:
  Migration\console\Migrations\<*migration name*>\import\migtmp
  From this directory, you can run a createWorkset_<number>.dmcli script for each project.
  If you want to load all projects and streams all at once at a later time, you can run the projects.cmd script located in the following directory:
  Migration\console\Migrations\<*migration name*>
- **Create baselines**: Select this option to create baselines from a script that is prepared during migration. Loading baselines can take some time; you can optionally de-select this option if you wish to save baseline loading for later. You can always re-run the execution stage and select this option then. To load the baselines later, you can also run Dimensions command-line scripts from the following directory:
  Migration\console\Migrations\<*migration name*>\import\migtmp
  From this directory, you can run a createBaseline_<number>.dmcli script for each baseline.
  If you want to load all baselines all at once at a later time, you can run the baselines.cmd script located in the following directory:
  Migration\console\Migrations\<*migration name*>

Choosing Validation Options

From the Execute Migration dialog box, you can choose the extent to which migrated data in Dimensions should be validated. From the Validation stage tab, choose from the following options:

- Select **Validate all file versions** to verify that the content of all migrated versions is the same in Dimensions as it is in CVS.
- Select **Validate tip versions only** to verify that the content of only the tip revision of every file is the same.

Starting Migration

Once you have set all of the options, click the **Start** button to start the migration.

NOTE Clicking the **Cancel** button may not immediately stop the migration process. Certain operations (specifically the import phase as well as trigger commands and baseline creation scripts) may continue until completed.

Using the Events Pane

As the migration proceeds and finishes, this pane contains error, warning, and informational messages.

Reviewing Migration Results

When a migration is completed, review the information in the status section of the Execute Migration window, verify that there were no unexpected warnings or errors during any of the migration phases (export, transform, import), and make sure that the expected data is in Dimensions. You can also click the **View Audit Report** button to display more detailed results information.

You can re-run the migration after fixing any errors.

Part 3

Migrating from Subversion

Chapter 10

About the Subversion Migration

Introduction

You can use the extended migration tools in Dimensions CM to migrate Subversion directories and objects to Dimensions.

The Migration Console is a graphical front end to a set of XML-based utilities that move data from supported sources into Dimensions products, allowing you to define how the data is mapped into Dimensions.

The Migration Console includes the following features:

- You can migrate all or part of a Subversion repository into a Dimensions CM project hierarchy.
- Migration preserves your Subversion repository structure.
- You can migrate your Subversion trunk, branches, and tags to separate projects and streams.
- You can migrate items to named branches in Dimensions.
- You can also optionally migrate tags to Dimensions baselines or attributes.

Data Mapping

The following table shows how the migration process transforms Subversion data into Dimensions data:

Subversion	Dimensions CM
Repository	Product
Users	Users
Files and revisions	Items and revisions; the original revision numbers from Subversion are migrated to a Dimensions attribute called SVN_REVISION
Directories	Folders
Tags	Baselines, attributes, projects, named branches
Branches	Projects, named branches

To learn more about the different types of Dimensions objects, such as products, projects, and design parts, please see *Getting Started with Dimensions CM*, available to download from the Serena Support Website.

What Data is Migrated?

The Migration Console migrates the following data from Subversion to Dimensions CM:

- All Subversion users.
- All directories, or a set of sub-directories, in a Subversion repository.
  - Subversion directories are migrated to folders in Dimensions.
  - Subversion revisions become item revisions.
- Individual revisions to a non-delta item library (File mode).
- The Subversion trunk is migrated to a project or stream.
- Subversion branches are migrated to projects or streams. You can also migrate items from the trunk or branch to named branches.
- Subversion tags are migrated to projects, baselines, an attribute called SVN_INFO, and named branches.
- CM changesets are created based on Subversion revisions.

What Data is Not Migrated?

The Migration Console does not migrate:

- User permissions and groups
- Multi-byte characters
- Local workspaces

Chapter 11

Preparing to Migrate from Subversion

System Requirements

Dimensions Requirements

To use the Migration Console with Subversion, you must:

- Install Dimensions 2009 R2 or later. For details, see the *Serena Dimensions CM Installation Guide* or the *Serena Dimensions CM Installation Guide*.
- Create and configure the target Dimensions product, including such configurations as the item library and upload rules. Please see the *Serena Process Configuration Guide*.
- If you will log in to your system as a different user than your Dimensions administration user (such as dmsys), add your system user to Dimensions with administrative privileges.
- If Single Sign-On (SSO) is configured for Dimensions, you must ensure that your operating system username match the SSO username and password.
- Run the Migration Console from an installation of the Dimensions Server. The Subversion client must also be installed to this system.
- Run the Migration Console from a system that will have at least 1 GB of RAM available to the Migration Console.
- Create the Dimensions product that will store the Subversion data you are migrating. You do not need to create the projects in advance.

Subversion Requirements

Subversion does not need to be installed to the migration system. Migration is performed against Subversion dump files. Please see "Preparing Subversion Folders and Files" on page 87.

IMPORTANT! Migrations must be performed on a local drive. This is the default configuration. If the download directory is on a network drive, migration will fail.

Storage Requirements

To ensure successful migration, follow these guidelines for disk storage:

- Make sure the location where temporary migration files are stored has at least twice the storage capacity of the size of the originating repository. For example, if you are migrating a repository that is 1 GB is size, make sure that the location for temporary files has at least 2 GB free. By default, this location is:
  c:\Documents and Settings\All Users\Application Data\Serena\Migrations
- Make sure that the location to which revisions are downloaded has space available that is at least equivalent to the size of the originating repository. By default, this is:
  C:\Documents and Settings\All Users\Application Data\Serena\Migrations\<*migration name*>\svnTemp\<*SVN server*>\<*SVN repository*>
  However you can customize this location during migration.
- Make sure that the Dimensions item library location has enough space to store all revisions to be migrated. If the originating repository is 1 GB and you are migration all revisions, make sure that at least 1 GB is available on the item library location.

Licensing Requirements

To run the Migration Console, you must have a valid Dimensions CM license.

Preparing Subversion Folders and Files

Before you migrate, perform the following steps:

- Make sure that there are no duplicate files or folders with the same name in the same directories in Subversion. **This includes files with the same name but with unique case**, for example source.xml and SOURCE.xml. In this example, one of the two file must be renamed. You must also ensure that no folders have the same names as files in the same directories.
- In unicode files that will be converted to ASCII format in Dimensions, you must convert end-of-line characters from UNIX-style LF to CR+LF. If you do not do this, migrated unicode ASCII files may not validate correctly.
- Make sure that any work in progress is completed in the repository to be migrated.
- Generate a dump of the Subversion repository. The Dimensions Migration Console requires the Subversion dump files, and cannot run a migration without them. **Please note** that if you will filter the dump files using the svndumpfilter utility with the `--drop-empty-revs option`, you must also use the `--renumber-revs` option to ensure that there are no gaps in the numeric revision sequence. If you do not, the migration console will return an exception due to missing revisions.

Creating Changesets from Subversion Revisions

By default Dimensions CM changesets are created from Subversion revisions. The changesets are for historical use only and cannot be used as stream versions, for example, to fetch stream sources from a date prior to a migration. Files deleted from Subversion are included in changeset history but are not migrated as Dimensions CM items.

Team Preparation Overview

When planning a migration, consider the following:

- **Plan and run a prototype.** Your test data should be representative of your production data in quantity and structure. If time permits, a trial conversion of your production data could be worthwhile. You can do this by creating a Dimensions base database just for testing the migration. For details on creating a base database using the create database (CRDB) command with the Dimensions DBA utility (dmdba), please see the *Serena Dimensions CM System Administration Guide*.
- **Prevent interruptions.** If the migration will run for a long time, take steps to ensure that no backups, planned network outages, software updates, or other interruptions occur.
- **Prevent log-ins.** If migrating into a live database with users, ask them not to log in during the migration. Anything they do will take resources away from the migration and could cause performance problems or other undesirable side effects.
- **Configure notification.** Be aware that if you migrate into a destination database with users, newly migrated objects might generate e-mail notifications. Test this behavior ahead of time, and take appropriate steps (for example, adjusting e-mail settings).

Migrating to UNIX and Linux Servers

Review this topic very carefully if you will migrate from Subversion to a Dimensions CM Server running on a non-Windows platform, such as supported UNIX and Linux platforms. Because the migration console is only supported on Windows, you must complete these steps in order to migrate to a remote Dimensions CM server from a Windows system.

Summary of Remote Migration Process

Stage	Description
1	If you have not already done so, install Dimensions and its repository to the target UNIX or Linux server. For details on installing and setting up the Dimensions CM server see the *Serena Dimensions CM Installation Guide* and *Serena Dimensions CM System Administration Guide*.
2	Identify a Windows system in the network that you can use to run the Migration Console. This system must meet the minimum system requirements for the Dimensions CM Server.
3	Install a remote DBMS client, such as Oracle SQL Net, to this Windows system. This is necessary so that you can then connect to the remote Dimensions CM database. For example, if you have purchased it, you can install the Serena Runtime to this system.
4	Install the Dimensions CM server, with the Migration Console and Common Tools options, to this Windows system. For a brief overview of the simplest Dimensions CM server installation scenario, please see the *Serena Dimensions Installation Quick Start Guide*.
6	Once installation to the Windows system is complete, you must complete the following steps from the Administration Console, to set up and configure network nodes. Please see the *Serena Dimensions CM System Administration Guide* for detailed information on managing network nodes. a Log into the Distributed Development | Network Administration view. b Select the **Network Nodes** tab. c Click the **New** button and choose **Physical Network Node**. The New / Edit Node dialog box appears. d Complete the dialog box to define this node as the remote Dimensions CM server to which you will migrate, including choosing the correct operating system from the **Operating System** list. e After you have defined the node, select it and then click the **+** icon under Base Databases to add a new base database to the network node. Configure this base database to point to the base database on the remote server. To define this, you must know the base database name and database connection string.

Stage	Description
7	On Oracle, you must now add the database connection as a local net service name. To do this: a If you installed the Serena Runtime, select Oracle - Dimensions | Configuration and Migration Tools | Net Configuration Assistant from the Windows Start menu. b On the wizard that appears, select the **Local Net Service Name configuration** option and click **Next**. c On the next page of the wizard, select the **Add** option and click **Next**. d Enter the database service name, such as DIM10, in the **Service Name** field and click **Next**. The service name is typically the same as the global database name. To determine the global database name on the Oracle server, contact your database administrator. e On the next page of the wizard, select **TCP** and click **Next**. f Enter the host name in the **Host name** field and choose the **Use the standard port number of 1521** option (unless the Oracle port number on the target server has been changed). Click Next. g When prompted to test the connection select the **Yes, perform a test** option and click **Next**. h When prompted, enter or confirm the net service name that the local database should use to interact with the remote database. In most cases, this should be the service name that you entered earlier, such as DIM10. i When you are asked to configure another net service name, choose **No**. Choose **Finish** when possible. j Reopen the Net Configuration Assistant. k Select the **Local Net Service Name configuration** option and click **Next**. l Select **Rename**. m Verify that the new connection appears in the drop-down list, and click **Cancel**.

Stage	Description
8	Before you can connect to the remote database from the Windows server, you must store your login information to a password storage file (registry.dat) on the Windows server. To do this, you must use the `dmpasswd` utility on the Windows server. The `dmpasswd` utility ensures that all Dimensions base database names, connection strings, and passwords are encrypted before they are written to disk. If the remote database server is Oracle, then the `dmpasswd` command will look something like the following: `dmpasswd <Base_DB>@<SID> -add -pwd <Password>` Where *Base_DB* is the base database name, *SID* is the Oracle system identifier, and *Password* is your login password. The base database name typically corresponds to your login name for oracle databases. For detailed information on the `dmpasswd` utility, including the correct command format for other DBMS platforms as well as additional options, please consult the *Serena Dimensions CM Command Line Reference*.

9	Test the database connection by making sure that the `pdiff` utility can connect to the base database from the Windows server. `pdiff` is a tool for importing and exporting data to and from a Dimensions CM product, using PDIFF files.
	Before you can test this, you must set the DMDB environment variable to the following value:
	`<Base_DB_name>@<DB_connection>`
	For example, if the base DB is named qlarius_cm and the DB connection is dim10, use the following command to set the environment variable:
	`set DMDB=qlarius_cm@dim10`
	For details on the DMDB environment variable, please see the *Serena Dimensions CM Adminstrator's Guide*.
	Then, to test the connection, you will run a command like the following from the command line, that will dump a partial process model from a product into a PDIFF file:
	`pdiff <product_id> -dump_cpl`
	Where *product_id* is the ID of the product from which you will dump the process model. For example, if you installed the Qlarius sample process model to the Dimensions CM server, the command would appear as follows:
	`pdiff qlarius_cm -dump_cpl`
	The command succeeds if the connection is made.
	For details on the `pdiff` utility and the `dump_cpl` function, please see the *Serena Dimensions CM System Administration Guide*.

Having completed the above steps, you can now set up the migration as you would for any other scenario.

Before you Migrate

Before you start a migration:

- To avoid connection time-outs when migrating, temporarily increase these values in `dfs/listener.dat` to:
  - `session_timeout: 259200`
  - `idle_timeout: 3600`
- Check that:
  - Upload Rules are configured for the migrated file types.
  - The target repository does not contain IDs that overlap with the migrated object IDs (streams, projects, baselines, and branches).
  - Oracle has sufficient memory available and that the limits in the SGA/PGA configuration are not too small.
  - The Dimensions CM Application Server and Migration Console have sufficient memory.

Chapter 12

Migrating from Subversion

Using the Migration Console

When you first run the Migration Console, you see a list of predefined migrations. You can sort this list by clicking on a column heading.

To define a new migration, click **New Migration**. To copy, edit, or run an existing migration, select the migration in the list, and then click the appropriate button.

NOTES

- You can click the **Delete Migration** button to remove a migration from the list, however this does not actually delete the migration definition. Doing so could delete migration data that you actually need.
- You can also optionally delete all migration data produced during the migration process.
- At any time when you are creating or editing a migration, you can click **OK** to save all changes and close the designer, **Cancel** to close the designer and receive a prompt asking whether you want to save changes, or **Apply** to save changes while leaving the designer open.

Migrating from Subversion

1. Click **New Migration** to define a new migration or select an existing migration and click **Edit Migration**.
2. On the **Welcome** screen enter a name for the migration in the **Migration ID** field. Select **SVN to Dimensions**.

   Click **Next**.
3. On the **Dimensions Connection** screen specify the Dimensions server connection information:

   a (Optional) From the **Profile** list select a an existing Dimensions profile.

   b In the **System** area enter the user name and password of the Dimensions CM account to be used by the Migration Console to login and retrieve data.

   c In the **Dimensions** area enter the Dimensions CM server name, database name, and database connection string.

   d Click **Test Connection** to check the connection.

   NOTE: The Dimensions CM server must be installed locally. If it is on a UNIX or Linux server see "Summary of Remote Migration Process" on page 89.

   Click **Next**.

4 On the **Dimensions Destination** screen specify where the migrate asses will be stored in Dimensions.

In the **Specify Dimensions destination to migrate into** area do the following:

a From the **Product ID** list select an existing Dimensions product.

b From the **Design Part** list select a design part or create a new one. If you create a new design part assign a design category and parent part.

In the **Specify where the trunk revisions are to be migrated to** area enter a name for the target stream or project. If you are migrating to a project choose the Dimensions project type.

In the **Specify what should we do with long comments** area select one of these options:

- Truncate comments over 1978 characters.
- Store all comments in Dimensions items. Select one of these options:
  - The Dimensions item type for the stored comments.
  - The relationship between the comment items and the items that they are associated with. Choose an existing relationship type or enter a new one.

Click **Next**.

5 On the **Dimensions Named Branches** screen specify how migrated items will be created in Dimensions by entering named branches:

- **Named branch for trunk items**

  All items from the trunk will have the following item specification:

  `<ItemID>.A-<ItemType>;<NamedBranch>#<Revision>`

  where `NamedBranch` is the branch name you entered.

- **Items in Subversion branches**

  All items from branches will have the following item specification:

  `<ItemID>.A-<ItemType>;<NamedDimBranch><SVNBranch>#<Revision>`

  where:

  - `NamedDimBranch` is the branch name you entered.
  - `SVNBranch` is the name of the originating Subversion branch.

- **Named branch prefix for tag items**

  All items from tags will have the following item specification:

  `<ItemID>.A-<ItemType>;<NamedBranch><SVNtag>#<Revision>`

  where:

  - `NamedBranch` is the branch name you entered.
  - `SVNtag` is the name of the originating Subversion tag.

TIP To automatically create named branch names that do not conflict with any existing named branch names in Dimensions, click **Generate unique named branches**.

Click **Next**.

6 On the **Subversion Repository** screen enter the path to the Subversion dump file that will be migrated or click **Choose File** to browse for one.

Click **Validate dump file** to check that the dump file is at the specified path and is valid. Depending on the size of the Subversion repository the verification may take from several minutes up to half an hour.

Select a temporary directory to use during the migration.

Click **Next**.

7 On the **Subversion Migration Details** screen specify how to migrate the Subversion repository. Select one of the following:

- Migrate the entire Subversion repository into the specified Dimensions project or stream.
- Migrate the Subversion trunk, branches, and tags into corresponding streams in Dimensions.

By default the option **Create changesets based on Subversion revisions** is selected.

Click **Next**.

8 On the next **Subversion Migration Details** screen specify the structure in the Subversion repository to be migrated:

- Specify the directory in Subversion that contains the trunk. The trunk will be migrated into the specified Dimensions product and stream/project.
- Specify additional directories in which to search for branches. Enter a comma-separated list of directories, for example:

  `branches/releases;branches/user;branches/test`

  searches in:

  - `branches/release`
  - `branches/user`
  - `branches/test`

  Branches will be migrated into the following Dimensions stream or project:

  `<product>:<stream/project>_BRANCH_<branch name>`
- Specify additional directories in which to search for tags. Enter a comma-separated list of directories. Tags will be migrated into the following Dimensions stream or baseline:

  `<product>:<stream/project>_TAG_<tag name>`

  - Choose whether to migrate tags as Dimensions baselines or streams. If you choose baselines only one revision of every file will be migrated. To preserve all revisions migrate to a stream.
  - Optionally migrate tags into an attribute in Dimensions called `SVN_INFO`.

9 Click **Finish**.

Copying a Migration

To create a new migration definition based on an existing one, select the migration that you want to copy, and then click **Copy Migration**.

Enter a name for the new migration. Only key configuration files are copied when you copy a migration.

Running a Migration

When you are ready to run a migration, complete the following steps:

1. From the Migration Console, select the migration that you want to run and click **Run Migration**. The Execute Migration dialog box appears.
2. Before starting the migration, set any migration options correctly. These include:
   - Choosing which of the migration stages you want to run. This includes the *generation* stage, *execution* stage, and *validation* stage. Please see "Choosing Migration Stages" on page 97.
   - Options for the execution stage, which is the portion of migration during which the Subversion files and data are loaded into Dimensions. Please see "Choosing Execution Options" on page 98.
   - Options for the validation stage, which is the portion of migration during which the migrated data in Dimensions is validated. "Choosing Validation Options" on page 99.
3. Then you can start and monitor the migration. See "Starting Migration" on page 99 and "Using the Events Pane" on page 99.

Choosing Migration Stages

When setting options from the Execute Migration dialog box, you can choose which migration stages you will run from the General tab.

For very large migrations, Serena recommends performing each stage separately.

Consider the following:

Enable this stage...	If you want to...
Generation	Generate the XML data files that control the Dimensions import. This stage may take several hours, depending on the size of the Subversion repository.
Execution	Move the data into Dimensions. This includes creating any baselines, projects, streams, and attributes that you defined when setting the migration options, as well as adding all files to Dimensions. You cannot complete this stage until you have completed the Generation stage. Consider running this stage on its own if you have already completed the transfer stage but still need to import the data into Dimensions. Note the following: ■ You must make sure that the Dimensions listener service is running, as it is required for this migration stage. ■ Depending on the size of the repository you are migrating, this stage may require between 12 and 36 hours. This includes import the data into Dimensions, and then creating all projects, streams, and baselines.
Validation	Verify that the data was all migrated correctly.

Choosing Execution Options

From the Execution stage tab, optionally select the following options:

- **Create Projects/Streams**: Select this option to create projects or streams in Dimensions during the execution stage. Loading projects can take some time; you can optionally de-select this option if you wish to save project and stream loading for later. In this case, only the mainline trunk project is created, however no branch projects or streams will be created. To load the projects or streams later, you can run Dimensions command-line scripts from the following directory:
  Migration\console\Migrations\<*migration name*>\import\migtmp
  From here, you can run a createWorkset_<number>.dmcli script for each project.
  If you want to load all projects all at once at a later time, you can run the projects.cmd script located in the following directory:
  Migration\console\Migrations\<*migration name*>
- **Create Baselines**: Select this option to create baselines from a script that is prepared during migration. Loading baselines can take some time; you can optionally de-select this option if you wish to save baseline loading for later. You can always re-run the execution stage and select this option then. To load the baselines later, you can also run Dimensions command-line scripts from the following directory:
  Migration\console\Migrations\<*migration name*>\import\migtmp
  From this directory, you can run a createBaseline_<number>.dmcli script for each baseline.
  If you want to load all baselines all at once at a later time, you can run the baselines.cmd script located in the following directory:
  Migration\console\Migrations\<*migration name*>

Choosing Validation Options

From the Execute Migration dialog box, you can choose the extent to which migrated data in Dimensions should be validated. From the Validation stage tab, choose from the following options:

- Select **Validate content of all file versions** to verify that the actual content in the file revisions is the same.
- Select the **Validate content only of tip versions** checkbox to only validate the size or content of the last revision of every file.
- Select the **Don't validate content at all** option to skip the validation stage. Validation can be time-consuming; this option can save time.

Starting Migration

Once you have set all of the options, click the **Start** button to start the migration.

NOTE Clicking the **Cancel** button may not immediately stop the migration process. Certain operations (specifically the import phase as well as trigger commands and baseline creation scripts) may continue until completed.

Using the Events Pane

As the migration proceeds and finishes, this pane contains error, warning, and informational messages.

Reviewing Migration Results

When a migration is completed, review the information in the status section of the Execute Migration window, verify that there were no unexpected warnings or errors during any of the migration phases (export, transform, import), and make sure that the expected data is in Dimensions. You can also click the **View Audit Report** button to display more detailed results information.

You can re-run the migration after fixing any errors.

Part 4

Migrating from ClearCase

Chapter 13

About the ClearCase Migration

Introduction

The ClearCase migration utility facilitates rapid migration of ClearCase data to Dimensions CM. The ClearCase migration utility is highly configurable. It is driven by command scripts, allowing you to migrate any number of projects into Dimensions using the same configurations.

You can migrate one or many ClearCase projects to a single Dimensions product or multiple distinct products. Any mix of Projects and Products is possible.

You can also migrate users and groups to Dimensions users and roles. You can map user permissions for ClearCase projects to role assignments on specific products or design parts.

Migration Utility Components

The migration utility consists of a set of individual components that work together through the use of XML-based export and import formats.

- **ClearCase export (ccexport)** - Reads a file (*cvt_data*) that is produced when you run the ClearCase `clearexport_ccase` command, and produces an XML file that the migration utilities can then use. The ccexport utility also produces a script to check out all versions from ClearCase.
- **Transform (transform)** - Processes XML input files, and applies a set of transformation rules to produce a DimensionsXML result file.
- **Dimensions Import (dmimport)** - Imports DimensionsXML into Dimensions populating a Dimensions base database (which may be initially empty) with Dimensions objects. Process Model data is created (Products, Types, Users, Roles, Role Assignments, Attributes, Valid Sets, etc.) as well as Change Documents and Items.

Usage Scenario

The following suggests a workflow to follow when using the ClearCase migration utility. Review and consider these recommendations before you get started planning and running your migration.

1 **Installation and Familiarization.** Install the migration utilities. Read the documentation provided with the migration utilities.

2 **Development.** Define the migration model through iterative export, transformation, and import tests. Construct the configuration files and scripts that will drive the migration tools for the projects you are migrating.

   a Perform one or more exports of existing data using CcExport. Follow the pre-migration checklists specific to each type of export. Start with small subsets of data, work with large data sets once the migration model is fine tuned and working smoothly.

b Create a transformation configuration (Transform.xml) for the migration. Start simple, and eventually make appropriate use of the various transformation strategies (ItemStrategy, ChgDocStrategy, BaselineStrategy, UserAndGroupStrategy).

c Import into a Dimensions test product. Confirm that the data is migrating to Dimensions using the desired mapping. Work with stakeholders to validate the various usage scenarios in Dimensions.

d Repeat the above steps as needed, deleting the Dimensions test product and re-importing, while evolving the transformation configuration until the desired mapping is achieved.

3 Test Migration.

a Perform the steps outlined in step 2 using your full data set.

b Measure the time it takes to complete the migration.

c Scrutinize error logs and audit files.

d Perform a full validation of the data in Dimensions CM using the provided validation scripts.

4 Production Migration.

a Plan the production migration event with project stakeholders.

b Lock users out of the source projects in order to prevent further development on the projects to be migrate.

c Perform the migration.

d Preserve the migration scripts and audit logs as a permanent record of what was migrated.

Chapter 14

Preparing to Migrate from ClearCase

System Requirements

Windows System Requirements

The following are the system requirements for the system to which you will install the migration utilities:

- Microsoft Windows XP Professional, Windows Server 2008, or Windows Server 2003.
- 256Mb of RAM
- 40mb of disk space for installation

UNIX System Requirements

See the Dimensions CM readme for a complete list of supported UNIX platforms for Dimensions

Supported Versions of ClearCase

Migration from ClearCase 4.x - 7.x is supported.

Supported Versions of Dimensions CM

Migration is supported to the following Dimensions CM releases:

- Dimensions CM 2009 R2

Prequisites for the Dimensions CM Import Utility (DmImport)

Dimensions CM must be installed on the system from which you will perform the import migration step. When installing Dimensions CM you must make sure to install the Migration Console and server components to this system.

Installing the Migration Utilities

The migration utility software is installed along with the Dimensions CM server.

What Is Installed?

The Migration directory should now contain the following, among other files:

- Transform.cmd - Transform

- DmImport.cmd - Dimensions Import
- setupenv - sets the environment for UNIX
- ccexport -ClearCase Export

Setting Up the Environment

Before you can run the migration utilities, you must set up some environment variables. Scripts are provided to do this.

Windows

On Windows, there are two setup scripts that you need to run, depending whether you use the Thompson Toolkit command-line tools:

- Configure and run the SetupEnv.cmd script before running any migration utilities from the Windows command-line.
- Optionally configure and run the SetupEnviron.sh script before using the Thompson Toolkit to run the migration utilities.

To set up a Windows environment:

1. Open the Migration\bin\w32\SetupEnv.cmd file (or SetupEnviron.sh if you are using the Thompson Toolkit) in a text editor and set these variables:
   - `MG_ROOT`: The root Migration installation directory. This is the parent directory of bin and lib.
   - `DM_ROOT`: The root Dimensions installation directory.

     Note that the `MG_ROOT` and `DM_ROOT` variables can also be set in the environment. The SetupEnv scripts will respect the environment values. Also note that the PATH for Dimensions is not set in the SetupEnv* scripts. This should be set in the environment's PATH variable.
2. From a command prompt run:
   `SetupEnv.cmd`.
3. Or, if you are using a Thompson Toolkit shell, source the SetupEnviron.sh script:
   `. SetupEnviron.sh`

UNIX

The `setupenv` script is used for UNIX command line sessions.

To set up a UNIX environment:

1. Edit the Migration/bin/unix/setupenv script in a text editor and modify the default values for the variables, or set the values as environment variables. The script will respect the environment values if they are defined. The variables are:
   - `MG_ROOT`: The root Migration installation directory. Set `MG_ROOT` to the absolute path to the directory where you installed the Migration.

- DM_ROOT: The root Dimensions installation directory. Set DM_ROOT to the absolute path to the parent directory of prog and mmi in your Dimensions installation. If DmImport will not be run on this platform and Dimensions is not installed, set the DM_ROOT variable to none:
  DM_ROOT=none

2 Source the setupenv script. If MG_ROOT is set in your environment, run the following command:

```
$ . $MG_ROOT/bin/unix/setupenv
```

or from the parent directory of the /Migration directory:

```
$ cd Migration/bin/unix
$ .  ./setupenv
```

Configuring Dimensions.properties

Before running a Dimensions import, you must configure a Dimensions.properties file. To do this, run the setupDmProp command from a command-line prompt.

Validating the Installation

Once the migration utility is configured, you can run migration commands directly from the command-line. Validate that the utilities (ccexport, transform, and dmimport) are installed correctly and ready to use by running test commands.

Windows

1 To validate that the utilities are installed correctly, run them from a command-line prompt. For example, run the transform command from the following location, when the migration utilities are installed to C:\dimensions:
c:\dimensions\migration\bin\w32\transform

The following text should appear in the command window:

```
C:\>java merant.adm.migration.transform.Transform
PVCS Migration Utility (transform) 1.1.0 (Build (tip)).
Copyright 2001-2002 MERANT.  All rights reserved.
Usage:  transform transformConfigurationFile ...
```

When you execute a migration utility command it automatically calls SetupEnv.cmd, if it has not already been called. The SetupEnv.cmd sets the MG_ROOT, DM_ROOT, PATH and CLASSPATH environment variables. The result is that once the environment is loaded, you can invoke the commands directly, for example:

C:\>transform

2 To validate that the ccexport utility is installed correctly, run the ccexport command from a command-line prompt:

```
C:\>ccexport
```

The following text should appear in the command window:

```
C:\>java merant.adm.migration.cc2xml.Cc2Xml
PVCS Migration Utility (cc2xml) 1.1.0 (Build (tip)).
```

```
Copyright 2001-2002 MERANT.  All rights reserved.
ERROR:  Unable to open cvt_data (The system cannot find the file
    specified)
```

UNIX

1 To validate that the utilities are installed correctly, run them from a command-line prompt. For example, enter the following to run the `transform` command. The command below assumes that you have already sourced `setupenv`:

```
$ transform
```

The following text should appear in the command window:

```
+ java merant.adm.migration.transform.Transform
PVCS Migration Utility (transform) 1.1.0 (Build B147).
Copyright 2001-2002 MERANT.  All rights reserved.
Usage:  transform transformConfigurationFile ...
```

2 To validate that the ccexport utility is installed correctly, run the `ccexport` command from a command-line prompt:

```
$ ccexport
```

The following text should appear in the command window:

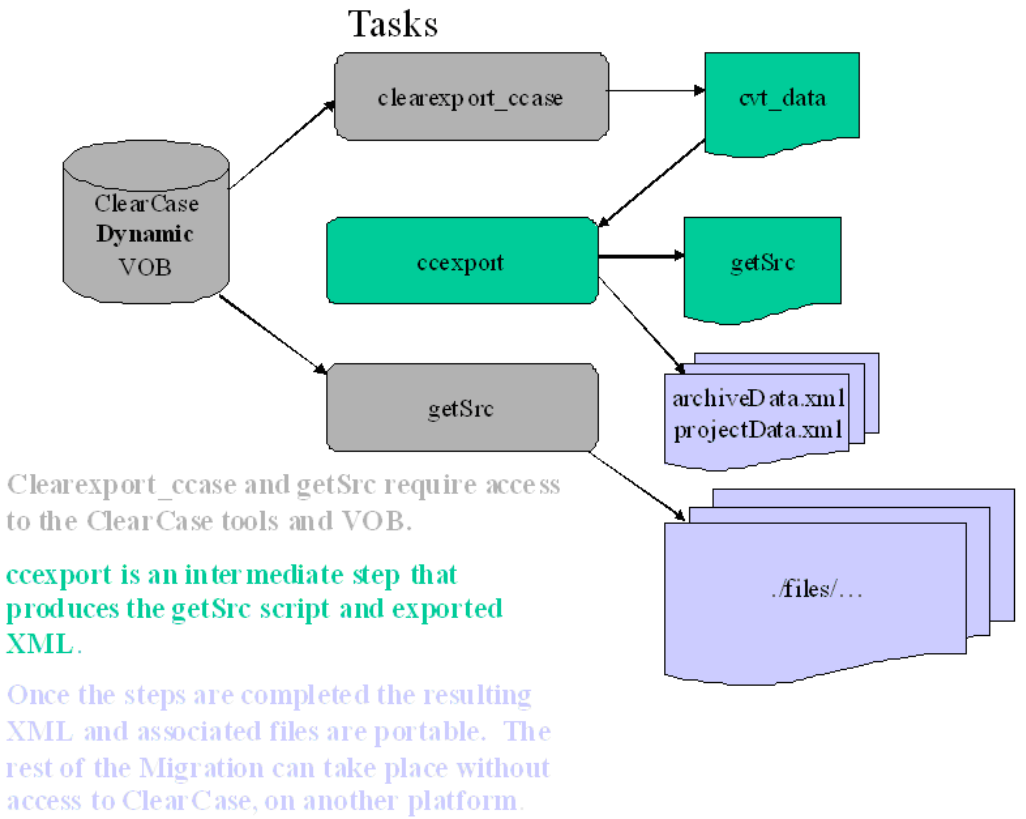
```
+ java merant.adm.migration.cc2xml.Cc2Xml
PVCS Migration Utility (cc2xml) 1.1.0 (Build B147).
Copyright 2001-2002 MERANT.  All rights reserved.
ERROR:  Unable to open cvt_data (No such file or directory)
```

Chapter 15

Migrating from ClearCase

ClearCase Export Overview

The diagram below shows the three steps that are part of the ClearCase export process.



The steps are as follows:

1. Export the data from ClearCase
2. Run the intermediate export step (ccexport) on the same system from which the initial ClearCase export was run
3. Run the data transformation
4. Import the data to Dimensions CM

Once the XML and associated files have been produced by the ClearCase export process (steps 1 and 2 above), the data is portable to another system. This allows the ClearCase export to be performed on a server that has ClearCase access but that does not have Dimensions access or software. The data can be moved to a system where the transformation and import stages of the migration can be completed.

Creating the ClearCase Output

Complete the following procedures before starting the import to Dimensions CM.

Ensuring ClearCase Access

Obtain the necessary access to ClearCase software and ClearCase data. You (or someone with ClearCase access) will need to run the ClearCase Export process on a machine that has a licensed copy of ClearCase installed (`clearexport_ccase`) as well as access to the dynamic view(s) to be exported. Note that you cannot export a snapshot view. The view must be dynamic.

There are two steps in the process that require access: `clearexport_ccase`, and `getSrc`.

If `clearexport_ccase` is performed on a Windows platform, then `ccexport` and `getSrc` should be run on a Windows platform. If `clearexport_ccase` is performed on a UNIX platform, then `ccexport` and `getSrc` should be run on a UNIX platform.

Generating a cvt_data file With clearexport_ccase

The first step is to export data from a ClearCase dynamic versioned object base (VOB) to a `cvt_data` file in a ClearCase data format. This step requires access to a licensed copy of ClearCase. It invokes the `clearexport_ccase` command.

Use ClearCase to produce the 'cvt_data' export file. This is a manual step to be performed using ClearCase. Make an empty working directory, cd there, and run the export command:

To generate a cvt_data file using the clearexport_ccase command:

1. From a command prompt, create an empty directory, for example called ccdata:
   `mkdir ccdata`
2. Go to the new directory:
   `cd ccdata`
3. From this directory, run the clearexport_ccase command and specify the VOB that you want to export:
   `clearexport_ccase -r /some/clearcase/dynamic/view`

There should now be a 'cvt_data' file in the working directory.

Alternatively, with the following sequence of commands, you can go to the view directory and create the cvt_data there. Then, move cvt_data to an empty work directory:

```
cd /some/clearcase/dynamic/view
clearexport_ccase -r .
mv cvt_data /my/work/area/mydata
```

Running the ccexport Command

This step:

- Converts the cvt_data file into XML formats that the transformation and import tools can use. The files are *archiveData.xml* and *projectData.xml*.
- Generates a *getSrc.cmd* file (Windows) or *getSrc* (UNIX) script. This file is used to extract files from ClearCase into a './files' subdirectory. These ClearCase files are referenced by the `<sourcePath>` elements the archiveData.xml file.
- Does not require access to ClearCase software or data

NOTE If you run `clearexport_ccase` on a Windows platform, then you should also run ccexport and getSrc on a Windows platform. If you run `clearexport_ccase` on a UNIX platform, then you should also run `ccexport` and `getSrc` on a UNIX platform.

To run ccexport:

1 From a command prompt, enter `ccexport`.

Messages such as the following should appear in the command prompt:

```
+ java merant.adm.migration.cc2xml.cc2xml
PVCS Migration Utility (cc2xml) 1.1.0 (Build B147).
Copyright 2001-2002 MERANT.  All rights reserved.
Processing /nts2-1/dv/migration/suite/CcExport/Qwest1/out/cvt_data...
...
Processing Complete.
...
Writing 'projectData.xml'...
...
Starting to output archiveData.xml
```

Running the getSrc Command

This step:

- Gets file content from ClearCase and stores it in the file system in the ./files subdirectory. It extracts every version of every ClearCase item to a file.
- Requires access to a licensed copy of ClearCase as it invokes the cleartool `get` command

To run the getSrc command:

1 Run getSrc from a command prompt. This command file was generated when you ran the ccexport command.

Transforming the Exported Data

See "Transform - transform" on page 121 for detailed information about usage and configuration. Make sure you have successfully completed the export from ClearCase, and have the exported data available.

The transform command transforms the exported data into a format that Dimensions CM can import. Configuration is defined in the Transform.xml file. Start with the simplest

possible Transform.xml file and test the transformation, then iterate the transformation until the result is correct. A sample is provided with the installation: <MG_ROOT>/Migration/docs/TransformSimple.xml. This sample has an empty ChgDocStrategy, empty ItemStrategy, and a UserAndGroupStrategy that turns off migration of users. This saves time during import and is useful for test iterations.

Consider the following when defining the basic parameters in the Transform.xml file:

- `ItemStrategy`
  - What item types, formats, lifecycles, and status do you want to assign using `DefaultContent`?
  - What files should be excluded?
  - What design part and project / stream path do you want to configure with `DesignPartLevel` and WorksetPathLevel?
- `BaselineStrategy`
  - What labels do you want to preserve in Dimensions? You can turn them into any mix of attributes, projects / streams, and baselines.
- `UserAndGroupStrategy`
  - Do you want to migrate user lists from ClearCase into Dimensions CM?
  - Do you want project level access rights to be converted into role assignments?

Importing to Dimensions CM

Complete the following procedures to import your exported ClearCase data to Dimensions CM:

"Configuring Upload Rules" on page 117

"Configuring the Tool Manager" on page 118

"Verifying Item Type Options" on page 118

"Verifying Attribute Options" on page 118

"Importing to Dimensions CM Using Dimensions Import" on page 118

Configuring Upload Rules

Before importing data into Dimensions, configure the Project Merge Tool upload rules in the Dimensions CM Administration Console, as follows:

1. In the Administration Console, display the Configuration Object Management | Upload Rules view. For more information on defining the default upload rules, please see the *Process Configuration Guide*.
2. Modify the rules for the Project Merge Tool from the default rules for IDEs.
3. Determine if `%GENBINARYDAT` is already defined for the Project Merge Tool default rules. If so, you do not need to continue any further. If not, continue to the next step.

4 Add a new row to the top of the Upload Rules:
```
Pattern MatchItem FormatItem Type
%GNENBINRYDAT
```

You may need to delete the existing top rule and re-add it below the new rule.

Configuring the Tool Manager

You must make that your User ID exists in Dimensions as a Tool Manager. The user ID should also be granted the WORKSET-MANAGER role on the $GLOBAL:$GENERIC workset. For details on adding an configuring users, please see the *Process Configuration Guide*.

Verifying Item Type Options

If you will import to a pre-existing process model with pre-defined item types, you must disable the the option to automatically generate an item type identifier if one is not provided. See the *Dimensions CM Process Configuration Guide* for details.

Verifying Attribute Options

When importing into an existing process model, it is recommended that the option to set all revisions of an item to the same value be set to **No** for attributes which will be used to store labels.

By default the migration will create a multi-value VCS_TAGS attribute that stores all labels. When DmImport creates attribute it sets this option to No.

Importing to Dimensions CM Using Dimensions Import

To import to Dimensions CM, you must use the Dimensions Import command-line utility.

To import to Dimensions CM:

1 Open a shell or command prompt window.

2 Enter:
`dmimport` *`DmData.xml Dmimport.cfg`*

Where *`DmData.xml`* specifies the location of the XML file you want to import, and *`Dmimport.cfg`* specifies the location of the DmImport configuration file to use.

NOTE If you have problems, make sure that the location of the Dimensions Import utility is in your path, or invoke it from the installation location.

The parameters for this process are specified in the configuration file, which is described in detail here: "Dimensions Import - DmImport" on page 132.

Troubleshooting

Consult the following to help troubleshoot migration:

Troubleshooting ClearCase Export

WARNING: you are running CcExport on Windows against a cvt_data file created on Unix

The `clearexport_ccase` command produces a cvt_data export file. If the cvt_data file came from a UNIX system, then ccexport must be run on UNIX. If the cvt_data file came from a Windows system, the then ccexport must be executed on Windows.

WARNING: The login 'ruser1' is used with multiple user names: 'John Doe1' and 'Johnny Doe1'

ClearCase associates a user ID (Eventuser) and user name (EventName) with each version, in the exported cvt_data file. When ccexport processes that file, if it encounters a later occurrence of a user ID which has a user name that does not match prior occurrences, it is flagged with the above error. The later occurrences with different user names are ignored. In the above example, the user created in Dimensions will have the user ID `ruser1` and name `John Doe1`. The username 'Johnny Doe1' is ignored.

Troubleshooting Dimensions Import

PDIFF

PDIFF uses a local connection to the Dimensions server via in order to provide the best performance when importing large quantities of data. Local connections can only be established using the actual logged in Windows user.

Unable to create the log file: java.io.IOException

If you see this error:

```
Unable to create the log file: java.io.IOException: The system cannot
find the path specified.  Check settings in configuration file.
```

Set the `TEMP_DIRECTORY` value in the DmImport.cfg file to a valid existing directory.

Error: rulebase cannot upload file <filename>

If this error occurs confirm that the the default rules have been set up correct for IDEs. See "Configuring Upload Rules" on page 117.

java.lang.NoClassDefFoundError: com/ibm/xml/parsers/ NonValidatingDOMParser

The following error occurs during DmImport:

```
Exception in thread "main" java.lang.NoClassDefFoundError:  com/ibm/xml/
parsers/NonValidatingDOMParser
```

When the migration configuration in the Dimensions xml4j_2_0_11.jar file is not set correctly in the CLASSPATH. This is a .jar file supplied with Dimensions. The migration process uses this via a CLASSPATH defined in the SetupEnv.cmd or setupEnviron scripts. See "Setting Up the Environment" on page 109.

java.lang.NoClassDefFoundError: oracle/jdbc/driver/OracleDriver

When running DmImport if you encounter the following error message:

```
Connecting to server...Exception in thread "main"
java.lang.NoClassDefFoundError

: oracle/jdbc/driver/OracleDriver
```

Confirm that classes12.zip is in your CLASSPATH, and confirm that you performed all of the installation and configuration steps.

Connecting to server...java.lang.NullPointerException at merant.adm.dimensions.util.Options.loadProperties(Options.java:84)

When running DmImport if you encounter the following error message:

```
Connecting to server...java.lang.NullPointerException

        at
merant.adm.dimensions.util.Options.loadProperties(Options.java:84)
```

Confirm that you performed the installation and configuration steps, by running setupDmProp. Also confirm that the Dimensions.properties file is in the appropriate directory (<MigrationInstallationDirectory>/migration/classes/merant/adm/dimensions) and that CLASSPATH includes the migration/classes directory.

See "Configuring Dimensions.properties" on page 110.

Item Id "" already exists - renamed to "2"

Notice the empty quoted string and the new item name which contains only numeric digits. If DmImport fails with errors similar to the above confirm that you performed the Check Item Type Options task in the Pre-DmImport Checklist. This error may occur when migrating into an existing process model if the specified Item Type option is enabled.

An error has occurred: java.lang.reflect.InvocationTargetException

Under certain circumstances DmImport may fail to connect to the server with this message:

```
Connecting to server...An error has occurred:
java.lang.reflect.InvocationTargetException
```

This may occur when the $GENERIC$GLOBAL project is chosen as the default but no project directory has yet been created for it. To work around this issue, open the desktop client and set a different project as the default.

Command Reference

See the following for detailed guidance on running the commands requried to migrate to Dimensions CM:

"ClearCase Export - ccexport" on page 121

"Transform - transform" on page 121

"Dimensions Import - DmImport" on page 132

ClearCase Export - ccexport

The `ccexport` command looks in the current directory for a file named cvt_data. This file is converted into an XML format that the Dimensions import tools can use.

Syntax

```
ccexport
```

Options

- There are no command line flags.
- There is no configuration file.
- See "Running the ccexport Command" on page 116 for detailed usage.

Transform - transform

The Transformation Process

The transformation process uses the export files from ClearCase along with the transformation configuration file (Transform.xml) in order to convert the data to an XML file that can be imported into Dimensions.

The Transform.xml file determines how your data will be migrated into Dimensions. This file can get very complex. You can use an XML editor to simplify your view of this file and make it easier to work with.

Invoking the Transform Utility

To invoke the Transform ulitiy:

1. Open a command prompt.
2. Enter:
   `transform Transform.xml`

NOTE If you have problems, verify that the Transform utility is in your path, or invoke it from the installation location.

Syntax Overview

Two TransformXML files are provided in the migration/bin/docs. Transform.xml is the file that you must edit to configure a transformation. TransformSimple.xml is a basic sample file that can be used as a basis for starting a migration. Any migration can be started with this file, then gradually evolved until the required migration specification is in place.

The transform configuration file is an XML file, and its root element is TransformXML. Within this element, there are several sub-elements, as shown below.

```
<TransformXML>
   <ProjectMap> . . . </ProjectMap>
   <ChgDocStrategy> . . . </ChgDocStrategy>
   <ItemStrategy> . . . </ItemStrategy>
   <RuleSet> . . . </RuleSet>
   <UserAndGroupStrategy> . . . </UserAndGroupStrategy>
   <BaselineStrategy> . . . </BaselineStrategy>
</TransformXML>
```

Each element in TransformXML is described in the following sections. Which elements you use will depend on what kind of data you want to transform. Many elements that specify default values may appear in <TransformXML>, <ProjectMap>, and <SourceProject>. The nearest default is used for any given context. When the elements are used in SourceProject, they are the default for that project. When they are used in the ProjectMap, they are the default for any SourceProject within that ProjectMap that does not have its own default. When they are used in TransformXML they are the default for any ProjectMap that does not have a more explicit default. For each element described below, the allowable contexts are specified.

<ProjectMap> Element

The `<ProjectMap>` element is the root of the transformation process, specifying one or more source projects and a destination Dimensions product. Transform.XML must contain at least one `ProjectMap` element, and may contain many. For each `ProjectMap` specified within Transform.XML, a set of source projects are transformed into DimensionsXML and a resulting destination XML file is produced. Each `ProjectMap` must produce a separate output file.

The transform process iterates through each `ProjectMap` element. It can be used to manage different migration scenarios, such as:

- When `ProjectMap` specifies a single source project and a single destination product, the source is migrated to the named destination product.
- When `ProjectMap` specifies a set of source projects and a destination product, all source projects are migrated to a single product.
- When a `ProjectMap` specifies a set of source projects and no destination product, each source project is migrated to a Dimensions product with a name that is derived from the source project name.

Example syntax:

NOTE This example shows all of the possible elements; your file may not be this complex.

```
<ProjectMap>
   <DestXmlFile>Dimensions_data_destination_file.xml</DestXmlFile>
   <DestProduct>name_of_destination_Dimensions_product</DestProduct>
   <DestPart>name_of_default_destination_design_part</DestPart>
   <DestParentPart>name_of_destination_parent_part</DestParentPart>
   <DestWorkset>name_of_default_destination_project</DestWorkset>
   <DestVariant>product_variant</DestVariant>
   <DestProductManager>product_manager</DestProductManager>
   <ItemLibrary>item_library_path</ItemLibrary>
   <ChgDocTypeForLongData>request_type</ChgDocTypeForLongData>
   <DestLifecycleID>destination_lifecycle_Id</DestLifecycleID>
   <DestStatus>destination_status</DestStatus>
   <PartCategory>part_category_for_design_parts</PartCategory>
   <SourceProject>
      <SourceXMLFile>source_xml_file.xml</SourceXMLFile>
      <SourceProjectType>source_project_type</SourceProjectType>
      <SourceProjectName>source_project_name</SourceProjectName>
   </SourceProject>
   <SourceProject>
      <SourceXMLFile>source_xml_file.xml</SourceXMLFile>
      <SourceProjectType>source_project_type</SourceProjectType>
      <SourceProjectName>source_project_name</SourceProjectName>
   </SourceProject>
</ProjectMap>
```

Options:

- **<DestXmlFile>** defines the name and location of the DimensionsXML output file. If this file exists it is overwritten. Exactly one <DestXmlFile> must be included in a <ProjectMap> element.
- **<DestProduct>** defines the destination Dimensions product. If this is not specified, then the name is derived from the source project name. The value must be entered in upper case letters. This is allowed in the <ProjectMap> and <SourceProject> elements.
- **<DestPart>** defines the default design part for any migrated objects that do not have a a default specified via a RuleSet. The value must be entered in upper case letters. If no <DestPart> is specified, the default is to create a second level design part in the Destination product, which has a name derived from the <SourceProject> element.

  For example, at the top level you have a product name (specified by the <DestProduct> element). Any secondary project names are used to create design part names for that product. This facilitates a migration scenario where multiple projects are migrated into a single Dimensions product.

  Design Part names produced by the transformation are stripped of any invalid characters. The allowed characters are: alpha-numeric, space, and underscore.

  This element is allowed in <ProjectMap> and <SourceProject>.
- **<DestParentPart>** defines the destination parent design part. This is only used if the specified <DestPart> does not exist, in which case it is created as a child of this part. The value must be entered in upper case letters. This element is allowed in <ProjectMap> and <SourceProject>.

- **<DestWorkset>** defines the destination workset. If specified, this becomes the default destination workset for any migrated objects that do not have a more explicit default. This element is allowed in <ProjectMap> and <SourceProject>.

  Note that a destination project or stream must be defined by either a <DestWorkset> or a <DefaultContent> element within a <RuleSet>.

- **<DestVariant>** defines the product variant. The default for this optional element is AAAA. This element is allowed in <ProjectMap> and <SourceProject>.

- **<DestProductManager>** defines the destination product manager. If not specified, the default product manager is pcms. Use this if you are running the import when logged in as a user other than pcms. This element is allowed in <ProjectMap> and <SourceProject>.

- **<DestProductManagerRole>** defines the role associated with the destination product manager. By default this is TEAM LEADER. If that role is not correct for the destination Dimensions process model, specify the correct role. This element is allowed in <ProjectMap>.

- **<ItemLibrary>** defines the item library path for the destination product. This element is allowed in <ProjectMap>.

  If the <ItemLibrary> element is specified as a child of <TransformXML> then the name of the destination product is appended to the path as a sub-directory. This allows you to specify a root item library directory in the Transform.xml file, and each product defined by multiple <ProjectMap> elements will have an appropriate product specific sub-directory. If the <ItemLibrary> is specified within a <ProjectMap> it is used as is.

  If the <ItemLibrary> element is not specified in the Transform.xml file, Dimensions uses the directory specified by DM_LIBRARY_DESTINATION in the DmImport.cfg configuration file.

- **<DestLifecycleID>** defines the lifecycle ID for destination items. If not specified, Dimensions defaults to the ENTRY_LEVEL process model ITEMS_LC lifecycle. A more specific lifecycle can be specified as part of the DefaultContent in a RuleSet. This element is allowed in <ProjectMap> and <SourceProject>.

- **<DestStatus>** defines the status for the destination items. If not specified, Dimensions import will default to the ENTRY_LEVEL process model QA_ACCEPTED status. This element is allowed in <ProjectMap> and <SourceProject>.

- **<PartCategory>** defines the part category for design parts. If not specified, Transform will default to using PROJECT or MODULE. If the parent part of the design part is a product, then the default part category of that design part is PROJECT. Otherwise the default part category is MODULE. This element is allowed in <ProjectMap> and <SourceProject>.

- **<SourceProject>** defines the source XML files to use. One or more <SourceProject> elements must exist. This element is allowed in <ProjectMap>. You can copy and repeat the source project element as many times as required to define all of the source project files you want to use. The syntax is as follows:

```
<SourceProject>
   <SourceXMLFile>source_xml_file.xml</SourceXMLFile>
   <SourceProjectType>source_project_type</SourceProjectType>
   <SourceProjectName>source_project_name</SourceProjectName>
</SourceProject>
```

  Where:

- <SourceProject> references a single source ClearCase project
- <SourceXMLFile> references a ClearCase export XML file.
- <SourceProjectType> ClearCase
- <SourceProjectName> is the project name. This is used to derive the destination design part if one wasn't specified.

<ItemStrategy> Element

The **ItemStrategy** element specifies the rules to use to map ClearCase archives and revisions into Dimensions Items. The strategy includes methods to derive destination design parts, worksets, workset paths, item types, item formats, etc., based on any part of the ClearCase entity path.

By default:

- Every revision of every ClearCase file is migrated into a Dimensions item.
- The destination design part for the current project becomes the owning design part for the destination item. If no part is specified, a second level design part is created in the destination product with the same name as the source project.

You can override the default behavior with several mechanisms, which can be used in cooperation with each other. The mechanism for specifying default values is the `RuleSet` element. An `ItemStrategy` element can specify that a named `RuleSet` be applied. A `RuleSet` element can include other `RuleSet` elements.

Syntax example:

```
<ItemStrategy
    DesignPartLevel="number_of_levels_for_same_name_design_parts"
    WorksetPathLevel="number_of_levels_for_workset_path"
    ApplyRuleSet="myRules">
   <RuleSet Name="myRules"> . . . </RuleSet>
   <Exclude>.*/obsolete/.*</Exclude>
</ItemStrategy>
```

Options:

- **<DesignPartLevel>** defines the number of levels in the ClearCase entity path to use for generating same-named design parts. This allows a specified number of levels of a ClearCase project folder hierarchy to be preserved in Dimensions as a design part hierarchy. A design part is derived directly from the entity path using this mechanism only if no `RuleSet` rule has provided a design part.
- **<WorksetPathLevel>** indicates the number of levels of depth in the Version Manager entity path to strip for the workset path. The default is for zero path components to be stripped, so that the ClearCase path is preserved in the project path. This element is optional.
- **<ApplyRuleSet>** specifies the name of a RuleSet element used to resolve defaults for this item strategy. This element is optional. If none is specified, no rule set is applied. Only one `ApplyRuleSet` element can be used.
- **<Exclude>** specifies a ClearCase entity path pattern (a regular expression). If a source ClearCase entity matches the pattern, it is not migrated. Zero or more Exclude elements can exist. The exclude pattern shown below causes any entity path that contains an 'obsolete' component to be excluded from the migration.
  `<Exclude>.*/obsolete/.*</Exclude>`

Usage notes:

- The `ItemStrategy` element can be specified at the `<ProjectMap>` level.
- When processing a `ProjectMap` specification, the nearest `ItemStrategy` element is used. There is no inheritance.
- `RuleSet` elements are separate entities that can be defined outside of `ItemStrategy` elements.
- The `ItemStrategy` element is processed in the following order:
  - **a** Exclusions are processed first. If the pattern matches, the ClearCase entity is not migrated.
  - **b** If a rule set is specified by the `ApplyRuleSet` element, it is applied.
  - **c** If no `owningPart` is assigned as default content from a match in an applied RuleSet, then if the DesignPartLevel was specified, it is applied.
  - **d** If no `worksetFilename` is applied as default content from a match in an applied `RuleSet`, then `WorksetPathLevel` is applied. If it was not specified, the default is zero, so zero paths are stripped from the `entityPath`.

The following table shows how the DesignPartLevel and WorksetPathLevel attributes work together.

VM Entity Path	Resulting Parts & Items	Comments

`DesignPartLevel="2"`
`WorksetPathLevel="2"/>`
 	
`level1/level2/level3/level4/filename.cpp`
`animals/bears/brown/bear.java`
`animals/bears/brown/bear.class`
	`TopLevelDesignPart`
	`Level1`
	`Level2`
	`level3/level4/filename.cpp [item]`
	`animals`
	`bears`
	`brown/bear.java [item]`
	`brown/bear.class [item]`
		Two levels of design parts are generated.
		
		When DesignPartLevel and WorksetPathLevel are identical, the sum of the design part hierarchy and the project path is equivalent to the original entity path.

`DesignPartLevel="0"`
`WorksetPathLevel="0"/>`
 	
`level1/level2/level3/level4/filename.cpp`
`animals/bears/brown/bear.java`
`animals/bears/brown/bear.class`
	`TopLevelDesignPart`
	`level1/level2/level3/level4/filename.cpp [item]`
	`animals/bears/brown/bear.java [item]`
	`animals/bears/brown/bear.class [item]`
		When both levels are zero, no design part is derived, and no part of the original path is stripped when generating the project path.

`DesignPartLevel="0"`
`WorksetPathLevel="1"/>`
 	
`level1/level2/level3/level4/filename.cpp`
`animals/bears/brown/bear.java`
`animals/bears/brown/bear.class`
	`TopLevelDesignPart`
	`level2/level3/level4/filename.cpp [item]`
	`bears/brown/bear.java [item]`
	`bears/brown/bear.class [item]`
		No design part generation when level=0.
		
		Project paths are stripped of one level.

VM Entity Path	Resulting Parts & Items	Comments
DesignPartLevel="3" WorksetPathLevel="3"/> level1/level2/level3/level4/ filename.cpp animals/bears/brown/bear.java animals/bears/brown/ bear.class	TopLevelDesignPart Level1 Level2 Level3 level4/ filename.cpp [item] animals bears brown bear.java [item] bear.class [item]	
DesignPartLevel="2" WorksetPathLevel="2"/> animals/bears/brown/bear.java stars/bears/brown/ constellation.java	TopLevelDesignPart animals bears brown/bear.java [item] stars bears02 brown/ constellation.java [item]	Same as the previous example, but illustrates that bears02 was created as a separate design part, avoiding a naming conflict.

<RuleSet> Element

The RuleSet element allows you to assign design parts, projects, item formats, and item types to specific classes of items. The purpose is to provide design part and workset overrides for subsets of the ClearCase content. Item format and item types can be derived by RuleSet element or they can be left to the import phase where existing Dimensions upload rules will be applied.

The rules in the RuleSet element are applied in a specific order in an attempt to match the ClearCase entity path to a pattern (a regular expression). When a match occurs, any default values from that rule are applied and the search continues, so that default values accumulate from all of the matching rules.

NOTE You can specify the target Dimensions project here, under DefaultContent. Remember that WorkSet must be specified for the revision.

Syntax example:

```
<RuleSet Name="unique_name_for_rule_set">
   <Rule EntityPathExp=".*\.java">
            <DefaultContent workSetSpec=workset_name>
            <type>SRC</type>
            <format>TXT</format>
            </DefaultContent>
   </Rule>
   <Rule EntityPathExp=".*/chess/server/\.*">
            <DefaultContent>
            <owningPart>SERVER</owningPart>
            <lifecycleId>CORE_LC</lifecycleId>
            <status>NEW</status>
            </DefaultContent>
   </Rule>
</RuleSet>
```

Options:

- **<Name>** defines a unique name within the context of the transformation. This name is used to reference `RuleSet` definitions from the `ApplyRuleSet` element under `ItemStrategy`, or from the `IncludeRuleSet` element under `Rule`. If the `Name` element is omitted, the name is the empty string "".
- **<Rule>** contains the following:
  - **IncludeRuleSet** specifies the name of a `RuleSet` element to include at this point in the ordered list of rules. This allows you to re-use `RuleSet` elements that are already defined. If this attribute is specified, no other attributes or elements should be specified.
  - **EntityPathExp** is a regular expression pattern that is matched against the originating ClearCase entity path. If it matches, the `DefaultContent` element specified in this rule is used.
  - **DefaultContent** specifies an arbitrary set of XML content. This content should be the child content of a `DimensionsXML/item` element.
- **<DestWorkset>** or **<DefaultContent>** specifies a destination Dimensions project. A `DestWorkset` element can be defined within the Transform.XML file, in the `ProjectMap` or `ServiceProject` elements. A <DefaultContent> element can be defined within an applied `RuleSet` for an `ItemStrategy` element.
- **<Type>** specifies the destination item type.
- **<format>** specifies the destination item format.
- **<owningPart>** specifies the destination design part for new items which match this rule. This may be specified in a hierarchical form to indicate part parentage,for example: grandParent/parent/part. If the part names match those that have been derived from entity paths (via `DesignPartLevel`) then this can be used in conjunctcion with `DesignPartLevel`. New `designPart` elements will be produced in the DimensionsXML for any part referneced within a hierarchical specification. For example, /A/B/C/D/E m cause 5 new `designPart` definitions to be created. The root of the path is always considered to be the Project Destination Design Part (specified by the `DestPart` element in `ProjectMap`).
- **<lifecycleId>** specifies the lifecycle that items matching this rule should be created with. This overrides the `DestLifecycleId` specified in `ProjectMap` (if any).

- **<status>** specifies the lifecycle state that the item should be created in. This overrides the DestStatus specified in ProjectMap.

Usage notes

- RuleSet elements work in conjunction with the Dimensions upload rules. The RuleSet settings are applied during the transform process, while the upload rules are applied during the import process.
- Zero or more RuleSet elements can be defined anywhere in the Transform.XML file.
- The patterns used by RuleSet elements are regular expressions, not wild cards as typically used by a command line shell. Therefore, when you want to match a file pattern such as *.java, you must specify .*\.java in order to match any sequence of leading characters followed by the literal .java. Consult any regular expression documentation for further details.
- Rules are processed in the order they are defined and all rules are processed. Multiple Rules may be matched to accumulate default values from multiple rules. For example, the following XML fragment is a set of Rule elements that would appear within a RuleSet. The text form of the XML is:

```
<Rule EntityPathExp=".*/client/.*">
                  <DefaultContent worksetSpec="ClientWorkset">
                  <owningPart>CLIENT</owningPart>
                  </DefaultContent>
</Rule>
<Rule EntityPathExp=".*/server/.*">
                  <DefaultContent worksetSpec="ServerWorkset">
                  <owningPart>SERVER</owningPart>
                  </DefaultContent>
</Rule>
<Rule EntityPathExp=".*\.java">
                  <DefaultContent>
                  <type>JAVA</type>
                  <format>TXT</format>
                  </DefaultContent>
</Rule>
<Rule EntityPathExp=".*\.cpp">
                  <DefaultContent>
                  <type>C</type>
                  <format>TXT</format>
                  </DefaultContent>
</Rule>
```

  In this example, the RuleSet specifies that the destination project and owning design part are based on client / server directories in the soruce entity path, while the file type and format are based on the filename extension. This would result in the following defaults for the files listed below:

  - some/client/file.x:ClientWorkset, CLIENT, BIN, BIN
  - some/server/file.javaServerWorkset, SERVER, JAVA, TXT
  - somefileMainWorkset, MAIN, BIN, BIN

<UserAndGroupStrategy> Element

This element specifies how users and groups will be migrated to Dimensions. By default, all users and all groups are migrated.

Syntax example:

```
<UserAndGroupStrategy>
   <ProcessUsers>Yes/No</ProcessUsers>
   <ProcessGroups>Yes/No</ProcessGroups>
   <ExcludeGroups>groups_to_exclude</ExcludeGroups>
   <RoleAssignmentStrategy>
            <CreateProjectAccessAssignments
            Role="specify_Dimension_role_name_to_use">
            </CreateProjectAccessAssignments>
            <CreateGroupAssignments>Yes/No</CreateGroupAssignments>
   </RoleAssignmentStrategy>
</UserAndGroupStrategy>
```

Options:

- `<ProcessUsers>` specifies whether or not to migrate ClearCase users. Set to `Yes` or `No`.
- `<ProcessGroups>` specifies whether or not to migrate ClearCase groups. Set to `Yes` or `No`.
- `<ExcludeGroups>` specifies a group name pattern (a regular expression) for groups you do not want to migrate. If a source group matches this pattern, it is not migrated.
- <RoleAssignmentStrategy> assigns roles to the users.
- `<CreateProjectAccessAssignments>` specifies a project role to assign to users in the destination design part, using the `Role` attribute. If no name is specified, the default is PROJECT_ACCESS.
- `<CreateGroupAssignments>` creates a corresponding role assignment for each ClearCase group that is migrated. Set to `Yes` or `No`.

Usage notes:

- At most, just one `UserAndGroupStrategy` element can be specified within the Transform.XML file.
- When source user IDs are too long (more than 25 characters), they are not migrated and an error is written to the audit file.
- Passwords are not migrated.
- When there are duplicate users, the first user encountered is migrated and any later references to that user are ignored by the transformation process. A warning is written to the audit file for each duplicate user.
- Groups can be migrated even if users are not, since this is the specification of the role name and description, not the members.
- Two levels of role assignments can be generated automatically.
- Project access means that if a user existed in the source project, then that user is given a project access role assignment at the project's destination design part.
- If you do not want to use all of the role assignments produced during the transformation, you can use an XML editor to modify the resulting DimensionsXML file and remove any unnecessary role assignments before running the import to Dimensions.

<BaselineStrategy> Element

The BaselineStrategy element defines the migration of labels to baselines, projects, and attributes. The BaselineStrategy element can be specified within the ProjectMap or SourceProject elements. When it is specified within the SourceProject element, it becomes the default for that project. When it is specified within the ProjectMap element, it becomes the default for any SourceProject within that ProjectMap element that does not have its own default. When it is specified within the TransformXML element, it becomes the default for any ProjectMap element that does not have a more specific default.

Example syntax:

```
<BaselineStrategy>
   <CreateAttribute Name="attribute name">
      <VersionLabel>label</VersionLabel>
      <VersionLabels>regular expression</VersionLabels>
      <PromotionGroup>promotion group name</PromotionGroup>
      <PromotionGroups>regular expression</PromotionGroups>
   </CreateAttribute>
   <CreateWorkset Name="project name">
      <VersionLabel>label</VersionLabel>
      <VersionLabels>regular expression</VersionLabels>
      <PromotionGroup>promotion group</PromotionGroup>
      <PromotionGroups>regular expression</PromotionGroups>
   </CreateWorkset>
   <CreateBaseline Name="baseline name">
      <VersionLabel>label</VersionLabel>
      <VersionLabels>regular expression</VersionLabels>
      <PromotionGroup>promotion group</PromotionGroup>
      <PromotionGroups>regular expression</PromotionGroups>
   </CreateBaseline>
</BaselineStrategy>
```

Options:

- **<CreateAttribute Name=*name*>** creates a multi-valued attribute to which to transform labels. The required Name parameter specifies the name of the attribute.
- **<CreateWorkset Name=*name*>** specifies a project into which to transform labels. The required Name parameter specifies the name of the project.
- **<CreateBaseline Name=*name*>** specifies a baseline into which to transform labels. The required Name parameter specifies the name of the baseline.
- **<VersionLabel>** specifies the name of the label to transform. Zero or more of these can be present within a <CreateAttribute>, <CreateWorkset>, or <CreateBaseline> element.
- **<VersionLabels>** specifies a regular expression for a set of labels. Zero or more of these can be present within a <CreateAttribute>, <CreateWorkset>, or <CreateBaseline> element. The regular expression is ignored if it is invalid.

Default behavior:

In the absence of a baseline strategy, all version labels are migrated to a multi-valued attribute named *VCS_TAGS*.

Dimensions Import - DmImport

The Dimensions import process uses the XML files from the transformation process to import ClearCase data into Dimensions. You can customize the import configuration file as needed. The configuration is straightforward, requiring that you define the file names and paths for the project files to be imported, and the name(s) for the resulting files.

Syntax example

```
TEMP_DIRECTORY path_and_directory_name
USER_NAME user_name
USER_PASSWORD password
DB_NAME database_name
DB_PASSWORD database_password
DM_LIBRARY_DESTINATION path_to_Dimensions_library
REMOTE_LIBRARY_NODE machine_name
REMOTE_NODE_USER_PASSWORD password
DM_LIBRARY_IS_DELTA Y or N
FORCE_TRANSFER_ARCHIVES Y or N
ONLY_TRANSFER_ARCHIVES Y or N
```

Options

- **TEMP_DIRECTORY** specifies the temporary directory to use to store files such as the PDIFF and process files during the import process. For example:
  `TEMP_DIRECTOR C:\\temp`

  This must be specified.

- **USER_NAME** specifies the user name for the user doing the import. This must match the PRODUCT_MANAGER for the Dimensions products being created. For example:
  `USER_NAME pcms`

  This must be specified.

- **USER_PASSWORD** specifies the password for the specified user. For example:
  `PASSWORD pcms_pw`

  This must be specified.

- **DB_NAME** specifies the name of the Dimensions database to use. For example:
  `DB_NAME pcms_tool`

  This must be specified.

- **DB_PASSWORD** specifies the password for the Dimensions database. For example:
  `DB_PASSWORD pcms_tool_password`

- **DM_LIBRARY_DESTINATION** specifies the path and name of the directory on the host machine where the product item libraries will be created. The item library used during import (creation) of a given imported object is determined as follows:

  a If the item library definition already exists for a type, it is used.

  b If an `<ItemLibrary>` element was specified in the Transform.xml file, it is used.

  c If `DM_LIBRARY_DESTINATION` is specified, it is used.

  This allows a clean import into existing process models where item libraries are already defined by type.

Per-product subdirectories are automatically added by DmImport if the DM_LIBRARY_DESTINATION setting is in effect. For example, for a product called *PROD*, if DM_LIBRARY_DESTINATION=C:\libraries, DmImport will store items for this product in c:\libraries\PROD\.

- **REMOTE_LIBRARY_NODE** specifies the node name of the machine where the Dimensions item library will be stored. For example:
  REMOTE_LIBRARY_NODE node_name
- **REMOTE_NODE_USER_PASSWORD** specifies a password for the current user on the remote node machine if different from the DB_PASSWORD specified. For example:
  REMOTE_NODE_USER_PASSWORD pcms_pw2
- **DM_LIBRARY_IS_DELTA** specifies whether delta or non-delta Dimensions item libraries are used. The default is Y, delta libraries are used. For example:
  DM_LIBRARY_IS_DELTA N

  If the Dimensions item library already exists then DmImport will attempt to import these items into the current library.
- **FORCE_TRANSFER_ARCHIVES** forces all archives to be transferred whether the PDIFF completed successfully or not. The default is to not to force transfer of archives. Set to Y to force the transport. For example:
  FORCE_TRANSFER_ARCHIVES Y

  When using this option, be aware that if an item already exists within Dimensions and the archive is forced into the Dimensions item library, the library file may have a different filename than the value stored in the database.
- **ONLY_TRANSFER_ARCHIVES** limits the import to just the archive transfer. No metadata will be imported and PDIFF will not be run. Items must already exist for successful completion of the archive transfer. Default is not to only transfer archives. Set this to Y to only transfer archives. For example:
  ONLY_TRANSFER_ARCHIVES Y

Usage notes

- Any PDIFF files (pdiff.*, .pdiff*) will be removed from the temporary directory before the import.
- You must create system users for each Dimensions user on the server platform.
- The PRODUCT MANAGER for the Dimensions products must be the same as the user_name specified for the import.
- Use forward slashes to specify Windows directories: "\\". If the length of an archive description or revision change description exceeds the maximum length allowed by Dimensions, the import process will truncate the description to the maximum size allowed. The full descriptions will be created as a detailed description in a related change document.
- If the transfer of archives fails because it cannot find the archive file then an empty archive is created in the Dimensions item library.
- To use Dimensions, you must connect with the desktop Client.

DmImport Output

When you run DmImport, output files are created in the temporary directory (TEMP_DIRECTORY) specified in the DmImport.cfg file. The following is a list of files that are created:

- *DmImport.log* captures what is displayed on screen.
- *pdiff.<workset id>* is the PDIFF file that DmImport runs. This can be removed after an import but it can be used if any problems have occurred.
- *<import date> _pdiff.log* contains the full PDIFF output. When DmImport run the full PDIFF output is not display on screen (or in the DmImport.log file). If any error should occur during PDIFF then they will be captured on screen but this file may provide more information.
- *<import_date>_audit.csv* contains a mapping from the original Archive name to the Dimensions item object that it has been imported to. This is to help visibility of where each archive has been imported to.

Migrating Across Machines

The Dimensions Import utility must be run on a Windows system. However, you can specify that the target machine for the data (the destination Dimensions server) is a different machine (Windows or UNIX).

Follow these steps if the destination Dimensions server is not the system that you are using to run the migration utility:

1. The system running the import must have the Dimensions CM server installed.
2. The destination Dimensions server must be registered with the Dimensions server as a network node via the Administration Console.
3. With Oracle, define a Local Net Service Name for the destination Dimensions server.
4. Search for the dimensions.properties file under the application server installation, for example under \webapps\dimensions\Web-inf\classes\merant\adm\dimensions. Open this file in a text editor.
5. Add the following properties to the dimensions.properties file:
   - `Host = <name of the server machine migrating from>`
   - `jdbc_host = <name of the destination Dimensions server>`
   - `twotask = <Oracle database alias of the destination server>`
   - `jdbc_sid = <Oracle database SID of the destination server>`

     `jdbc_sid` must be set to the Oracle SID of the database and **not** to the Oracle alias; this is a restriction with jdbc.
6. An environment variable called LOCAL must be set to the Oracle alias of the destination database. For example:
   `set LOCAL=pcms`

   After setting the LOCAL variable, run the `pcms exit` command to test the connection to the destination Dimensions server.

Part 5

Migrating from ChangeMan DS

Chapter 16

What Dimensions CM Offers ChangeMan DS Users

Overview

This chapter discusses the new functionality that you will get when you migrate from ChangeMan DS 5.7.x to Dimensions CM Dimensions CM.

The following documentation is useful when used in conjunction with this chapter:

- Dimensions CM: *Dimensions CM Getting Started Guide*.
- ChangeMan DS: Chapter 2, "Product Overview" of *User Guide*.

Terminology and Feature Comparison

The following table lists the terminology and functionality mapping between ChangeMan DS and Dimensions CM:

Table 16-1. Terminology and Feature Comparison

ChangeMan DS	Dimensions CM
Area scripts	Templates
Audit trail	History
Development areas	Work area
File archive	Item revision
File tree	Pedigree
File	Item. Note, however, that items can also represent entities other than files, for example, hardware part numbers.
Frozen releases	Baselines and customer releases (both objects with actionable lifecycles)
Global process flow	Global Stage Lifecycle (GSL)
Production areas/"Home"	Item libraries
Production, QA, and end user areas	Deployment area
Projects	Projects (but a full Dimensions object that can have types and actionable lifecycle states). This replaces the more limited workset in earlier versions of Dimensions.

Additional Features of Dimensions CM

The following table lists the additional or enhanced features offered by Dimensions CM Dimensions CM compared with ChangeMan DS 5.7:

Table 16-2. Additional Features Offered by Dimensions CM Dimensions CM

ChangeMan DS	Dimensions CM
Web client	A fully featured Web client. Enables use of Dimensions CM with a minimum of software installation.
N/a	Administration Web client. A dedicated tool separate from the desktop client (DS has its administration facilities within the desktop client).
N/a	Integrated change request management facility.
N/a	Integrated requirements management (available through the separately licensed, but integrated, Dimensions RM).
Limited lifecycle management.	All Dimensions objects (for example, items, requests, and baselines) have fully configurable lifecycles.
Limited metadata configuration management that doesn't allow audit tracking.	Full robust configuration management data. Allows you to track relationships between different types of configuration assets (audit tacking).
Frozen releases.	Actionable baselines and customer releases with lifecycle management.
Merging the content of two file versions and placing the chosen composite information in one of the file versions. You can merge any two files that have a common ancestor, which is the file version from which the two files were derived.	Merging the content of two or more project files or files in a working location relative to a selected ancestor. You can select which changes to include and save the merged file to a target file as a new version in the database or a new file in the working location.
Integrations for Eclipse and Visual Studio .NET	Additional rich client interfaces: Windows Explorer, Eclipse, and Visual Studio .NET.

What's Different between ChangeMan DS and Dimensions CM

Comparing the Repositories

The following table lists the differences between the code repository in ChangeMan DS and the Dimensions CM repository:

Table 16-3. Repository Comparison

ChangeMan DS	Dimensions CM
Area management: ■ Physical repository of source code. ■ An area can reference any directory or directory structure as long as the Communication Agent is running on the host.	■ Item libraries: • Can have separate operating-system libraries dependent on the Dimensions CM type attribute (item type/file extensions). • ASCII or binary data format. • The "library" filename is different from the item filename, unlike in DS where they are the same. ■ Both physical and logical breakdowns supported. ■ Upload rules implement relationships between files and libraries. You can specify target location within the application's structure. ■ Libraries are not viewable unless if you have the appropriate permissions. ■ On line and offline (air gap) replication of items and associated metadata through the separately licensed but fully integrated Serena Dimensions Replicator product.

Lifecycle Options

The following table lists the differences between the process flow in ChangeMan DS and Dimensions CM:

Table 16-4. Defining Global Lifecycle

ChangeMan DS	Dimensions CM
Forced into choosing one of the four pre-defined specific lifecycles related to the four different types of area: ■ Production ■ Development ■ Quality Assurance ■ End User.	■ Can define your own global lifecycle using your own terminology. ■ Can also define "off normal" states, but these should be used sparingly and with circumspect.

Work Areas and Deployment Areas

The following table lists the differences between work and deployment areas in ChangeMan DS and Dimensions CM:

Table 16-5. Work Areas and Deployment Areas

ChangeMan DS	Dimensions CM
CMDS development area: ■ Working location (sandbox) for users to check in and check out.	Work area (associated with particular projects): ■ Working location (sandbox) for users to check in and check out.
CMDS production, QA/bug fix, and end user areas.	Deployment area (associated with particular projects): ■ Flexible names and quantities as per your global lifecycle. ■ On a project-by-project basis can have manual or automatic deployment. The latter can take advantage of a lifecycle state to automatically trigger a deployment.

Baselines and Releases

The following table lists the differences between baselines and releases in ChangeMan DS and Dimensions CM

Table 16-6. Baselines and Releases

ChangeMan DS	Dimensions CM
CMDS frozen release of all components	Baseline: ■ Snapshot at a particular time of a full configuration/inventory of components. ■ A common starting point for creating new customer releases. ■ Ability to base a new baseline on Dimensions CM change requests versus an existing baseline. ■ Ability to base a new baseline on templates to filter what is included. These templates can also be based on lifecycle states.
N/a	Release: ■ Generated from baseline. Typically a release is targeted for a particular customer. The release can be used to filter just what assets the customer receives. ■ Can also be used to track what has been released to a customer.

Audit Trails

The following table lists the differences between how ChangeMan DS and Dimensions CM record change history:

Table 16-7. Audit Trails	
ChangeMan DS	**Dimensions CM**
CMDS audit trail.	History. Available for different types of Dimensions CM object, for example, items, projects, baselines, and requests. Textual or graphical representation. The graphical representation is called pedigree. You can perform differences on the pedigree.

User, Groups, and Roles

The following table lists the differences between users, groups, and roles in ChangeMan DS and Dimensions CM:

Table 16-8. Users, Groups, and Roles	
ChangeMan DS	**Dimensions CM**
User: ■ Administrators control access by assigning a ChangeMan DS user id. ■ Authentication is internal to ChangeMan DS and requires internal password maintenance.	User: ■ Administrators control access by assigning a Dimensions CM user id. ■ User must also exist as an operating-system user. It is this that validation/authentication is performed against; also, this means there is no requirement for internal password maintenance. Lightweight Directory Access Protocol (LDAP) authentication is also supported. ■ A Dimensions CM administrator can grant or explicitly deny permissions to a user. Denial overrides any grants.
Groups: ■ Administrator grants group rights by making individuals members of a group. ■ Three principal groups: • Administrators. • Power users. • End users.	Groups: ■ Analogous to DS. ■ Admin group has all privileges. It must exist and must contain at least one Dimensions CM user. ■ A Dimensions CM administrator can grant or explicitly deny permissions to a group. Denial overrides any grants.

Table 16-8. Users, Groups, and Roles	
ChangeMan DS	**Dimensions CM**
N/a	Roles: ■ An association that determines what lifecycles you can define or actions you can perform. These are not global, so they provide great granularity. ■ Specific roles "out of the box". ■ A Dimensions CM administrator can grant or explicitly deny permissions to a role. Denial overrides any grants.

Checking Out, Checking In, and Revisioning

The following table lists the differences between checking in, checking out, and revisioning in ChangeMan DS and Dimensions CM:

Table 16-9. Checking In, Checking Out, and Revisioning	
ChangeMan DS	**Dimensions CM**
Check in and check out: ■ Performed with respect to development areas.	Check in and check out: ■ Transparently uses appropriately assigned item library. ■ Sandbox is determined before you do the operation by means of the project and its associated work area. Configuration of the work area can be done by user or admin.
N/a	Dimensions CM desktop client "edit" function for quick changes. Dimensions CM automatically checks out, associates a pre-defined editor, and on editor closure checks in the item
DS built in editor.	Dimensions doesn't have a built in editor. You automatically associate a third party editor based on, for example, the item type. This enables you to use the most appropriate editor.
Revision number: ■ Assigned at check in. ■ No flexibility.	Revision number: ■ Reserved at check out. ■ Flexibility: For example, you can specify a specific revision number. ■ Generally full revisions, rather than deltas. ■ The item library filename includes the revision number, for example, demo_app-01.c_1 for file `demo_app.c`. ■ The DS migration utility migrates DS revision history appropriately to Dimensions CM.

Deploying Assets

The following table lists the differences between deploying assets in ChangeMan DS and Dimensions CM:

Table 16-10. Deploying Assets	
ChangeMan DS	**Dimensions CM**
CMDS production, QA/bug fix, and end user areas.	Manual: ■ Deploy individual items. ■ Also have the ability deploy based on a change request.
	Automatic: ■ Ability to automatically deploy at a particular lifecycle stage. ■ Cannot currently mix manual and automatic deployment in same project. Have to elect for one or the other per project.
	General: ■ Can deploy to several areas at once. Define multiple deployment areas at a particular stage in a project. Make an association of those areas to a stage in a project. ■ The Serena Mover product allows scheduling of deployment. ■ You can limit deployment areas by means of permissions.
Deployment scripting: ■ Post deployment scripts only.	Deployment scripting: ■ Pre-deployment, post-deployment, or "on failure" scripts.

Build Management

The following table lists the differences between build management in ChangeMan DS and Dimensions CM:

Table 16-11. Build Management	
ChangeMan DS	**Dimensions CM**
Build processes.	Build processes. ■ Build processes are very similar. ■ Permissions maintained.

Licensing

The following table lists the differences between licensing in ChangeMan DS and Dimensions CM:

Table 16-12. Licensing

ChangeMan DS	Dimensions CM
■ Named licensing.	■ Named or concurrent licensing. ■ Migrating DS users will need, specifically, a Dimensions CM license. ■ To use integrated online or offline replication you will need an additional Serena Dimensions Replicator license. ■ To use integrated requirement management you will need an additional Dimensions RM license.

Chapter 17

Preparing to Migrate

Goals of Migration

The goals of migrating from ChangeMan DS to Dimensions CM are to:

- Convert ChangeMan DS assets to Dimensions CM assets with minimal disruption.
- Support incremental upgrades.
- Automatically migrate ChangeMan DS functionality to corresponding Dimensions CM where possible.

The Migration Process

The migration process involves:

- Determining the scope of the migration, see "Determining the Scope of the Migration" on page 148.
- Creating pre-configuration information, see "Pre-Migration Requirements" on page 149.
- Reviewing what gets migrated, see Chapter 18, "What Gets Migrated from ChangeMan DS Objects to Dimensions CM?".
- Performing the migration, see Chapter 19, "Using the ChangeMan DS Migration Utility" and Chapter 19, "Using the ChangeMan DS Migration Utility".

Determining the Scope of the Migration

To assess the scope of the migration, you need to determine:

- What is to be migrated: Is this a full or partial migration of assets?
- Where in Dimensions CM are the ChangeMan DS objects to be migrated? This can be:
  - A local location.
  - An online or offline replicated location.
- What is the impact of the migration on your current ChangeMan DS users.

Pre-Migration Requirements

Local Windows Migration System Requirements

Constituent	Requirement
DS server host.	DS server 5.7.2.
DS area host	■ DS Agent 5.7.2 installed. ■ Running a Dimensions CM Listener. ■ A Dimensions CM server host name registered in the operating system host table.
Migration utility host.	■ JVM 1.4.x or later installed. ■ Dimensions CM desktop client installed. ■ An existing DSN pointing to the DS Server database. ■ The correct values for the following environment variables: • DM_ROOT The path setting for the root of the Dimensions CM server installation, see the *System Administration Guide.* • DMDB The connection string for the Dimensions CM database, see the *System Administration Guide*. • DIMENSIONS_REMOTE_HOME Blank or no entry. • TRANSFER_REMOTE_PATH Blank or no entry.
Dimensions CM server host.	■ Running a Dimensions CM Dimensions CM server. ■ Running Tomcat. ■ DS Communication Agent 5.7.2 or Server running.

Pre-Configuring Dimensions CM for the Migration

The implementation of automatic pre-configuration includes automatic creation of pre-set items such as:

- DS-like roles in addition to the existing roles in Dimensions.
- Area types (Production, Development, Q&A and End User).
- Lifecycles (Process flows) for the various area types.
- Products
- Design parts

When you migrate one or more products should be pre-configured prior to the migration, or configured and created during the migration process.

Chapter 18

What Gets Migrated from ChangeMan DS Objects to Dimensions CM?

Migrating User and Group Information

The following properties can be retrieved from the DS database.

- User Name: (the OS user should be created with the same name)
- Password (if the OS user has been created and the password satisfies security requirements
- Full User Name

For all the groups, the corresponding roles should be assigned. The roles may be picked up by the administrator from the existing standard roles, such as `PROJECT-MANAGER` or `CHANGE-MANAGER`.

Standard roles may be mapped based on the existing set of permissions in DS. For example, the `PROJECT-MANAGER` role (because of the logical similarity of projects and areas in DS and Dimensions CM) would be assigned to the user or group if the following permissions are set:

- Create Project/Area
- Edit Project/Area
- Delete Project/Area

To better create an environment similar to DS, a set of "stock" roles should be created in Dimensions CM.

Adding Groups to Dimensions CM

The administrator may choose one of the following scenarios when migrating groups:

- Create the same group in Dimensions CM. Then the administrator will select a set of roles.
- Delete a group. In this case the administrator should select a set of roles for all orphan users (that don't belong to any of the groups).
- Combine groups and create a new one. All the users will be placed to the new group. Then the administrator will select a set of roles.
- Rename a group. All users will be placed in the new group. Then the administrator will select a set of roles as for option 1.

When all the groups have been created in Dimensions CM, all the roles previously selected by the administrator will be applied to the groups.

The next step is to put all the users previously created and registered with Dimensions CM into corresponding groups.

Creation of the Dimensions CM groups does not affect OS groups.

Applying Roles

The following table contains a mapping of DS permissions to Dimensions CM privileges. The description column contains DS constants to be used to access the corresponding permission setting. There are two subsets of permissions—user and administrative.

Table 18-1. Mapping of DS Permissions to Dimensions CM Privileges		
Name	**Description**	**Dimensions CM privilege**
General permission		
Validate user against OS.	The user login is being validated against the OS on the specified host.	N/A However, if the host is the same as the Dimensions CM host, this information may be used to map the corresponding user in DS to the Dimensions CM user.
File permissions		
Check out.	The user can check out and retrieve the file.	UPDATE_ITEM_PRIV
Check in.	The user can check in a file that has been previously checked out.	UPDATE_ITEM_PRIV
Second check out.	Allows the user to check out a previously checked out file.	UPDATE_ITEM_PRIV
Check in overwrite.	Allows the user to check in a file that hasn't been check out.	UPDATE_ITEM_PRIV
Check in merge.	Allows a merge operation on check in. This is useful to avoid problems when inexperienced developers might break the code during a merge operation.	UPDATE_ITEM_PRIV
Development merge. Production merge.	Allows the user to execute a merge.	UPDATE_ITEM_PRIV
Project permissions Similar to file permissions, but the operation is executed when the project name is specified.		
Check out.	The user can check out a file.	UPDATE_ITEM_PRIV
Check in.	The user can check in a file that has been previously checked out.	UPDATE_ITEM_PRIV
Second check out.	Allows the user to check out a previously checked out file.	UPDATE_ITEM_PRIV
Check in overwrite.	Allows the user to check in a file that hasn't been checked out.	UPDATE_ITEM_PRIV

Table 18-1. Mapping of DS Permissions to Dimensions CM Privileges		
Name	**Description**	**Dimensions CM privilege**
Check in merge.	Allows a merge operation on check in. This is useful to avoid problems when inexperienced developers might break the code during a merge operation.	UPDATE_ITEM_PRIV
Create release.	Allows the user to create a frozen release.	CREATE_BASELINE_PRIV
Delete release.	Allows the user to delete a frozen release.	DELETE_BASELINE_PRIV
Attach files.	Files can be attached to the project.	ADDITEM_WORKSET_PRIV
Detach files.	Files can be removed from the project (not from the source control).	ADDITEM_WORKSET_PRIV
Audit permissions		
Approve transactions.	Allows the user to approve pending check in operations.	ACTION_ITEM_PRIV
Delete transactions: check in.	Treated as a rollback to the version that is previous to that created by the check in operation.	DELETE_ITEM_PRIV EDITATTRIB_ITEM_PRIV
Delete transactions: branch.	Treated as a rollback to the branch that is previous to that created by the check in operation.	EDITATTRIB_ITEM_PRIV
Delete transactions: loading.	A file can be removed from the source control.	DELETE_ITEM_PRIV
Administrative permissions		
Areas Create: ■ Production ■ Development ■ Q/A ■ End User	Allows the user to create the corresponding type of areas:	CREATE_AREA_PRIV
Areas Edit: ■ Production ■ Development ■ Q/A ■ End User	Allows the user to edit the corresponding type of areas	UPDATE_AREA_PRIV

Table 18-1. Mapping of DS Permissions to Dimensions CM Privileges		
Name	**Description**	**Dimensions CM privilege**
Areas Delete: ■ Production ■ Development ■ Q/A ■ End User	Allows the user to delete the corresponding type of areas.	DELETE_AREA_PRIV
Projects: ■ Create ■ Edit ■ Delete	Allows the corresponding operations for projects.	The corresponding privileges: CREATE_WORKSET_PRIV N/A DELETE_WORKSET_PRIV
Users: ■ Create ■ Edit ■ Delete	Allows the corresponding operations for users.	USERMAN_PRIV
Groups: ■ Create ■ Edit ■ Delete	Allows the corresponding operations for groups.	USERMAN_PRIV
Process: ■ Global edit ■ Application create/edit/del ■ Project create/ edit/del	Allows the corresponding operation with the process flow.	LIFECYCLEMAN_PRIV

Migrating Files

Overview

The following table describes what properties of files are migrated.

Table 18-2. File Registry	
Description	**Is it transferable to Dimensions CM?**
File name	yes
Current file version. The last version in the repository.	This version number will be used to recreate all file versions.
The production path for the file.	This information will be used to transfer a file from DS repository.
The production file CRC32 checksum value.	This information may be used to validate the file contents.

Table 18-3. Transaction Table and File Archive	
Description	**Is it transferable to Dimensions CM?**
The file name without the path.	Will be used to locate a file in DS repository.
The transaction type: o, i, L, etc. See the detailed description in the text.	This will be used to pick transferable type of transactions to re-create a file history information.
The completion status: C, A, P, etc. See detailed description in the text.	This will be used to pick transferable types of transactions to re-create file history information.
A user-provided or automatic description of the transaction.	yes
The user who performed the operation.	yes
The file version used in the operation.	This will be used to pick a transferable type of transactions to re-create a file history information.
The full qualified path in the production archive. Maybe one of the following: ■ "delta" file ■ full text file ■ binary file ■ zipped file image	This will be used to re-create previous version of the file.

Branch information will be used to recreate the file hierarchy in Dimensions CM from the following properties.

Table 18-4. Branch and Merge Information

Property	Description
Transaction number	This contains a transaction number.
New file	The new file name after a branch. This may be different from the original file.
New area	The subarea name for a new file.
New path	The production file for a new file. This is identical to the vcs03 field home_path.
New Host	The host name.
New Version	For a branched file this version is always 1. For a merge, this is the new version as a result of the merge
Entry type	'b' for branch 'm'. for merge
Original file 1	The first file for a merge or the original file for a branch.
Original file 2	The second file for a merge. N/A for a branch.
Original Area 1	The subarea for the first file for a merge or a branch.
Original Area 2	The subarea for the second file for a merge. N/A for a branch.
Original path 1	The production path for the first file.
Original path 2	The production path for the second file. N/A for a branch.
Original host 1	The host for the first file.
Original host 2	The host for the second file. N/A for a branch.
Original version 1	The production version for the first file.
Original version 2	The production version for the second file. N/A for a branch.

Migration Considerations

Recovery

The migration process generates a `pdiff` file for all the file versions in the repository. Exporting all the file versions to a `pdiff` interchange format could require a lot of free space. It should be possible for the administrator to stop the process at any time. If the migration process is executed for a production area (all files from a production area are included) it is advisable to execute the process in small chunks, to monitor the migration process, and have the ability to cancel it.

Item Names

In the Dimensions CM repository, each item has a specification:

```
<product-id>:<item-id>.<variant>-<item-type>;<revision>
```

and this is unique per product.

A file migration takes place within a specified product, therefore, all files with identical file names are renamed in order to be able to add them to the Dimensions CM repository. All the mapping for identical files will be stored in the local state repository. This approach will allow the identification of an exact item identifier in Dimensions CM for each DS production file. For partial migration (by project or area), the local state repository will be used to avoid migrating files twice, and to obtain a corresponding item identifier.

IMPORTANT! Rename or move any files in DS with the same name and path. Filename conflicts will result in an error during migration.

Item Types

The other part of the file identifier is an item type, which specifies the type of contents. The item type is recognized using the following two techniques:

- File extension (the file extension mapping should be created).
- File contents (as being used in DS to recognize binary and text files). However the text recognition routine should be adjusted to specify which types of files should be treated as text. This will ensure that files are allocated an item type that correctly distinguishes between files of text and of binary content.

Based on the resulting item type, the corresponding lifecycle is assigned, and the item created at the initial state of that lifecycle.

If the archive cleanup feature is enabled, some versions of the file may be removed (starting from the oldest). In this case the first available revision number will not be equal to 1.

If the archive directory is not specified for an area, all the file versions will be recognized as "latest" and only one revision will be created in the repository.

The fact that the area has been migrated is recorded in the local state repository.

Migrating Areas

The properties of areas that are migrated are:

- Area Path—The full qualified path to the area location.
- Area Host Name—the host name and port to access the files
- Area Type—Production, Development, QA, or End User
- Archive Path—The full qualified path to the location of all the archives, used to migrate the files
- Area description
- Area ID

The following will be used for Notifications:

- User Area—area name used to subscribe/unsubscribe users for notifications.

- User Node—check in, check out, or approval

NOTE Notification Rules need to be set up in Dimensions CM for users to receive the required notifications.

Mapping between DS Area Types and Dimensions CM Areas

Dimensions CM has:

- Work Areas—used by developers to work on the files from a particular project
- Deployment Areas—used to contain item files for items that have been deployed to a particular stage in the Global Stage Lifecycle. A deployment area is defined in the database and associated with a specific stage.

The types of DS Areas are mapped as follows:

DS area type	Dimensions CM 10 area type
Development	Work Area—An area that defines a physical working location for a project.
QA Production End User	Deployment Area associated with a particular deployment stage in the Global Stage Lifecycle. You select the corresponding stage to associate with each of these DS areas when you run the migration utility.

Migrating Projects

The following properties of projects are transferred to Dimensions CM:

- Project description. Note that this will be combined with proj_notes1 and proj_notes1 fields if those are not empty.
- Project identifier or project name. A prefix may be appended to this name.
- Project Owner—User or group name that created a project. (Used to provide history of when the project was created and by whom.)
- Project Parent—If not empty, the project of which this is a child. The project hierarchy will be recreated using the RWWS command (relate workset to workset) providing a relative path as ".\"

The following properties of project attachments are used to generate a list of all the production files (All project attachments are considered to be for "latest" production files):

- Project name
- File name
- Subarea name
- Relative path This will be used to add the file to a corresponding location using the AIWS (Add Item to Project) command.

Migrating Frozen Releases

You can migrate frozen releases to Dimensions CM as baselines. The following are descriptions of the database fields and what information can be transferred to Dimensions CM.

- Project name—used to create a hierarchy
- Frozen release name—used to create a baseline with a similar name
- Frozen release description
- Date
- User—name of the user who created the frozen release.

The following properties are migrated for frozen release attachments to determine what file components need to be transferred:

- File name
- Subarea name
- File version. This may be zero to specify the "latest" repository version.
- Relative path to frozen release root. This will be used to determine a target location in the temporary project.

Migrating Build Information

ChangeMan DS Build Component Objects

The following sections detail the DS build component objects, their attributes, and corresponding Dimensions CM objects, and also the DS build components relationships between build entities used to import them to Dimensions CM

Build

Bellow are field descriptions with corresponding Dimensions CM object and corresponding Dimensions CM object attribute.

DS attribute Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Build’s parent project.	BuildProject	Name
Build’s parent frozen release.	BuildProject	Name
Build’s name, unique in project and release name space.	BuildConfiguration	Name

DS attribute Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Target OS user ID.	BuildArea	User
Path, used by build to execute build commands.	BuildArea	Path
Target host name.	BuildArea	Host
Build date	BuildConfigVersion	Time
Target OS user password	BuildArea	Password

Build Compiler

Below is storage for build file content description with corresponding Dimensions CM object and Dimensions CM API to import object to Dimensions CM.

DS Build attribute/ Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Build_compiler_content DS build Pre-, Global-, Post- compiler	BuildConfiguration	Script with property: Script.PRE_SCRIPT Script.MAIN_SCRIPT Script.POST_SCRIPT

Target

Bellow are field descriptions with corresponding Dimensions CM object and Dimensions CM API to import object to Dimensions CM.

DS attribute Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Build’s parent project.	BuildProject	Name.
Build’s parent frozen release.	BuildProject	Name.
Build’s name, unique in project and release name space.	BuildConfiguration	Name.

DS attribute Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Dependency's file name.	BuildConfiguration	Target Name (if depend_file was empty). Otherwise. Source Name.
Dependency's area name.	BuildConfiguration	Target Location (if depend_file was empty). Otherwise. Source Controlled Assert.
Unique in Build's name space dependency ID.		Target ID (if depend_file was empty). Otherwise. Source ID.

Target Compiler

Below is description of storage-file content for target Compiler values with corresponding Dimensions CM object and Dimensions CM API to import object to Dimensions CM.

DS attribute Description	Corresponding Dimensions CM object	Dimensions CM object attribute
Target Compiler—content of DS build script for target.	BuildConfiguration	Script with property Script.MAIN_SCRIPT

Target RelPath

The target is migrated with its relative path.

What's Not Migrated

The following major features are not migrated:

- (POA) Package Oriented Approach
- Process Flows
- Build History

Chapter 19

Using the ChangeMan DS Migration Utility

Overview

The Serena DS to Dimensions CM Conversion Utility allows you to migrate objects from ChangeMan DS to the equivalent objects in Dimensions CM using a UI Wizard that steps you through the process. You can:

- Migrate users and groups:
  - All users and groups.
  - Those users and groups that own the objects you are migrating.
  - Specified groups together with all their users.
  - Specified groups and specified users.
- Migrate areas:
  - All areas.
  - Specified areas.
  - Choose to migrate only the latest versions of the files, or a specified number of previous versions of the files.
  - Choose which Dimensions CM deployment stages to associate with each type of ChangeMan DS area.
- Migrate projects and releases
  - All projects.
  - Specified projects.
  - Migrate all frozen releases for a project or choose specific ones.
- Migrate Files
  - The attached files for a migrated project are migrated as Dimensions CM items belonging to that project.
  - The files in a migrated area are migrated as Dimensions CM items.

You can perform the migration all at once, or in a number of stages. You can also perform the migration in separate stages for users, areas, or projects.

IMPORTANT! Each object is only migrated once. It is not migrated again in subsequent runs of the utility. A file that has been migrated with an area is not migrated again with its project, it is only attached to the project.

You can save the configuration settings that you have specified as a configuration file in XML format and reload them into the utility at a later stage.

Before You Start

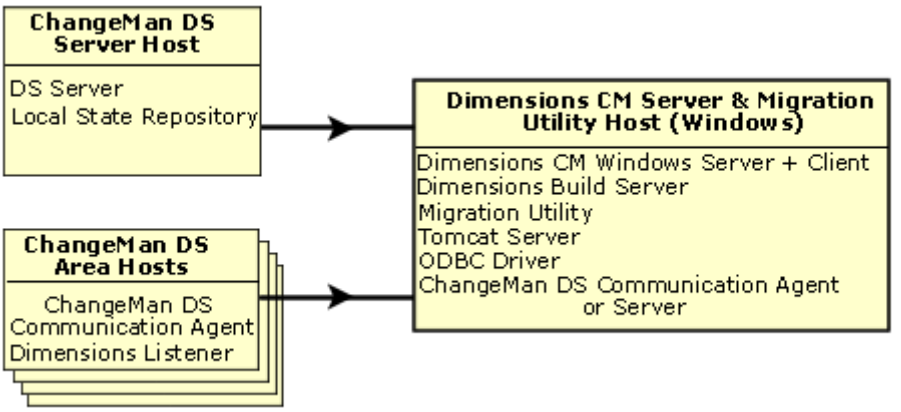
You will need to have the DS Communication Agent or Server installed on the machine from which you are running the utility. For details of this, see the *Serena ChangeMan DS Installation Guide*.

You will need to install the Dimensions CM Agent on the ChangeMan DS Area host machine and create a user account for the user under which you will be running the migration utility. This will normally be the Dimensions CM System Administrator, the default user is dmsys. For details see the *Dimensions CM Installation Guide*.

You need to run the conversion utility on a machine that has a Dimensions CM server installation, since it needs to access some of the installed `.dll` files. This is the case whether you are migrating to a Windows or a UNIX Dimensions CM server. The two scenarios are described below.

Local Migration on a Windows System

In this scenario the migration utility is run on the machine that hosts the Dimensions CM server to which you are migrating the ChangeMan DS objects.



What you need to have set up is summarized below.

ChangeMan DS Server Host Requirements:

- Running ChangeMan DS Server

ChangeMan DS Area Host Requirements:

- Running ChangeMan DS Agent
- Running a Dimensions CM listener
- The Dimensions CM Server host name is registered in the OS host table

Dimensions CM Server & Migration Utility Host (Windows):

- JVM 1.4x or later is installed
- Dimensions CM server plus client installed
- An existing DSN pointing to the ChangeMan DS Server database
- Running Tomcat
- ChangeMan DS Communication Agent or Server

- Set the correct values of environment variables:

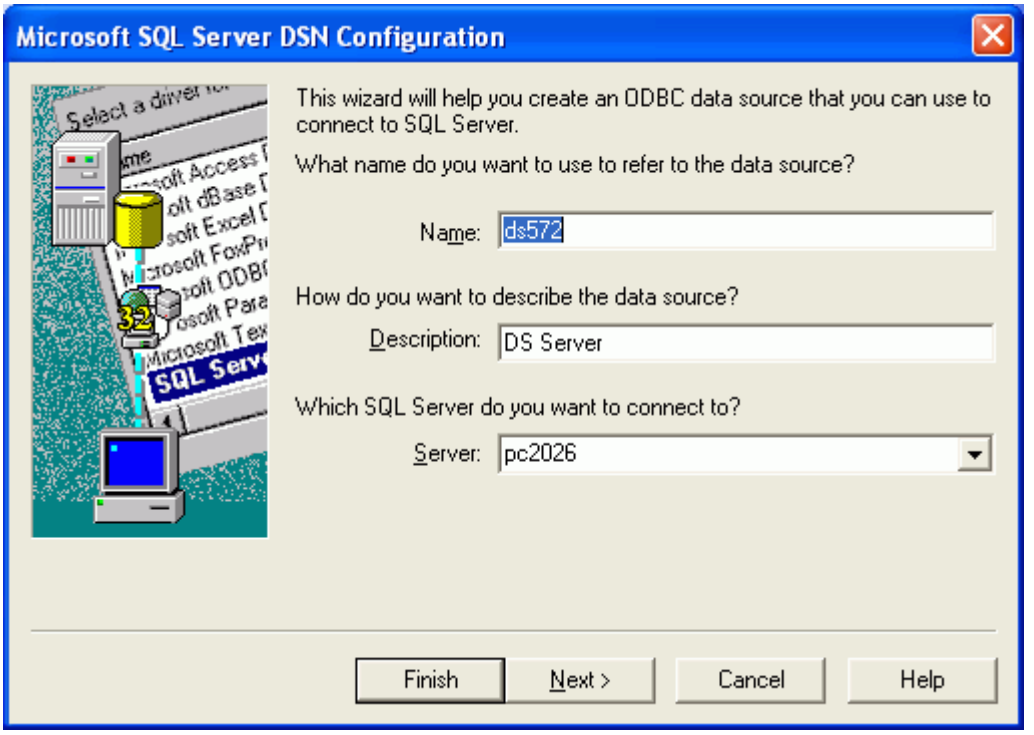
Environment variable name	Value/Description
DM_ROOT	Path to the root of the local Dimensions CM Server installation (For more details, see the *Dimensions CM System Administration Guide*.)
DMDB	Dimensions CM database connection string (For more details, see the Dimensions CM System Administration Guide.)
DIMENSIONS_REMOTE_HOME	Null value, or not set at all.
TRANSFER_REMOTE_PATH	Null value, or not set at all.

Set the DSN to Access the ChangeMan DS Database

You need to set the DSN for the ODBC driver on the Windows machine hosting the migration utility to reference the ChangeMan DS server database. The following example shows this for Windows XP and SQL Server.

To configure DSN for the ChangeMan DS database you are migrating from:

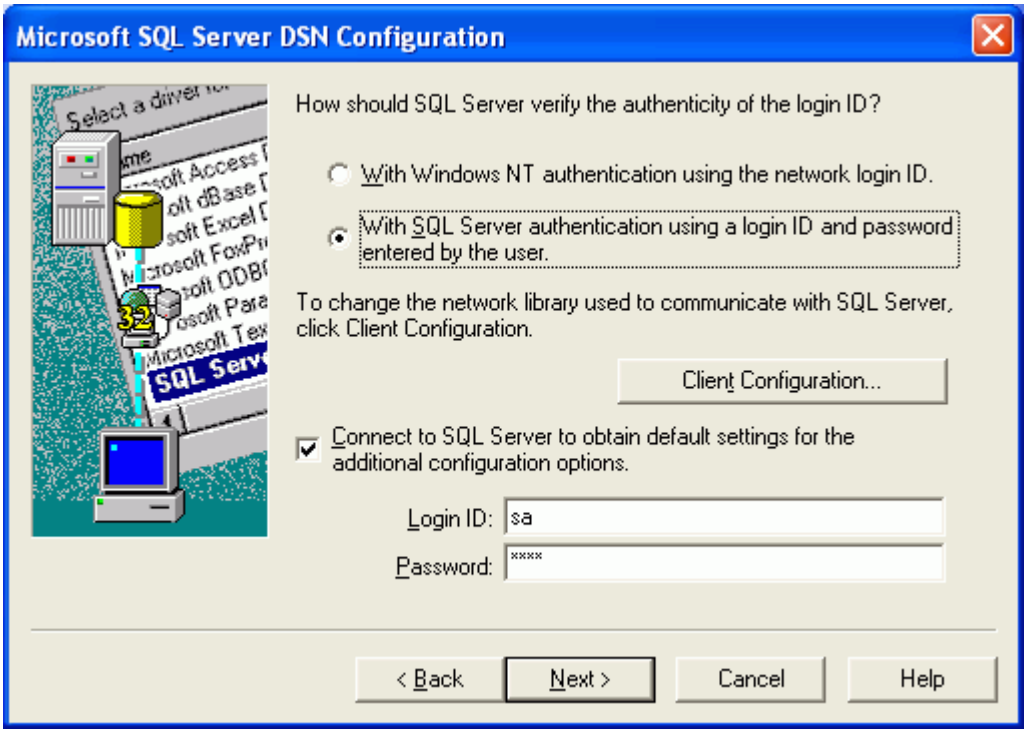
1. Select Control Panel | Administrative Tools | Data Sources (ODBC).
2. Select the System DSN tab.
3. Click **Add**.
4. In the **Create New Data Source** dialog box, select the type of database to which you will be connecting, for example SQL Server.
5. Click **Finish.**



6 For the **Name** field, type the name of the data set, and enter a **Description**.

7 For **Server**, enter the name of the ChangeMan DS server.

8 Click **Next**.



9 Select **With SQL server authentication**...

10 Select **Connect to SQL server to obtain default settings for the additional configuration options**, and enter a **Login ID** and **Password** to be use for the connection.

11 Complete the rest of the wizard.

Configure Your Dimensions CM Process Model

You may want to set up a Dimensions CM product and configure the various object types to suit your ChangeMan DS functionality instead of using the default Dimensions CM product. For details see Chapter 17, "Preparing to Migrate" on page 147.

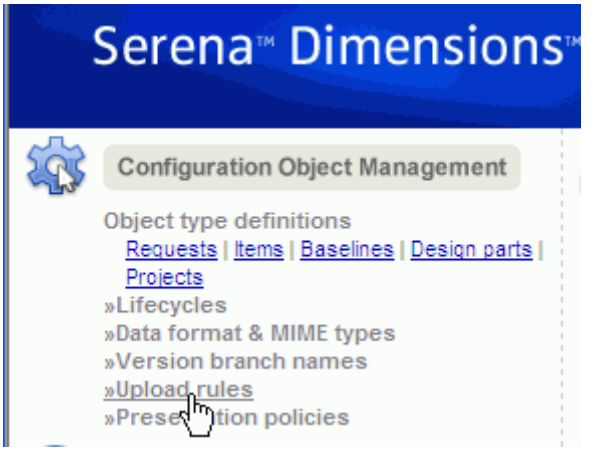
Configuring the Item Types

For any files to be migrated, you need to make sure that the option **Auto-generate identifier** is unset for the item types corresponding to those files. The item type for a migrated file is determined from the Upload rules for the product to which it is being migrated.

To configure the item types:

1 Open the Dimensions CM Administration Console for the server to which you are migrating. For details of how to do this, see the Process Configuration Guide.

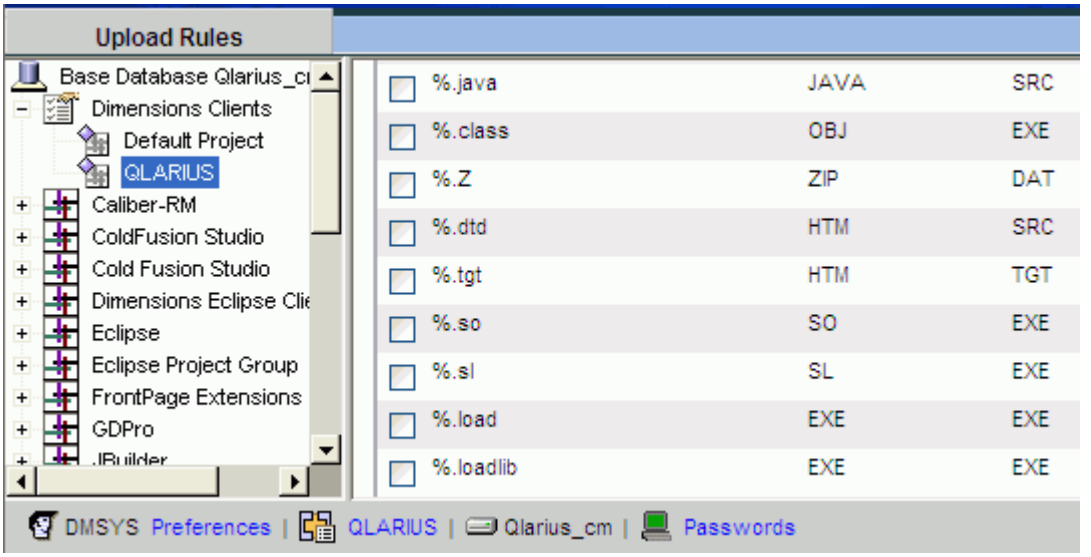
2 Select Configuration Object Management | Upload rules



3 Expand the **Dimensions Clients** node and select the product to which you are migrating the files, for example QLARIUS.

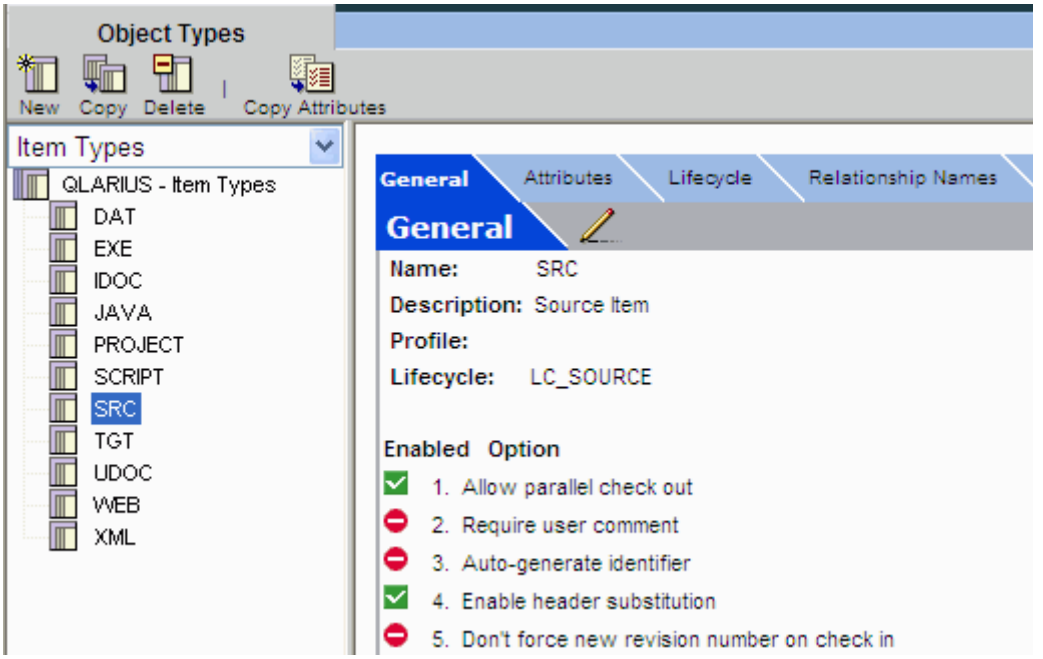
4 Locate the file matching patterns for the types of file you are matching and determine the item types for each one.

For example, if one of the file types you are migrating has an extension of `.java`, the file matching pattern `%.java` will result in items of type SRC being created for those files.



5 Select Configuration Object Management | Object Type Definitions | Items.

6 For each item type required for the migration, select the item type on the left-hand side.



If **Auto-generate identifier** is enabled:

a Click the edit button: 

b Select the Options tab.

c Uncheck **Auto-generate identifier.**

d Click **OK**.

Running the Conversion Utility

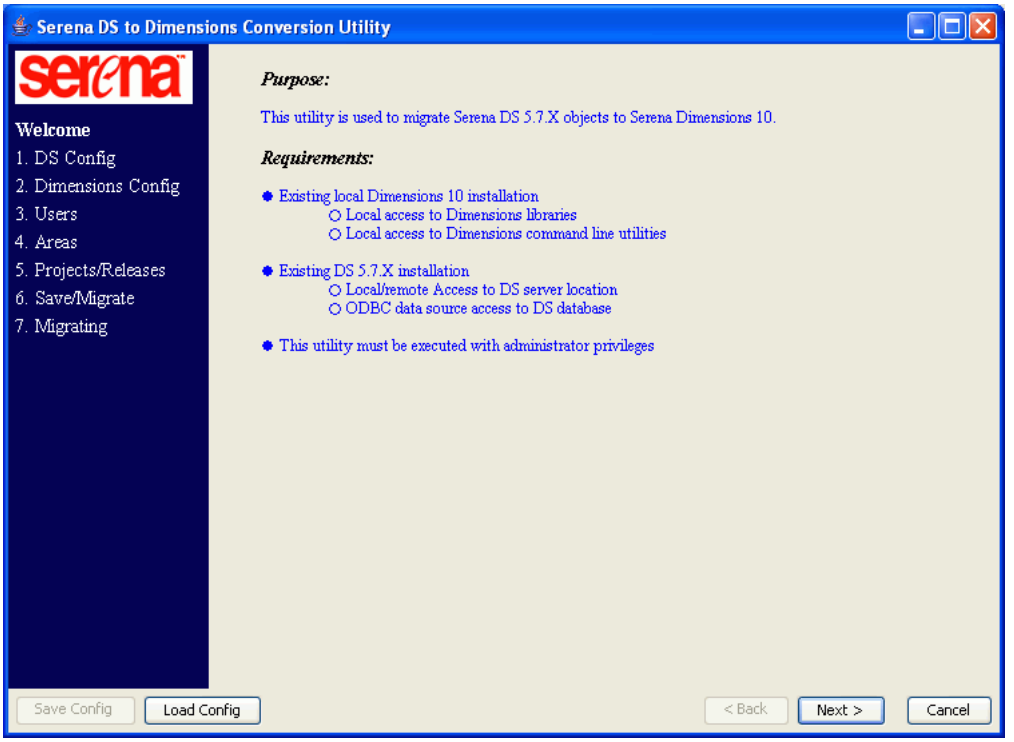
Purpose Run the Conversion utility when you want to migrate some or all of the users, areas, or projects from Serena ChangeMan DS to Dimensions CM.

To run the conversion utility

1 Do one of the following:

- Log in as the Dimensions System administrator. For a default installation, this user is dmsys. In a command window, run the supplied .bat file for the type of migration you want to run:
  - `run.bat` for a migration to a local Dimensions CM Windows server
- Logged in as your user, in a command window, use the runas command to run the required .bat file, described above, as Dimensions System administrator.

The Welcome page appears:



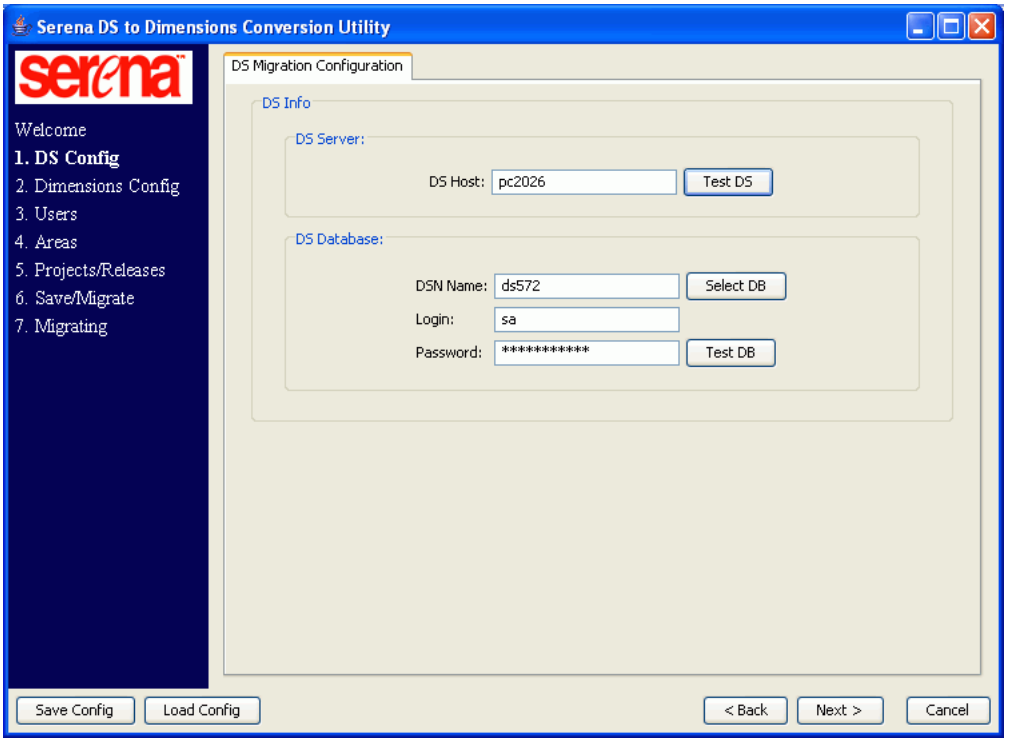
2 Do one of the following:

- If you want to use a configuration you have previously saved, click the **Load Config** button and browse for the configuration file you previously saved.

  A dialog appears saying "Wait Please, loading saved configuration..." and then the Save/Migrate page is displayed. Continue as described in Step 23 on page 179.

- If you are configuring a new migration, click **Next**.

The DS Config page appears.



3 Enter the details for the ChangeMan DS Server. For details of the fields on this page, see "DS Config Page" on page 186.

- For **DS Host**, enter the name of the machine on the network that hosts the ChangeMan DS installation from which you want to migrate.
- If you want to check that the machine exists on the network, click the **Test DS** button. After a pause, a message informs you whether the test is successful.
- Enter the **DSN name** for the ChangeMan DS installation that you are migrating from, or click the **Select DB** button and select it from the list.

IMPORTANT! To display DSN names on a Windows 2008 64 bit server, run the following application: `<System Root>\syswow64\odbcad32.exe`

Or, run ODBC from the Control Panel and display the **User DSN** tab.

- Enter the **Login** and **Password** for the database.

  To verify the login details, click **Test DB**.

4 Click **Next**.

The utility will check the ChangeMan DS Server details, and if there is a failure to access the server you will be informed with a message, and you will not be able to continue to the next page.

If the details are correct, the Dimensions Config page appears:

5 Enter the details for the Dimensions CM server to which you want to migrate. For details of the fields on this page, see "Dimensions Config Page" on page 186.

- Enter the login information:

- **User ID**: The Dimensions System administrator. For a default installation, this user is dmsys.
- **Password**: The password.
- **Server**: The name of the system to which the Dimensions CM server is installed.
- **DB Name**: The name of the database to which you want to migrate the DS information. For example, if this is the Qlarius sample database, enter `qlarius_cm`. If the server has the Payroll sample database installed, enter `intermediate`.
- **DB Connection**: The connection string for the Dimensions CM database. This is the connection string that was defined during installation of the database. By default, this is `DIM10`.

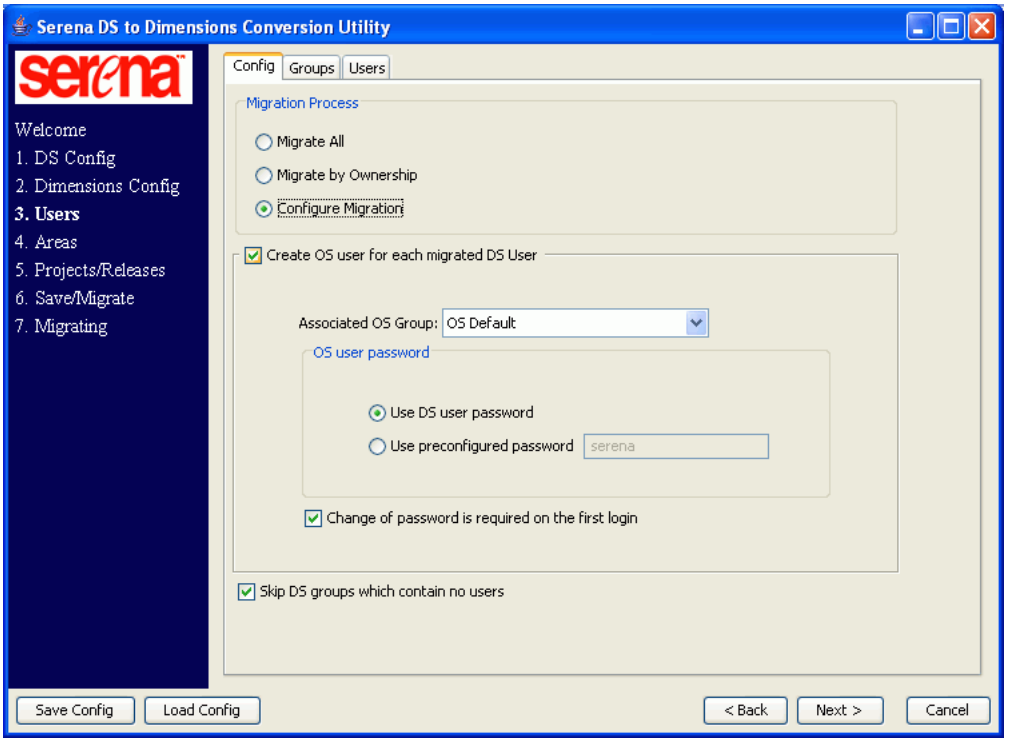
- To populate the drop-down list for **Product Name**, click **Load**. Select the name of the Dimensions CM product to which the migrated ChangeMan DS projects will belong. Note that if you want to use a new product you will first need to create it using the Administration Console.
- For **Design Part**, select the owning design part to which you want any newly-created items from the migration to belong.
- If required, enter the prefixes to be used for any new areas or projects that will be created in the Dimensions CM database as a result of the migration. (This will help you to distinguish them from any existing ones.)

  Use Area Prefix, select the checkbox and enter the required prefix for new areas.

  Use Project Prefix, select the checkbox and enter the required prefix for new projects.

6 Click **Next**.

The Users page is displayed.

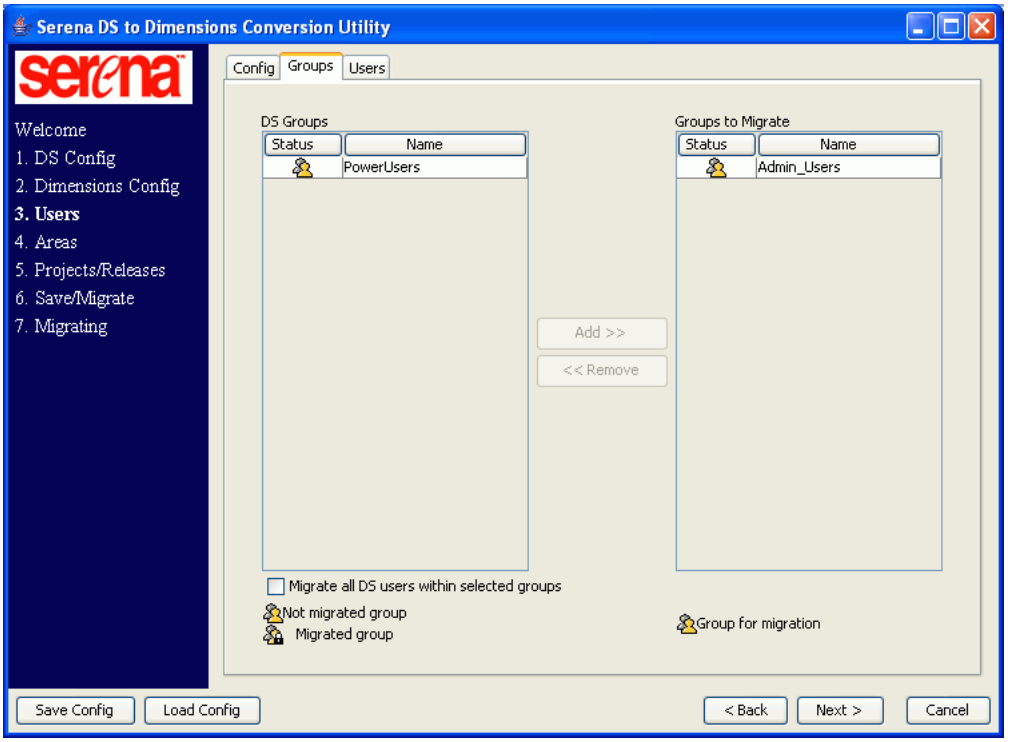


7 Enter the details in the Config tab to determine how you want users and groups to be migrated. For details of the fields on this tab, see "Users Page Config Tab" on page 187.

- Select the option for the migration process:
  - **Migrate All**. Select this option to migrate all the users and groups from the ChangeMan DS server to the Dimensions CM server.
  - **Migrate by Ownership**. Select this option to migrate users and groups from the ChangeMan DS server to the Dimensions CM server based on the ownership of migrated areas and projects. For any area or project that is migrated, the owner and the users from its authorization list will also be migrated.
  - **Configure Migration**. Specify the migration of users according to specific options.
- If you do not want empty ChangeMan DS groups to be migrated, select **Skip DS groups which contain no users**.

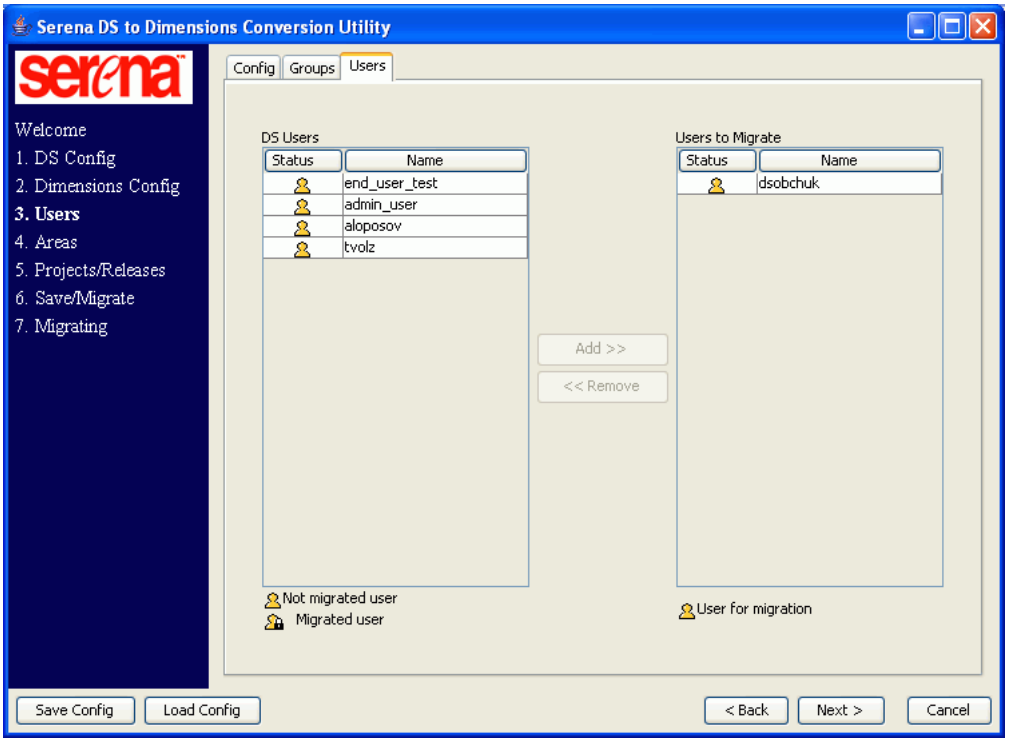
8 If you have not selected **Configure Migration**, the **Groups** and **Users** tabs will be disabled; therefore continue at Step 11 on page 175.

9 Click the Groups tab.



- If you want any ChangeMan DS groups to be migrated to Dimensions CM, select the group(s) in the **DS Groups** list and click the **Add >>** button to move them to the **Groups to Migrate** list. If you want to remove any group(s) from the **Groups to Migrate** list, select them and click **Remove**.
- If you want to migrate all the users that belong to these groups, select **Migrate all DS users within selected groups**.

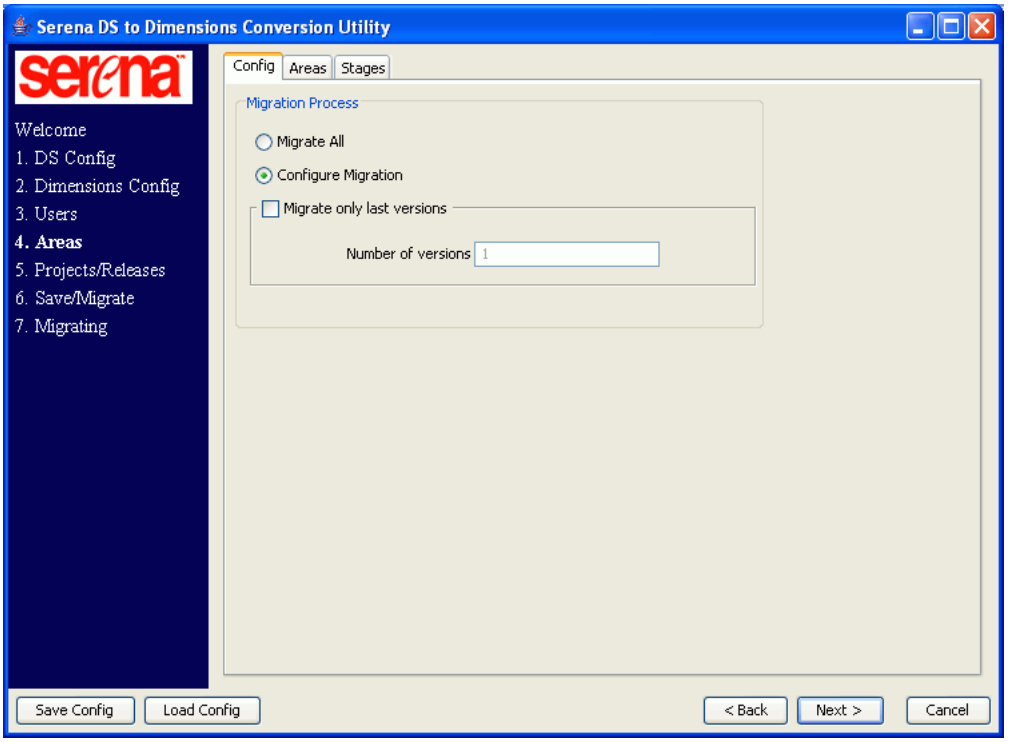
10 If you want to migrate specific users, click the Users tab.



- If you want any ChangeMan DS users to be migrated to Dimensions CM, select the user(s) in the **DS Users** list and click the **Add >>** button to move them to the **Users to Migrate** list. If you want to remove any user(s) from the **Users to Migrate** list, select them and click **Remove**.

11 Click **Next**.

The Areas page is displayed.

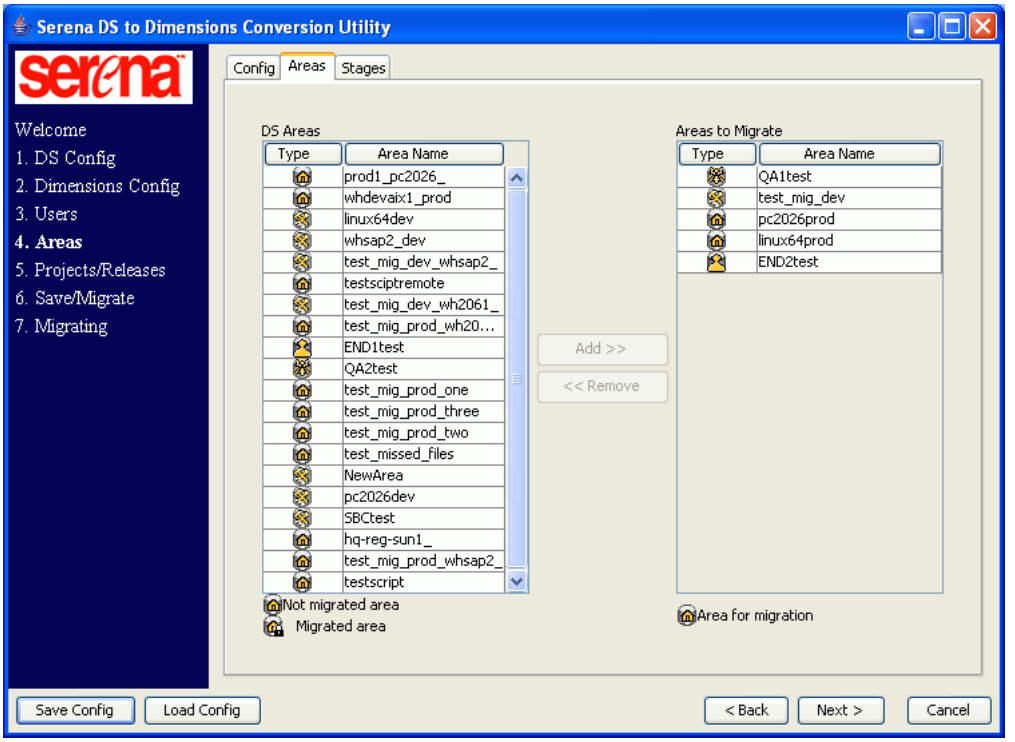


Enter the required information for the Config tab. For details of this tab, see "Areas Page Config Tab" on page 189.

NOTE The user under which you are running the utility (for example, DMSYS) will need to have write permissions to the areas on the ChangeMan DS host where the areas are located.

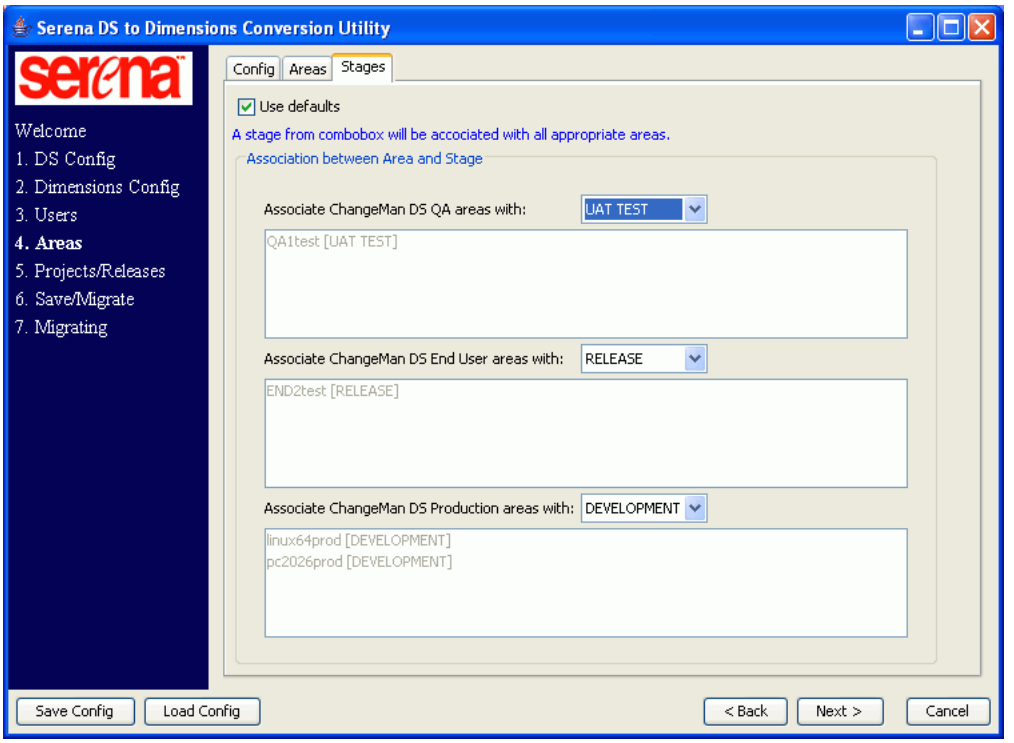
- If you want to migrate all areas from the ChangeMan DS server, select **Migrate all**.
- If you want to migrate specific areas or versions of areas, select **Configure migration**:
  - If you only want to migrate the latest versions, select **Migrate only last versions**.
  - If you only want to migrate a number of previous versions, enter the number in **Number of versions**.

12 Click the Areas tab. For details of this tab, see "Areas Page Areas Tab" on page 189



13 If you want any specific areas to be migrated to Dimensions CM, select them in the **DS Areas** list and click the **Add >>** button to move them to the **Areas to Migrate** list. If you want to remove any user(s) from the **Areas to Migrate** list, select them and click **Remove**.

14 Click the Stages tab. For details of this tab see "Areas Page Stages Tab" on page 190.



15 Select the deployment stage in Dimensions CM that you want to be associated with each type of area migrated from ChangeMan DS. Select a stage from the list for:

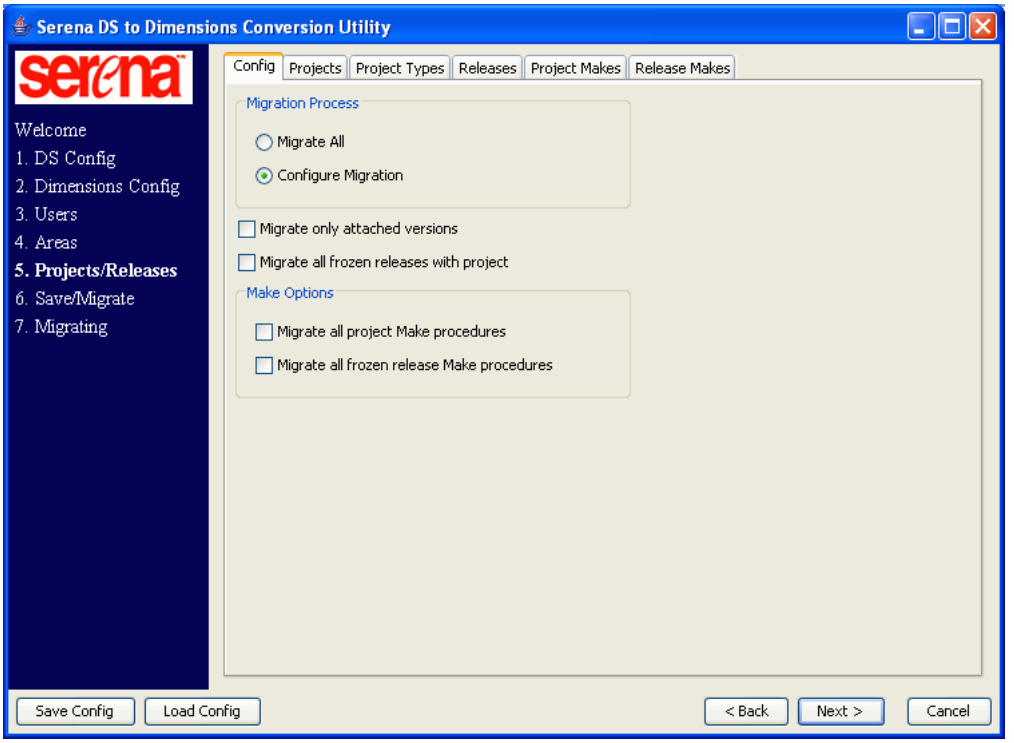
- **Associate ChangeMan DS QA areas with**
- **Associate ChangeMan DS End User areas with**
- **Associate ChangeMan DS Production areas with**

16 Display the Area password tab in order to specify the users that will be used to create areas in Dimensions. This tab lists the areas that you have chosen to migrate, and gives you the ability to enter a user ID and password for each area.

17 Select the Area location tab to specify the path for the new areas. This tab lists the areas that you have chosen to migrate and gives you the ability to enter a new path for each. By default, this is set to the same path as the originating DS area. Any new path will be an absolute path to an area on the same host where the DS area is located.

18 Click **Next**.

The Projects/Releases page appears.



Enter the required information for the Config tab. For details of this tab, see "Products/Releases Page Config Tab" on page 190.

- If you want to migrate all projects from the ChangeMan DS server, select **Migrate all**.
- If you want to choose how projects are migrated, select **Configure Migration**:
- If you only want to migrate versions of files that are attached to a migrated project, select **Migrate only attached versions**.
- If you want to migrate all the frozen releases of a migrated project as Dimensions CM baselines, select **Migrate all frozen releases with project**.

19 Click the Projects tab.

If you want any specific projects to be migrated to Dimensions CM, select them in the **DS Projects** list and click the **Add >>** button to move them to the **Projects to Migrate** list. If you want to remove any project(s) from the **Projects to Migrate** list, select them and click **Remove**. For further details of the Projects tab, see "Projects/Releases Page Projects Tab" on page 191.

20 Click the Project Types tab.

For further details of this tab, see "Projects/Releases Page Project Types Tab" on page 191.

Note that if you want to create project types with specific properties for the migrated ChangeMan DS projects, you can create them using the Administration Console.

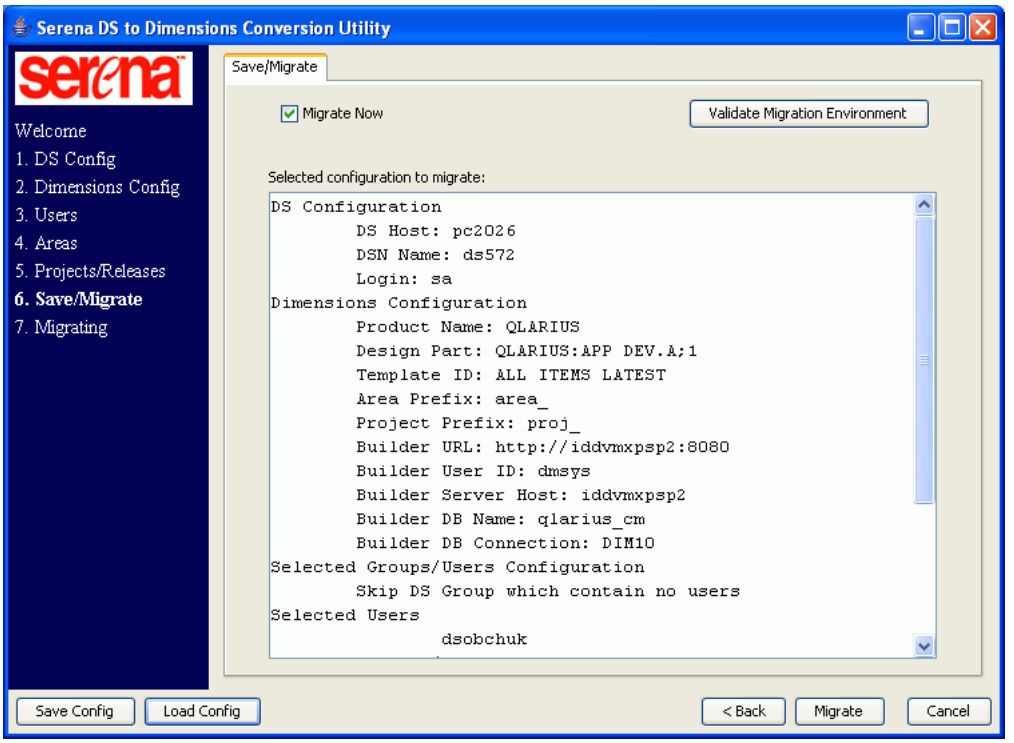
- If you want to use the default project types for migrated projects, select **Use defaults**.
- If you want to set Dimensions CM project types for specific projects, deselect **Use defaults**. For each project for which you want to set a project type:
  - Select the project in the list
  - Select the type from the **Associate project with corresponding type** list.

21 Click the Releases tab.

For further details of this tab, see "Projects/Releases Page Releases Tab" on page 191.

22 Click **Next**.

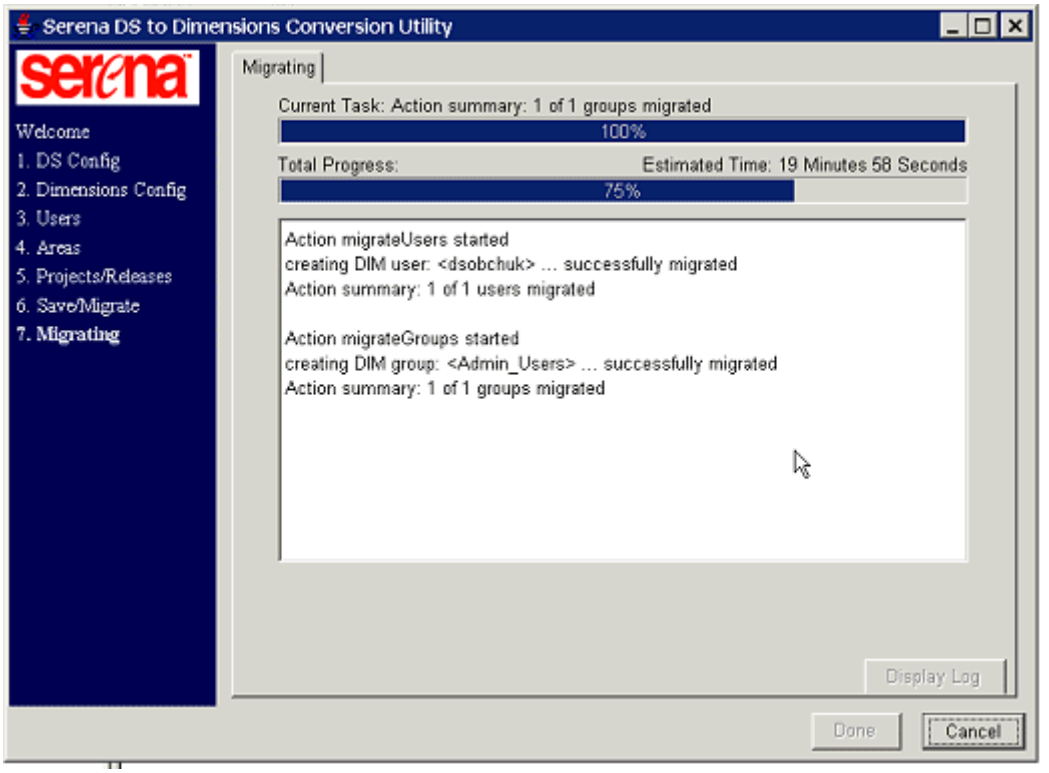
The Save/Migrate page appears displaying the details of the migration you have configured.



23 Check the Migration details displayed.

- If you want to change any of the details, use the **Back** button or click the link to the page on the left-hand side to return to the required page(s) and change them.
- If you want to save these configuration details, click **Save Config** and use the browse button to select a location for the configuration file.
- If you want to perform a test of the migration without actually migrating anything, click **Validate Migration Environment**. After a pause, messages will be displayed informing you of the results of the validation.

24 If you want to start the migration click **Migrate**.

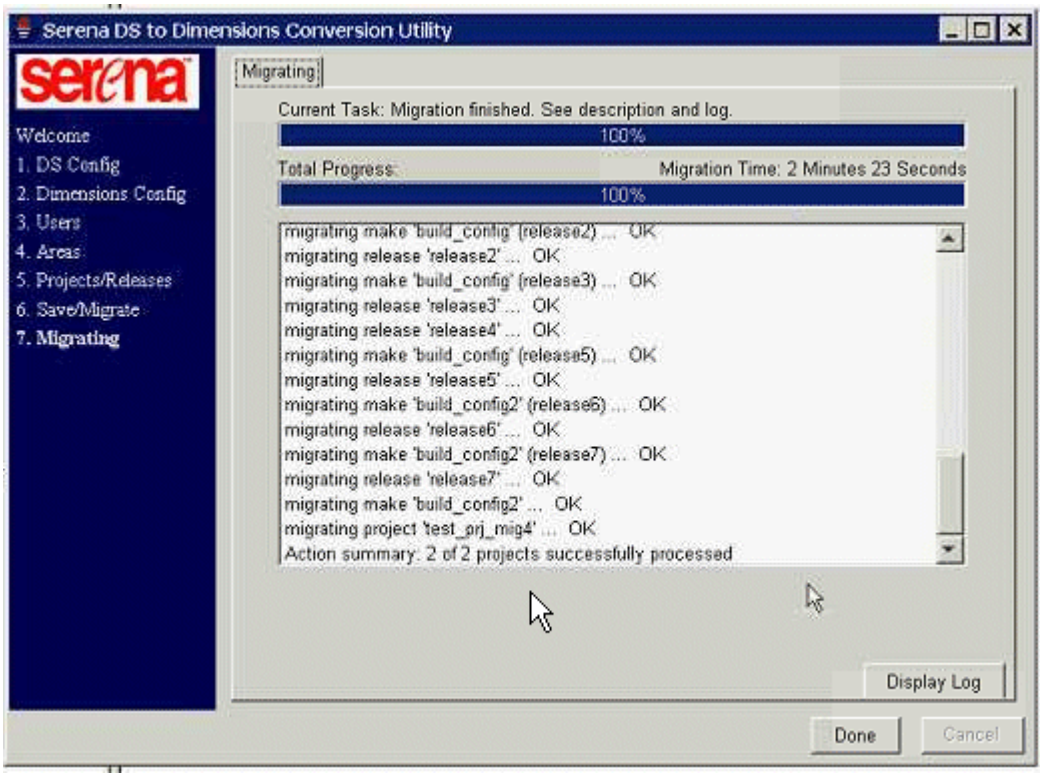


The Migrating page is displayed showing progress bars. Messages are displayed showing the results of the migration as they occur.

If you want to cancel the migration process before it has completed, click **Cancel**. The process will be aborted. Any migrations of projects will be rolled back.

If a failure occurs at any point, text will be displayed in red. If this has occurred as part of the migration of a project, the updates will be rolled back.

When the migration process has completed, a line labeled "Action summary" is displayed showing the outcome of the migration.



To view the log for the migration, click **Display Log**. Note that entries for this run will be appended to the end of the log after any entries for previous runs of the migration.

To exit the utility, click **Done**.

Finding Migrated Objects in Dimensions CM

You can use the web client or desktop client, and the Administration Console to check how objects have been migrated to Dimensions CM.

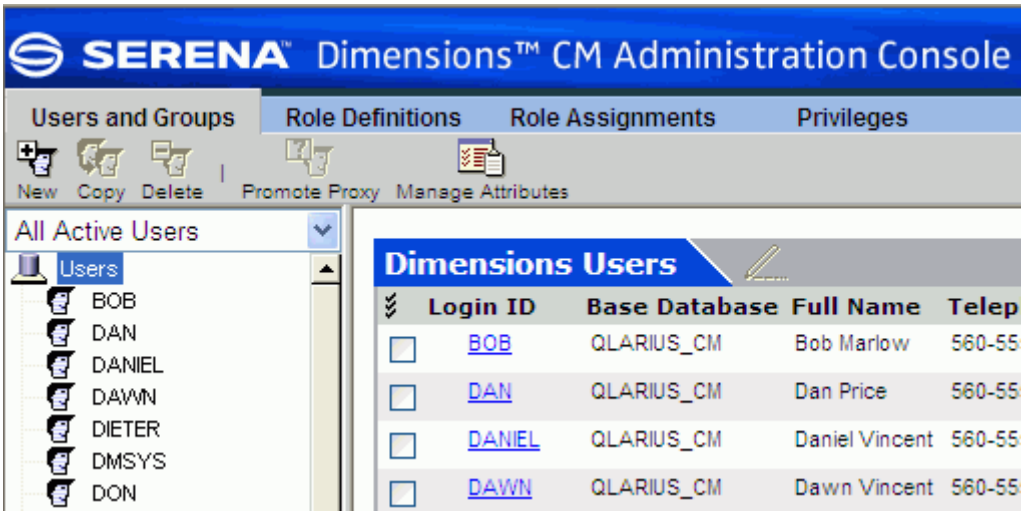
For details of the Administration Console see the Dimensions CM Process Configuration Guide.

For details of the web client and desktop client, see the Dimensions CM User's Guide

Users and Groups

To view migrated users and groups, open the Administration Console.

From the main window, select Users and Roles | User and Group Registration.



The drop down list shows all active users, selecting one of these users in the navigation area displays the details of the user.

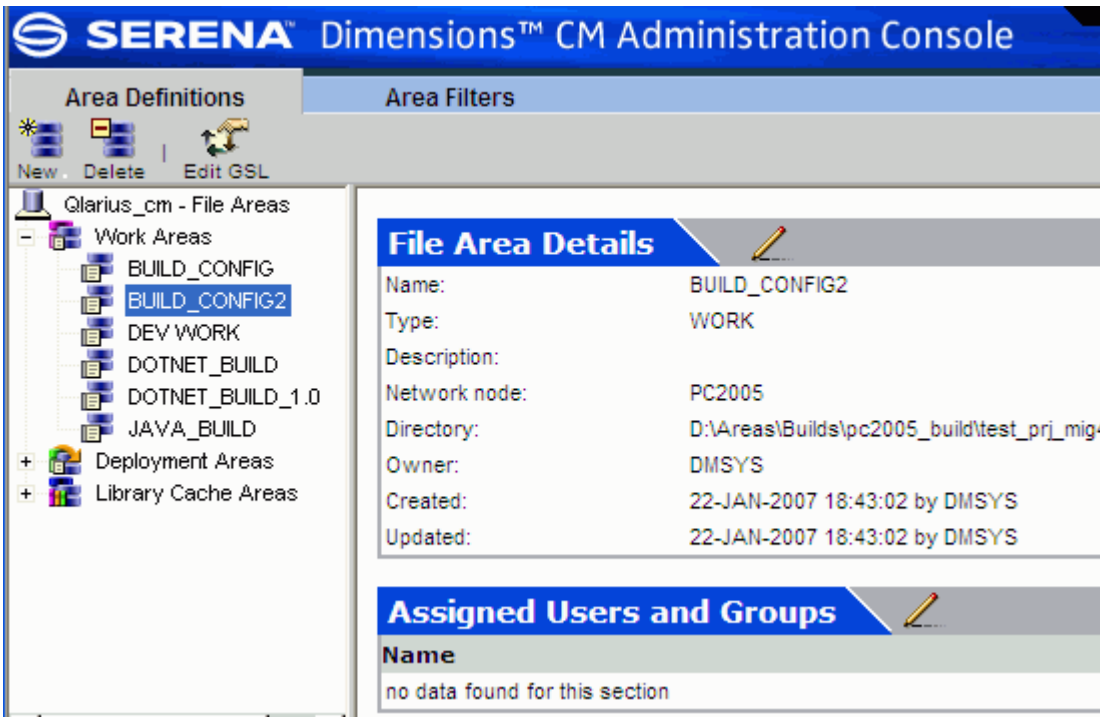
To view the groups, select **All Groups** from the drop down list.

For further details about areas in Dimensions CM, see the Process Configuration Guide or the online help in the Administration Console.

Areas

To view areas, from the main window, select Distributed Development | Area Definitions.

Expand the node for the type of area

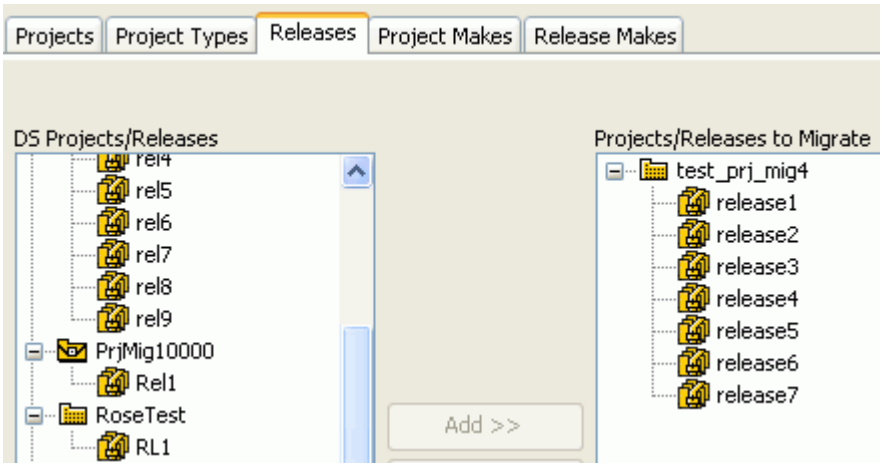


Selecting the icon for an individual area displays its details and the users and groups assigned to it. If a ChangeMan DS production area has been migrated, there will also be a new project created containing all of its files.

For further details about areas in Dimensions CM, see the Process Configuration Guide Area Definitions chapter or the online help in the Administration Console.

Projects and Releases

A project and its frozen releases, as in the example below:



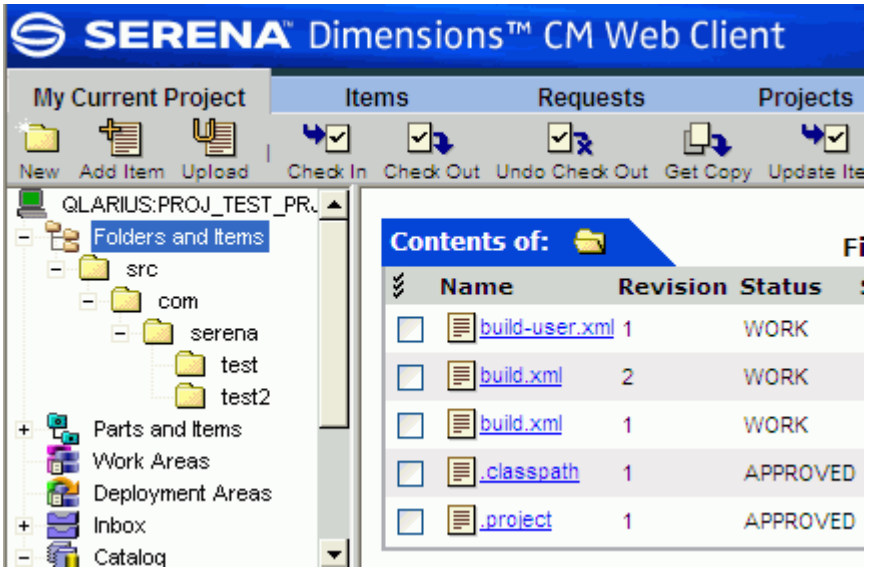
are migrated as a project and related baselines.

To view a migrated project:

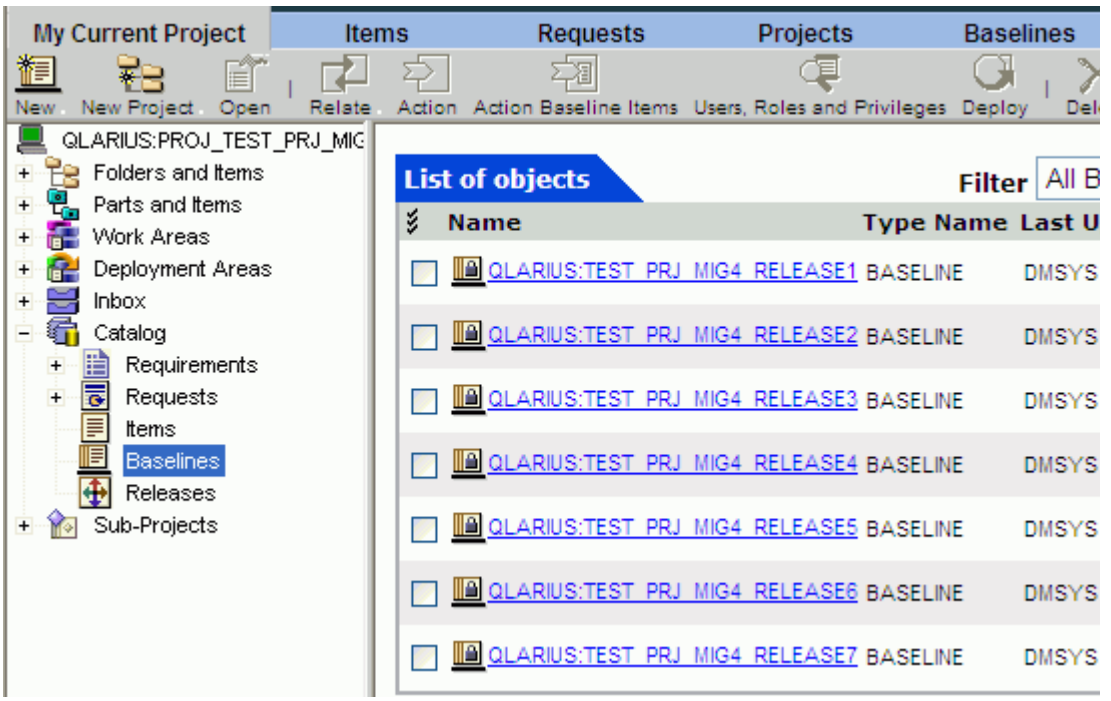
1. Start the web client.
2. Click the project link on the status bar at the bottom of the window:
   

3. In the Set Current Project dialog box, select the project from the list and click OK.
4. Click the My Current Project tab to see the details of the project.

5 Expand the **Folders and Items** node to view the migrated files.



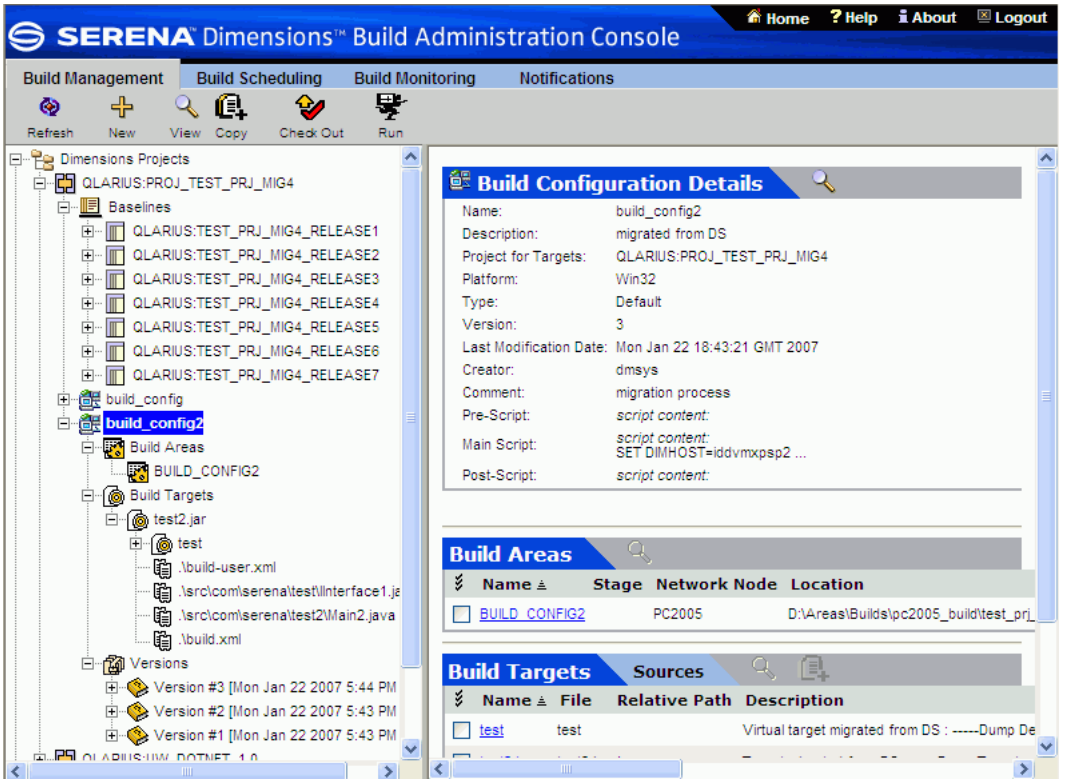
6 To view the migrated frozen releases for that project, expand the **Catalog** node, and select **Baselines**.



1 Start the Administration Console, and select Distributed Development | Build Administration.

2 On the Build Management tab, expand the Dimensions Projects tree and select the project.

Expanding the build configuration node reveals the build areas and targets.



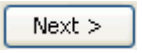
For further details of Dimensions Build Administration, see the *Build Tools User's Guide*.

The Conversion Utility

The Dimensions CM is a wizard that consists of a number of pages, some of these pages have a number of tabs. These are described in the following sections:

- Welcome page, (described here). This page appears when you first launch the utility.
- "DS Config Page" on page 186.
- "Dimensions Config Page" on page 186.
- "Users Page" on page 187.
- "Areas Page" on page 189.
- "Projects/Releases Page" on page 190.
- "Save/Migrate Page" on page 192.
- "Migrating Page" on page 192.

The conversion utility has the following buttons at the bottom of the pages:

Click this button	To ...
	Save the configuration settings that you currently have set. This opens a browse dialog enabling you to save the configuration settings to a file. The file is saved in XML format.
	Load the configuration settings from a selected file. This opens a browse dialog enabling you to retrieve configuration settings from a file you previously saved using the **Save Config** button.
	Return to the previous page.
	Continue to the next page.
	Cancel the migration and quit the conversion utility. If this is the Migrating page, and the migration has not yet completed migrating a project, the changes will be rolled back.

You can also click the link for a particular page on the left-hand side to go to that page.

DS Config Page

Use the DS Migration Config page to specify the server and database connection details for the ChangeMan DS server to be accessed for the migration.

DS Migration Config Tab

Field	Description/Rules and Guidelines
DS Host	Type the name of the ChangeMan DS server from which you want to migrate.
Test DS	Click this button to check that the server exists on the network.
DSN Name	The DSN name of the ChangeMan DS data source.
Select DB	Click this button to select a ChangeMan DS source from the host machine.
Login	The Login ID for the server.
Password	The password for the login above.
Test DB	Click this button to check that the selected server can be connected to using this user login and password.

Dimensions Config Page

Use the Dimensions Config page to specify the Dimensions CM server and database connection details and the product and design part to which the ChangeMan DS objects are to be migrated.

Dimensions Migration Configuration Tab

Field	Description/Rules and Guidelines
User ID	Type a user ID that is registered on the Dimensions CM server.
Password	Type the password for the user ID that you entered.
Server Host	Type the name of the Dimensions CM server.
DB Name	Type the name of the Dimensions CM database.
DB Connection	Type a connection string for the Dimensions CM database.
Product Name	Select the name of the Dimensions CM product to which the migrated projects will belong.
Load	Click this button to populate the product list with the products for the database
Design Part	Select the design part to which the migrated products will belong.
Use Area Prefix	Use a prefix to be used for the names of the migrated areas in Dimensions CM (enter a prefix).
Use Project Prefix	Use a prefix to be used for the names of the migrated projects in Dimensions CM (enter a prefix).

Users Page

Use the Users page to determine how you want ChangeMan DS users and groups to be migrated to Dimensions CM. It consists of a group of tabs, which are described in the following sections:

- "Users Page Config Tab" on page 187.
- "Users Page Groups Tab" on page 188.
- "Users Page Users Tab" on page 188.

Users Page Config Tab

Use the Config tab of the Users page to determine which users or groups are to be migrated.

Config Tab

Field	Description/Rules and Guidelines
Migration Process	The method to be used to migrate users
Migrate All	Migrate all DS users
Migrate by Ownership	Migrate users and groups from the ChangeMan DS server to the Dimensions CM server based on the ownership and authorization lists of migrated areas and projects.
Configure Migration	Configure the migration of users according to the options below and the specific selections made on the Groups tab and Users tab.
Skip DS groups which contain no users	Select this option if you do not want empty DS groups to be migrated.

Users Page Groups Tab

Use the Groups tab of the Users page to select the ChangeMan DS groups to be migrated. This tab is disabled unless you have selected **Configure Migration** on the Config tab.

Groups Tab

Field	Description/Rules and Guidelines
DS Groups	The available DS groups to be migrated.
Groups to Migrate	The DS groups that are selected to be migrated.
Add >>	Click this button to move a group from the **DS Groups** list to the **Groups to Migrate** list.
<< Remove	Click this button to remove a group from the **Groups to Migrate** list to the **DS Groups** list.
Migrate all DS users within selected groups	Select this option to automatically migrate all the users that belong to the selected groups.

Users Page Users Tab

Use the Users tab of the Users page to select individual ChangeMan DS users to be migrated. This tab is disabled unless you have selected **Configure Migration** on the Config tab.

Users Tab

Field	Description/Rules and Guidelines
DS Users	The available DS users to be migrated.
Users to Migrate	The DS users that are selected to be migrated.
Add >>	Click this button to move a user from the **DS Users** list to the **Users to Migrate** list.
<< Remove	Click this button to remove a group from the **Users to Migrate** list to the **DS Users** list.

Areas Page

Use the Areas page to determine how you want ChangeMan DS areas to be migrated to Dimensions CM. It consists of a group of tabs, which are described in the following sections:

- "Areas Page Config Tab" on page 189.
- "Areas Page Areas Tab" on page 189.
- "Areas Page Stages Tab" on page 190.

Areas Page Config Tab

Use the Config Tab of the Areas Page to determine the way in which areas are to be migrated.

ConfigTab

Field	Description/Rules and Guidelines
Migrate All	Migrate all the areas from the DS server.
Configure Migration	Migrate areas according to the configuration below and the selections on the Areas tab.
Migrate only last versions	Migrate only the latest versions of files in these areas.
Number of versions	Migrate this number of latest versions. (Enter an integer.)

Areas Page Areas Tab

Use the Areas tab of the Areas page to determine which specific areas you want to migrate.

Areas Tab

Field	Description/Rules and Guidelines
DS Areas	The available DS areas to be migrated.
Areas to Migrate	The DS areas that are selected to be migrated.
Add >>	Click this button to move an area from the **DS Areas** list to the **Areas to Migrate** list.
<< Remove	Click this button to remove an area from the **Areas to Migrate** list to the **DS Areas** list.

Areas Page Stages Tab

Use the Stages tab of the Areas page to choose which deployment stage in Dimensions CM to associate with the types of ChangeMan DS area being migrated.

Note that ChangeMan DS development areas are always migrated as work areas and do not have a stage associated with them.

Stages Tab

Field	Description/Rules and Guidelines
Use defaults	Select this option to use the default stages.
Associate ChangeMan DS QA areas with	Select the Dimensions CM stage with which to associate DS QA areas.
Associate ChangeMan DS End User areas with	Select the Dimensions CM stage with which to associate DS End User areas.
Associate ChangeMan DS Production areas with	Select the Dimensions CM stage with which to associate DS Production areas.

Projects/Releases Page

Use the Projects/Releases page to determine how you want ChangeMan DS projects, frozen releases,

- "Products/Releases Page Config Tab" on page 190.
- "Projects/Releases Page Projects Tab" on page 191.
- "Projects/Releases Page Project Types Tab" on page 191.
- "Projects/Releases Page Releases Tab" on page 191.

Products/Releases Page Config Tab

Use the Config Tab of the Products/Releases Page to determine.

ConfigTab

Field	Description/Rules and Guidelines
Migrate All	Select this option if you want to migrate all of the projects from the DS server.
Configure Migration	Select this option if you want to configure specific options for migrating projects.
Migrate only attached versions	Select this option if you want to migrate only those versions of a file that have been attached to a migrated project.
Migrate all frozen releases with project	Select this option if you want to migrate all frozen releases for any project that is migrated.

Projects/Releases Page Projects Tab

Use the Projects Tab of the Products/Releases Page to determine which projects and frozen releases are migrated.

Projects Tab

Field	Description/Rules and Guidelines
DS Projects	The available DS projects to be migrated.
Projects to Migrate	The DS projects that are selected to be migrated.
Add >>	Click this button to move a project from the DS Projects list to the Projects to Migrate list.
<< Remove	Click this button to remove a project from the Projects to Migrate list to the DS Projects to list.

Projects/Releases Page Project Types Tab

Use the Project Types tab of the Products/Releases page to specify a project type for the new Dimensions CM projects created from the migrated ChangeMan DS projects.

Project Types Tab

Field	Description/Rules and Guidelines
Use defaults	Select this option if you want to use the default project type for migrated DS projects.
Associate project with corresponding type.	Select a project type for the project selected in the list.

Projects/Releases Page Releases Tab

Use the Projects tab of the Products/Releases page to determine which frozen releases to migrate to Dimensions CM baselines.

Releases Tab

Field	Description/Rules and Guidelines
DS Projects/ Releases	The available DS releases to be migrated.
Projects/Releases to Migrate	The DS releases that are selected to be migrated.
Add >>	Click this button to move a release from the DS Projects/Releases list to the Projects/Releases to Migrate list.
<< Remove	Click this button to remove a project from the Projects/Releases to Migrate list to the DS Projects/Releases list.

Save/Migrate Page

Use the Save/Migrate page to view the configuration settings you have selected, to save the configuration, or to test or start the migration process.

Save/Migrate Tab

Field	Description/Rules and Guidelines
Validate Migration Environment	Click this button to test the migration process and produce a log file without actually performing the migration.
Selected configuration to migrate	This area displays the details of the configuration you have selected to use for the migration.

Migrating Page

Use the Migrating Page to view the progress of the migration, to cancel the process, or to view the log when the process has completed.

Migrating Tab

Field	Description/Rules and Guidelines
Current Task Action Summary	A summary of the migration progress and a progress bar.
Task Progress	The estimated time for the total migration and a progress bar.
Display Log	Click this button to display a log of the migration. This option is only available when the migration has completed.
Done	Click this button to quit the utility. This option is only available when the migration has completed.
Cancel	Click this button when the migration is running to abort the migration and roll back the changes. This option is only available when the migration has not yet completed.

Chapter 20

Migrating ChangeMan DS Command Line Scripts

Overview

The ChangeMan DS migration utility automatically migrates ChangeMan DS command line scripts so that they will continue to work in Dimensions CM. The migration utility accomplishes this by wrapping ChangeMan DS command syntax with equivalent Dimensions CM command syntax. Many ChangeMan DS commands and parameters can be mapped to comparable commands and parameters in Dimensions CM. However, some ChangeMan DS commands have no equivalent in Dimensions CM.

Many commands and parameters do not have a one-to-one equivalent in Dimensions CM; these may be mapped to a series of Dimensions CM commands that then approximate the same functionality.

This chapter provides information on the specific ChangeMan DS commands and parameters that are mapped to commands and parameters in Dimensions CM. For detailed information on the Dimensions CM command line, see the *Serena Dimensions CM Command-Line Reference*.

Migrating Areas Commands

The migration console wraps ChangeMan DS areas commands with the equivalent Dimensions CM commands.

Area Types Mappings

The migration console maps area types in ChangeMan DS to the equivalent area types in Dimensions CM:

ChangeMan DS	Dimensions CM
Development	Work
QA	Deployment
Production	Deployment
End User	Deployment

Creating Network Nodes

The migration console uses the Dimensions CM `CNN` (create network node) command to create network nodes for each of the network locations where an area resides.

Creating Areas

The -actcreate command in ChangeMan DS maps to the CA (create area) command in Dimensions CM. Parameters map as follows:

ChangeMan DS	Dimensions CM
area	*area-name*
areahost	NETWORK_NODE
areapath	DIRECTORY
desc	DESCRIPTION
areatype Values are: p (production area) d (development area) t (QA area) e (end user area)	TYPE Mapped values are: DEPLOYMENT WORK DEPLOYMENT DEPLOYMENT If the value is DEPLOYMENT, a STAGE parameter specifies the stage that the area is associated with.
user	USER
password	PASSWORD
Not applicable	TRANSFER_SCRIPTS If the area is a DEPLOYMENT type of area, this parameter defines the pre, post, and fail scripts. The scripts wrap Dimensions CM command line syntax around the originating ChangeMan DS scripts.
areaowner	OWNER
addauth	USER_LIST
Not applicable	STATUS If the area is a DEPLOYMENT type of area, this parameter is set to default value of ONLINE.
deletefrom	NOKEEP This qualifier can only be defined for DEPLOYMENT areas.
Not applicable	KEEP This qualifier can only be defined for DEPLOYMENT areas

Updating Areas

The -actedit command in ChangeMan DS maps to the UA (update area) command in Dimensions CM. Parameters map as follows:

ChangeMan DS	Dimensions CM
area	*area-name*
Not applicable	NEW_NAME
Not applicable	NETWORK_NODE
Not applicable	DIRECTORY

ChangeMan DS	Dimensions CM
desc	DESCRIPTION
areatype Values are: p (production area) d (development area) t (QA area) e (end user area)	TYPE Mapped values are: DEPLOYMENT WORK DEPLOYMENT DEPLOYMENT If the value is DEPLOYMENT, a STAGE parameter specifies the stage that the area is associated with.
user	USER
password	PASSWORD
Not applicable	TRANSFER_SCRIPTS If the area is a DEPLOYMENT type of area, this parameter defines the pre, post, and fail scripts. The scripts wrap Dimensions CM command line syntax around the originating ChangeMan DS scripts.
addauth delauth	USER_LIST
Not applicable	ADD Used with the USER_LIST parameter for adding authorized users and groups to the area.
Not applicable	DELETE Used with the USER_LIST parameter for deleting authorized users and groups from the area.
Not applicable	STATUS If the area is a DEPLOYMENT type of area, this parameter is set to default value of ONLINE.
deletefrom	NOKEEP This qualifier can only be defined for DEPLOYMENT areas.
Not applicable	KEEP This qualifier can only be defined for DEPLOYMENT areas
Not applicable	FILTER This qualifier can only be defined for DEPLOYMENT areas

Listing Areas

The -actlist command in ChangeMan DS maps to the LA (list areas) command in Dimensions CM. Parameters map as follows:

ChangeMan DS	Dimensions CM
area	area-name
Not applicable	WORKSET

ChangeMan DS	Dimensions CM
areatype Values are: p (production area) d (development area) t (QA area) e (end user area)	TYPE Mapped values are: DEPLOYMENT WORK DEPLOYMENT DEPLOYMENT If the value is DEPLOYMENT, a STAGE parameter specifies the stage that the area is associated with.
areahost	NETWORK_NODE
Not applicable	OWNER
Not applicable	STATUS If the area is a DEPLOYMENT type of area, this parameter is set to default value of ONLINE.

Deleting Areas

The -actdelete command in ChangeMan DS maps to the RA command in Dimensions CM. Parameters map as follows:

ChangeMan DS	Dimensions CM
area	area-name

Migrating Project Commands

Where applicable, ChangeMan DS project commands are migrated to equivalent Dimensions CM commands. The -cmnproj command in ChangeMan DS is wrapped with the equivalent project command in Dimensions CM, such as DWS, SWS, RWWS, SWF, SCWS, AIWS, RIWS, RWS, and RENAME. Where a one to one mapping is not possible, the DS commands are mapped to a sequence of commands that provide comparable functionality in Dimensions CM.

Generating a List of Projects

The -actlist action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>
- -outfield: <*output fields n, d, o, r, s*>
- -projestatus: <*status*>
- -listattach: <*attach fields n, v, r*>
- -outfile: <*file*>
- -header

- -subheader
- -recordsep: <*separator*>
- -fieldsep: <*separator*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -listauth
- -listrelease: <*release fields*>
- -projtype: <*type*>
- -projpriority: <*priority*>
- -nlreplace: <*string*>
- -listindent: <*string*>

Creating Projects

The -actcreate action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>
- -projestatus: <*status*>
- -desc: <*description*>
- -projnote: <*note*>
- -projnotefile: <*notefile*>
- -projparent: <*parent*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -projowner: <*user*>
- -projtype: <*type*>
- -projpriority: <*priority*>
- -addauth: <*user*>
- -projpermfile: <*file name*>

Editing Projects

The -actedit action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>
- -newname: <*new name*>
- -projestatus: <*status*>
- -desc: <*description*>

- -projnote: <*note*>
- -projnotefile: <*notefile*>
- -projparent: <*parent*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -projowner: <*user*>
- -projtype: <*type*>
- -projpriority: <*priority*>
- -delauth: <*user*>
- -addauth: <*user*>
- -projpermfile: <*file name*>

Deleting Projects

The -actdelete action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>

Copying Lists Between Projects

The -actcopy action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project1: <*target project name*>
- -project2: <*source project*>
- -copyattach
- -copyreplace

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -copyauth

Attaching and Detaching Files

The -actattach action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>
- -homearea: <*area*>
- -homepath: <*path*>
- -file: <*file name*>
- -recurse

- -relpathprefix: <*prefix*>
- -projattfile: <*file*>
- -relpath: <*relative path*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -homearea <*area*>

Comparing Projects

The -actdiff action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -productID: <*product*>
- -project1: <*project name*>
- -project2: <*project name*>
- -release1: <*release name*>
- -release2: <*release name*>
- -diffoption: <*May be:*
  - *n: Compare new files in project 2*
  - *d: Compare deleted files in project 2*
  - *m: Compare files with different versions or relative paths in project 2:*
  - *s: Compare files unchanged in project 2*>
- -outfields: <*May be:*
  - *n: Original name*
  - *r: Original relative path*
  - *v: Original version*
  - *o: Modified name*
  - *s: Modified relative path*
  - *w: Modified version*
  - *t: How the projects differ*
  - *a: Original area*
  - *p: Original path*
  - *d: Original description*
  - *b: Modified area*
  - *q: Modified path*
  - *e: Modified description*>
- -outfile: <*file*>

- -header
- -recordsep: <*separator*>
- -fieldsep: <*separator*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -homearea <*area*>

Freezing Projects

The -actfreeze action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -project: <*project name*>
- -release: <*release*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -desc <*description*>

Listing Frozen Release Attachments

The -actlistrelease action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -release: <*release*>
- -outfields: <*May be:*
  - *n: File name*
  - *v: Production version*
  - *r: Relative path*
  - *a: Production area*
  - *p: Production path*>
- -outfile: <*file*>
- -header
- -recordsep: <*separator*>
- -fieldsep: <*separator*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

-desc <*description*>

Editing Project Releases

The -acteditrelease action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -DSdata
- -project: <*projname*>
- -release: <*release*>
- -newname: <*release name*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

-desc <*description*>

Attaching, Detaching, Editing Attachments From a Frozen Release

The -actattachrelease action of the -cmnproj command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -release: <*release*>
- -relpath: <*relative path*>
- -newrelpath: <*relative path*>
- -newrelpathprefix: <*relative path prefix*>
- -filever: <*file version, mapped when attaching files*>
- -homearea: <*area*>
- -homepath: <*path*>
- -file: <*file name*>
- -recurse
- -relpathprefix: <*prefix*>
- -projattfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -desc <*description*>
- -filever: <*file version, ignored when editing attachments*>

Migrating User Commands

Where applicable, ChangeMan DS user and group commands are migrated to equivalent Dimensions CM commands. The -cmnuser command in ChangeMan DS is wrapped with the equivalent user command in Dimensions CM, such as UREG, RREG, XREG, UUA, AUR, GGRP, AUGRP, AGRPU, UGRP, DGRP, and PRIV. Where a one to one mapping is not possible, the DS commands are mapped to a sequence of commands that provide comparable functionality in Dimensions CM.

General Notes on Parameters and Options

- <login> parameters are always ignored.
- -verbose displays additional information on all actions.
- -outfile: <*file name*> logs errors and results in the specified file, for all actions.
- The -help, -displayparams, and -interact parameters are not implemented in the mapped actions.
- The return code for each wrapper is 1 when successful, and -1 when failed. If execution fails the entire operation is canceled.

Generating a List of Users and Groups

The -actlist action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -username: <*user name*>
- -groupname: <*group name*>
- -output fields: <*May be:*
  - *n: User / group name*
  - *d: Group description*
  - *f: User's full name*
  - *a: Mail address*
  - *s: Mail server*
  - *Additional output fields are ignored, see below*>
- -listusers
- -listgroups
- -listmember
- -listperm: <*permissions; some may be ignored*>
- -outfile: <*file*>
- -header
- -subheader
- -recordsep: <*separator*>
- -fieldsep: <*separator*>
- -nlreplace: <*nlreplace*>
- -listindent: <*listindent*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -listrestrict: <*restrictions*>
- -output fields: <*Ignored output fields are t, c, l, h, u, o*>

Creating Users

The -actcreateuser action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -username: <*user name*>
- -userpsw: <*password*>
- -fullname: <*full name*>
- -mailaddress: <*mailing address*>
- -addmember: <*groupname*>
- -filepermdef: <*default*>
- -filepermfile: <*file*>
- -projpermdef: <*default*>
- -projpermfile: <*file*>
- -auditpermdef: <*default*>
- -auditpermfile: <*file*>
- -adminpermdef: <*default*>
- -adminpermfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -mailserver: <*server*>
- -valuseros
- -agenthost: <*host*>
- -valaltuser: <*user*>
- -domainname: <*domain*>
- -trsfmdtrfld: <*fields*>
- -allowdircrt
- -trustnl
- -restrictbyip
- -addip: <*IP address*>
- -restrictbynl
- -addnl: <*login name*>
- -lytpermdef: <*default*>
- -lytpermfile: <*file*>
- -clntpermdef: <*default*>
- -clntpermfile: <*file*>
- -setclnttmplt: <*template*>

Editing Users

The -actedituser action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -username: <*user name*>
- -userpsw: <*password*>
- -fullname: <*full name*>
- -mailaddress: <*mailing address*>
- -newname: <*new name*>
- -addmember: <*groupname*>
- -delmember: <*groupname*>
- -filepermdef: <*default*>
- -filepermfile: <*file*>
- -projpermdef: <*default*>
- -projpermfile: <*file*>
- -auditpermdef: <*default*>
- -auditpermfile: <*file*>
- -adminpermdef: <*default*>
- -adminpermfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -mailserver: <*server*>
- -valuseros
- -agenthost: <*host*>
- -valaltuser: <*user*>
- -domainname: <*domain*>
- -trsfmdtrfld: <*fields*>
- -allowdircrt
- -trustnl
- -restrictbyip
- -addip: <*IP address*>
- -delip: <*IP address*>
- -restrictbynl
- -addnl: <*login name*>
- -delnl: <*login name*>
- -lytpermdef: <*default*>
- -lytpermfile: <*file*>

- -clntpermdef: <*default*>
- -clntpermfile: <*file*>
- -setclnttmplt: <*template*>

Creating Groups

The -actcreategroup action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -groupname: <*group name*>
- -desc: <*description*>
- -addmember: <*group name*>
- -filepermdef: <*default*>
- -filepermfile: <*file*>
- -projpermdef: <*default*>
- -projpermfile: <*file*>
- -auditpermdef: <*default*>
- -auditpermfile: <*file*>
- -adminpermdef: <*default*>
- -adminpermfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -valuseros
- -agenthost: <*host*>
- -valaltuser: <*user*>
- -domainname: <*domain*>
- -trsfmdtrfld: <*fields*>
- -allowdircrt
- -trustnl
- -restrictbyip
- -addip: <*IP address*>
- -restrictbynl
- -addnl: <*login name*>
- -lytpermdef: <*default*>
- -lytpermfile: <*file*>
- -clntpermdef: <*default*>
- -clntpermfile: <*file*>
- -setclnttmplt: <*template*>

Editing Groups

The -acteditgroup action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -groupname: <*group name*>
- -newname: <*new group name*>
- -desc: <*description*>
- -addmember: <*user name*>
- -delmember: <*user name*>
- -filepermdef: <*default*>
- -filepermfile: <*file*>
- -projpermdef: <*default*>
- -projpermfile: <*file*>
- -auditpermdef: <*default*>
- -auditpermfile: <*file*>
- -adminpermdef: <*default*>
- -adminpermfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -valuseros
- -agenthost: <*host*>
- -valaltuser: <*user*>
- -domainname: <*domain*>
- -trsfmdtrfld: <*fields*>
- -allowdircrt
- -trustnl
- -restrictbyip
- -addip: <*IP address*>
- -restrictbynl
- -addnl: <*login name*>
- -delnl: <*login name*>
- -lytpermdef: <*default*>
- -lytpermfile: <*file*>
- -clntpermdef: <*default*>
- -clntpermfile: <*file*>
- -setclnttmplt: <*template*>

Deleting Users or Groups

The -actdelete action of the -cmnuser command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -groupname: <*group name*>
- -username: <*user name*>

Migrating File Transfer Commands

Where applicable, ChangeMan DS file transfer commands are migrated to equivalent Dimensions CM commands. The -cmnxfer command in ChangeMan DS is wrapped with the equivalent transfer command in Dimensions CM. Where a one to one mapping is not possible, the DS commands are mapped to a sequence of commands that provide comparable functionality in Dimensions CM.

General Notes on Parameters and Options

- <login> parameters are always ignored.
- -verbose displays additional information on all actions.
- -outfile: <*file name*> logs errors and results in the specified file, for all actions.
- The -help, -displayparams, and -interact parameters are not implemented in the mapped actions.
- The return code for each wrapper is 1 when successful, and -1 when failed. If execution fails the entire operation is canceled.

Performing Transfers

The -acttransfer action of the -cmnxfer command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -file: <*file name*>
- -filever: <*file version*>
- -fromarea: <*source area*>
- -frompath: <*source path*>
- -toarea: <*target area name*>
- -topath: <*target path*>
- -project: <*project name*>
- -readonly
- -desc: <*description*>
- -transfertype: <may be fh, prh>

- -auditpermdef: <*default*>
- -auditpermfile: <*file*>
- -adminpermdef: <*default*>
- -adminpermfile: <*file*>

The following parameters are not mapped, as there are no equivalents in Dimensions CM:

- -valuseros
- -agenthost: <*host*>
- -valaltuser: <*user*>
- -domainname: <*domain*>

Checking Out

The -actcheckout action of the -cmnxfer command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -file: <*file name*>
- -filever: <*file version*>
- -fromarea: <*area*>
- -frompath: <*path*>
- -project: <*project name*>
- -readonly
- -desc: <*description*>
- -list: <*list of files*>

Checking In

The -actcheckin action of the -cmnxfer command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- -file: <*file name*>
- -filever: <*file version*>
- -toarea: <*area name*>
- -topath: <*path*>
- -project: <*project name*>
- -readonly
- -desc: <*description*>
- -list: <*list of files*>

Approving and Rejecting Transfers

The `-actapprove and -actreject` actions of the `-cmnxfer` command are mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- `-file: <file name>`
- `-filever: <file version>`
- `-area: <area name>`
- `-fromarea: <area name>`
- `-frompath: <path>`
- `-project: <project name>`
- `-readonly`
- `-desc: <description>`
- `-homearea: <home area>`
- `-homepath: <home path>`
- `-toarea: <area name>`
- `-topath: <area path>`
- `-list: <list of files>`

The following parameter is ignored:

- `-nlreplace: <character>`

Canceling Check-out

The `-actcancelcheckout` action of the `-cmnxfer` command is mapped to equivalent scripts in Dimensions CM, where appropriate. The following parameters are mapped to equivalent Dimensions CM command line scripts:

- `-file: <file name>`
- `-fromarea: <area name>`
- `-frompath: <path name>`
- `-project: <project name>`