MICRO FOCUS

Dimensions CM

Getting Started Guide

Product version: 14.5.1

Last updated: December 10, 2019

Table of Contents

Chapter 1

What You can do with Dimensions CM

In this Chapter

Overview

This chapter describes what Dimensions® CM can enable your company to do. Dimensions CM is a change, configuration, build, and release management product. Dimensions CM offers major advantages in usability, performance, support for modern development methods, and much more.

Manage Applications

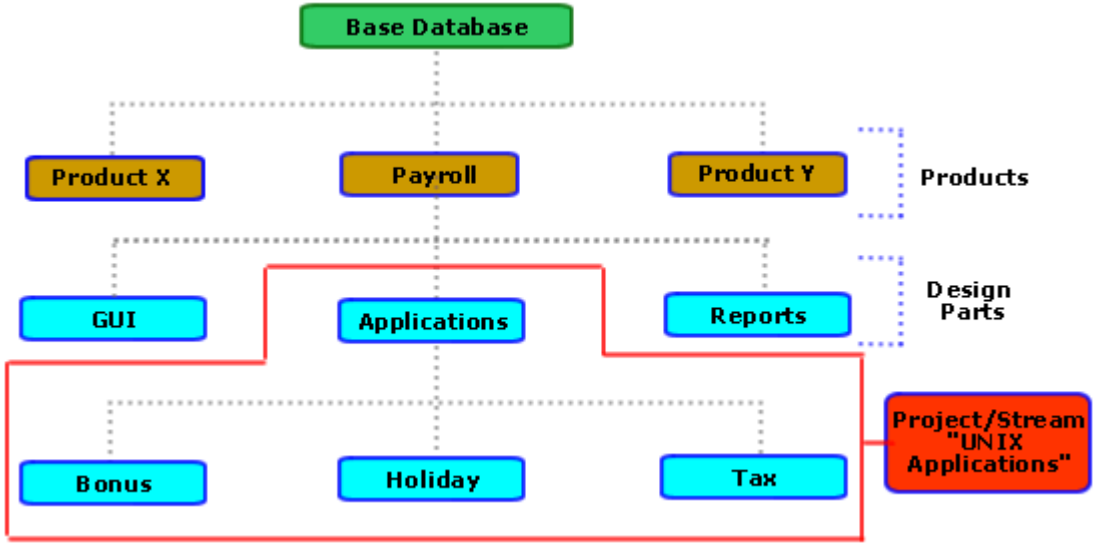
Dimensions enables you to manage assets, processes, and change across a whole enterprise. It does this by:

- Enforcing your best practices.
- Ensuring workflow through established checkpoints and approvals.
- Providing auditability and traceability.
- Providing advanced reporting and metric capabilities.
- Enabling the reuse of assets and best practices for higher productivity and efficiency.

With Dimensions you can control, track, and configure the items that comprise your application**.** These could be the files within a software product or could be other types of asset. A software product, for example, can easily consist of many thousands of hardware, software and documentation items. These are all stored under Dimensions as versioned *items*.

The projects that are typically undertaken can be configured to contain different components of your product that are developed separately by different teams, or different versions of the same component developed in parallel. You can use Dimensions *products* to contain portions of your application that are developed independently, and you can structure a product into functional subdivisions called *design parts* that can be related in a hierarchical structure. Products are contained within a base database and you can have different products within that single database. These products can be tailored to have parts of their process models defined differently, for example they could follow different

development lifecycles or have different types of item. The diagram below illustrates an example of how this might be structured.



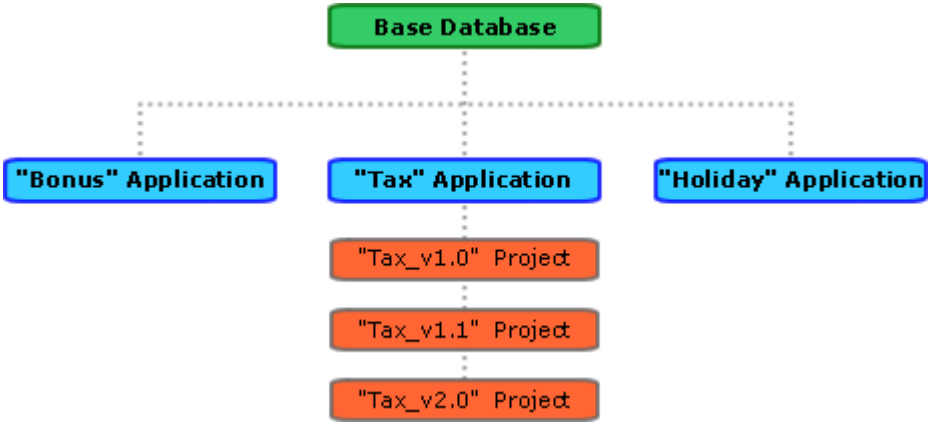
The main components are:

- **Base database**: This a basic instance of a Dimensions process model. The process model defines a collection of rules that govern the development of applications.
- **Product**: This is a major unit of development. Some of the rules in the process model can be configured differently from one product to another.
- **Design Part**: This is a logical subdivision of a product. Design parts can be subdivided into smaller parts that are all related in a hierarchy. They form a breakdown of the product into smaller functional components.
- **Item**: This is a file or other type of asset that logically belongs to a design part, such as a source file or a design document.
- **Item Revision**: This is a specific version of an item, and it is identified by its revision number.
- **Project**: This is a type of container for a part of the product that is under development. For example, it could be a UNIX version of the applications portion of the Payroll product for the Version 2 Release.

This project would contain particular versions of the items that have been developed for that purpose.

- **Stream**: A stream is a type of container for development work similar to a project, but is used to isolate work on features developed by different sprint teams. For an explanation of the difference between projects and streams, see "Projects and Streams" on page 10.

As an example, consider a company that produces a number of software products, one of which is a payroll system. The payroll system could be configured as a Dimensions product, with the company's other products configured as separate Dimensions products. Logically this product consists of three applications: Bonus, Holiday, and Tax Calculations. These three parts could be worked on by three separate project teams. They could be defined as three design parts.



When changes are made to enhance the product or fix bugs, the different versions of these items need to be tracked and controlled so that the correct versions are configured into a build of the product for testing and releasing to users and customers.

Requirements Management

You can optionally use the Requirements Management tool Dimensions RM to define requirements and link these to the components of your Dimensions products that are affected by them.

Manage the Workflow

Manage Workflow Consists of the following topics:

Lifecycles and Attributes

Items, requests, projects/streams, and other types of Dimensions objects have lifecycles associated with them to enable them to follow a path of approved states. You can define different types of objects and have them follow different lifecycles. You can set up the necessary privileges required to determine which users are allowed to *action* (promote) an object through a given state transition.

A Dimensions object can have a number of attributes that store information about it, for example creation date, status, etc. You can configure these attributes so that only certain users can update them at specified lifecycle states.

Rules and Privileges

The permissions to make changes within the database can be controlled so that different individuals have those permissions for different parts of the product, or for different types of object. You can:

- Assign users or groups of users and grant them privileges to perform certain actions, such as create, update, etc. under a given set of conditions (rules).
- Assign the authority to action objects for different stages in the lifecycle.

Inboxes and Notifications

When an item, request or other object is awaiting some action by a user it appears in their *inbox*. You can view your inboxes for each class of object in the Dimensions GUIs. You may only be able to perform certain actions on an object provided it is in your inbox.

Dimensions has a sophisticated facility for configuring email notifications that can be sent to users when certain events occur. You can determine the conditions under which an e-mail is triggered and which users will receive it. Dimensions can "Digest" multiple messages to reduce the number of emails sent.

Electronic Signatures

In some environments it may be desirable to have an electronic signature applied, either when a particular type of object is actioned to a certain state in its lifecycle, or a particular attribute of an object is updated. Dimensions has the ability to define authentication points, at which the user making the change must re-enter their user name and password in order to perform the action. There is a stored audit trail of all such attempts at authentication by users.

Control Your Assets

Projects and Streams

Dimensions allows you to use *projects* or *streams* to organize different areas of development. The files for a particular strand of development are stored in Dimensions *as items* under Version Control. Each project or stream has a folder structure in which these items are organized.

Projects and streams are designed and work in different ways; also, the Dimensions CM functions for projects and streams differ. Streams have been designed to be better suited to collaborative development using a "copy, modify, merge" process, where the developer works on a copy of the files in their work area, and then delivers the changes back to the repository. Projects are more appropriate to a "lock, modify, unlock" development process, where a developer checks out the files before making their changes.

You can configure your process model to allow users to work with: only projects, only streams, both projects and streams.

Projects

Dimensions CM projects are better suited to more traditional software development methodologies, and non-software uses such as documentation, hardware assets etc.

Projects differ from streams in that you can perform operations on individual files, such as check out and check in. You can also have parallel versions of the same code in a project because there can be more than one tip revision of the same item. You can carry out different strands of development work in the same project by associating a different branch name for versions of the same items within it. Projects can also be related in a hierarchical structure together with other parent and child projects.

Streams

Streams are designed for carrying out small amounts of change and integrating them with the main body of code before starting on a new set of changes. Streams differ from projects in that you do not perform operations on individual items in the repository, but you copy a complete set of files, make changes, and deliver them back to the repository after building and testing the code. Streams facilitate an iterative development process where the developers resolve any conflicts, and build and test the application in their work areas before committing those changes to the repository.

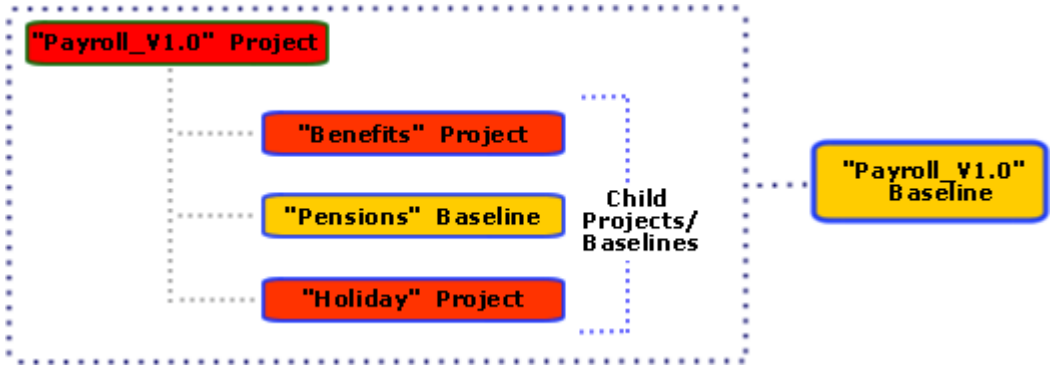
Streams enforce a single line of descent. They do not contain parallel branches of the same items, therefore it is easy to build working code from the tip revisions at any point in time.

Tip Baselines

When it has reached some state of completion or milestone, the contents of a project or stream can be frozen to preserve the versions of the items within it. You do this in Dimensions by creating tip baselines. In the case of a project, this can include one or more of its sub projects. Projects can contain baselines as well as other projects as subcomponents in their hierarchical structures.

A tip baseline:

- Captures the state of the files in the complete project hierarchy.
- Captures the relationships and paths of related components.
- Captures the design part structure that spans the files in the baseline automatically.
- In the case of projects, if other projects or baselines have child relationships to the project being baselined then they are themselves included in the baseline



When a project is created from a baseline, the original hierarchical structure is restored. You can begin a new phase of development by creating a new project from a baseline.

Optimistic Locking

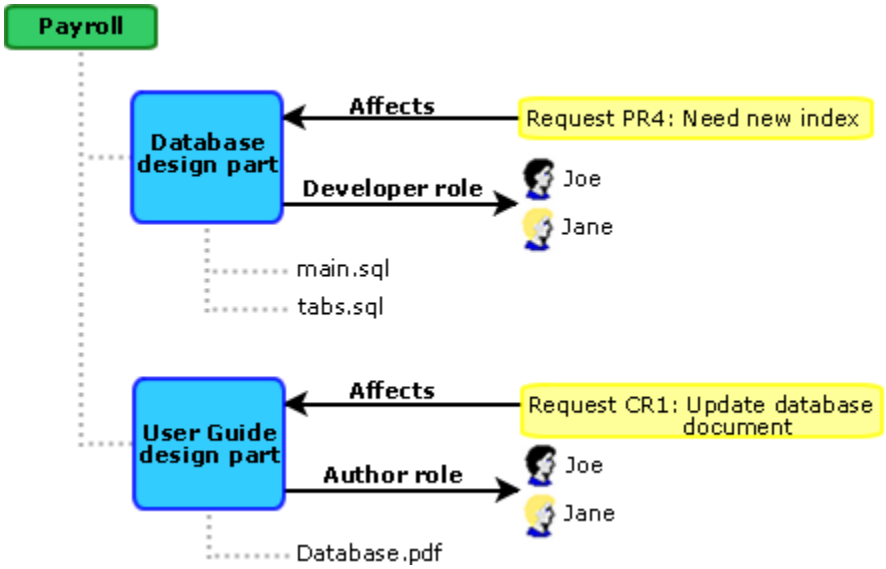
Dimensions supports optimistic locking. Optimistic locking enables you to check a file in without first having checked it out. Dimensions maintains metadata in the client working location to record the changes that have occurred in relation to the database. You can enforce security by using Change Management rules to specify that a user requires a request in order to update an item in the database.

Design Parts and Roles

Identifying the design parts of your product enables you to relate items and requests to that part of the product to which they apply and group them together.

Within parts of a product, you may want to restrict the ability to perform certain actions and changes to certain individuals within a team. Subdividing the product into design parts enables you to assign the roles for these tasks to different individuals for each component.

Design Parts and roles



Control Your Changes

Control Your Changes consists of the following topics:

Tracking Issues and Enhancements

There are two change tracking systems that you can use in Dimensions CM. You can use *requests* that are controlled within Dimensions CM and requests that are provided from Solutions Business Mashups (SBM). You track issues, bug reports, or enhancement requirements in Dimensions using these requests. You can manage the changes to items by relating groups of items and requests together. Items, i.e. versioned files, can be related to requests as *Affected*, meaning that a new version needs to be created in order to satisfy the

requirements of that request. The new versions of items that have been created are related as *In response to* the request meaning that the items are being updated to satisfy its requirement. You can relate groups of requests to a parent request, and thus track all the items involved as a set of related changes.

With Dimensions CM requests (but not with SBM requests) a frozen configuration of items can be created as a baseline by including or excluding the items related to a set of requests together with other selection criteria, such as the request's status. This means that the configuration for a particular strand of development can be recorded and recreated.

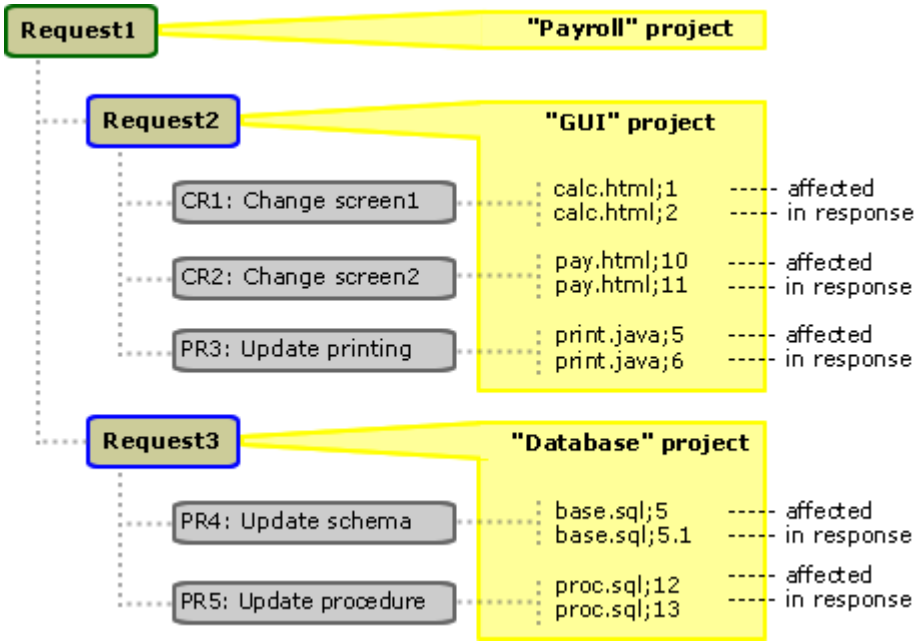
Packaging Requests

Also, with Dimensions CM requests, a useful way of including the correct items for a build is by packaging them together using a single parent request or requirement. You can also use requests to merge a set of changes from one project or stream into another.

These changes can be related to a request in a number of different ways:

- Files to be updated can be directly related to the request.
- The request can consist of other related child requests to form a hierarchy.
- The request can have various types of other requests (defects/ enhancements/issues etc.) related to it with their associated related items.

The package request also has a lifecycle to reflect the approval process for the set of changes.



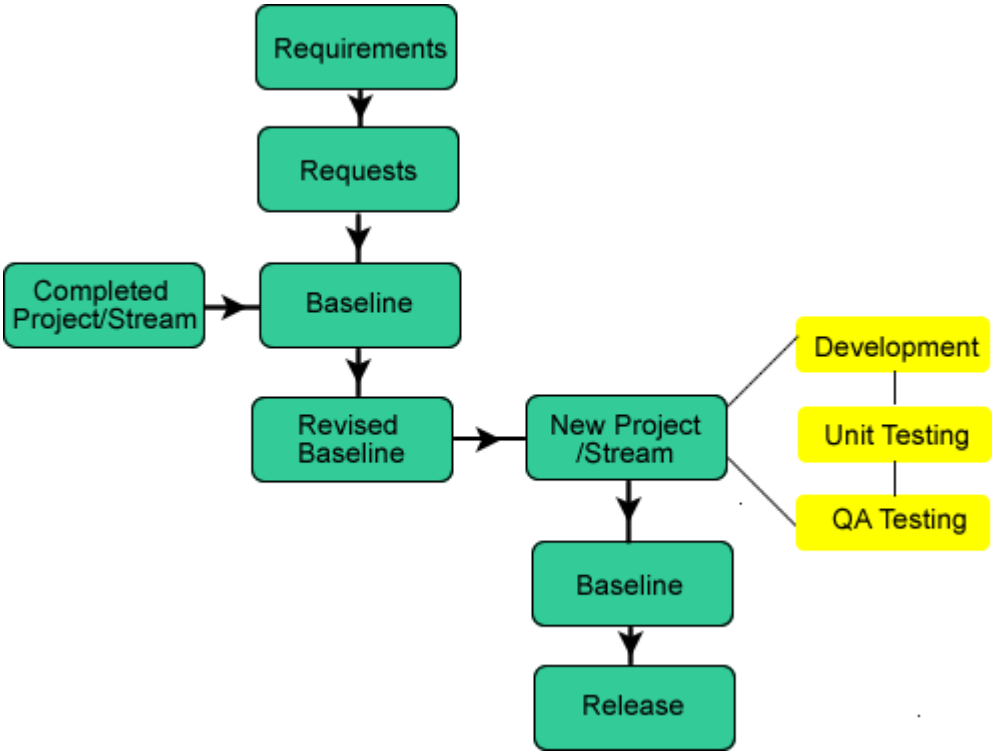
The package request might, for example, move from the "Planning" state to "Under Work". Developers fix issues, relating the changed files to the requests. The items to which the requests are related are therefore included in the package request. If say, during development, the Development manager realizes one enhancement will not make the build date, it can be unrelated from the request.

The changes can then be built and when all work is complete the package request can be moved to state "QA".

Baselines

In Dimensions you can create a *baseline.* A baseline is a frozen configuration of the versions of items that went into a particular deployment of a project/stream or strand of development. You can create a baseline in conjunction with a template that defines which types of item and which versions are included. A baseline can be used to create a new project or stream from which another development can begin.

You can create a *revised baseline* which consists of an existing baseline to which a delivered set of requests has been applied, so that versions of the items created in response to those requests are captured in a new baseline.



Work Areas

As a developer, you can associate a project or stream with an area on your local disc, or an area located on another part of the network, so that when you check out the items under development, they are copied to that area and are under the required folder/directory structure. This area is referred to as a *work area*. A user can associate their own separate area with a project/stream or associate an area which is shared by a team.

Such areas can also be defined under Dimensions CM control in the Administration Console by associating them with a Dimensions CM defined network node and folder location.

Deployment Areas

Dimensions has the facility to allow a project manager to define *deployment areas* for projects/streams whose items have reached a particular stage of development. The term *deployment* in Dimensions CM refers to the process of copying item files to a controlled location when they have reached a particular stage of development. You can define stages in your software development lifecycle, such as "Development", "QA", "LIVE", and you can have item files automatically copied to the associated areas when they have reached the corresponding state of approval for that particular stage.

This process of approval is called *promotion* and is linked to the Global Stage Lifecycle (GSL). The GSL is the lifecycle that items follow through the deployment process, and deployment areas are associated with these stages. There is one single GSL defined for the base database, but you can change this or configure your own GSL. Stages in this lifecycle can, optionally, also be associated with the lifecycle states defined for different types of object, described in "Lifecycles and Attributes" on page 9 above, so that the two processes can be linked.

Developers do not manually check items in or out of these areas but they can be viewed and accessed from within the GUI interfaces.

These *Deployment* areas can have filters applied to determine the types of files they will contain, so that, for example, you could have different areas for versions of the software for different operating systems.

When items are promoted to a stage, you can choose to have them automatically deployed to one or more areas associated with that stage by defining them as default areas, or you can have the option of deploying them manually later. The deployments can also be scheduled to take place at a specified time in the future.

You can promote:

- **Files**—versions of individual files can be selectively promoted to the required stage and copied to the corresponding promotion area.
- **Requests**—A request can be promoted so that the versions of the item files created in response to that request are also promoted, and optionally deployed to the promotion area when the request is promoted to that stage. You can have the child requests also automatically promoted when the parent request is promoted. This means that a request can be used to deploy a controlled "package" of changes to the designated areas when they are ready. Refactoring

changes, such as the renaming or moving of files, can also be deployed using this method.

- **Baselines**—A frozen configuration of items can be promoted from one stage to another so that its item files are promoted and copied to the deployment area when the baseline is promoted.

You can configure scripts to perform chosen tasks whenever files are deployed to the area. There can be separate scripts for before and after an area is populated, and for when an error condition occurs. For example, you could run a script to tidy up the area before the files are copied to it, and you could perform a build of the software after the deployment has taken place. Dimensions maintains full audit trails for these operations.

Deployment jobs can be viewed and managed using the Deployment tab in the web client. Deployment of files can be re-scheduled to occur at specified times in the future, and can be rolled back or canceled.

For Comprehensive details about how to use Dimensions CM deployment, see the Deployment Guide.

Control and Manage Your Builds

Configuring Builds

Dimensions incorporates a comprehensive Build Management tool that:

- Allows version control of build configuration information (allows repeatable builds).
- Enables building of selected deployment areas, work areas, and baselines.
- Supports execution and scheduling of remote builds.
- Can gather bills of materials and preserve targets.

You can have builds performed automatically when the items involved have reached a particular stage in their lifecycle. You can also select different targets to be built for versions of your project intended for different platforms. Dimensions has its own build engine to perform the builds or you can use other build engines such as OpenMake, Ant, GNU

Make etc. allowing easy integration with existing build scripts and engines.

Dimensions CM also provides integrations for CruiseControl and Ant which are more suitable for agile customers using streams.

that you can use to access Dimensions CM functionality from within an Ant build script.

Controlling Releases

After deployment and testing, copies of a particular configuration of the product can be captured as a baseline and made available as a release. For example, when a baseline passes product and system testing, it is considered ready and can be released to customers using the Dimensions release management facilities. These facilities allow a full or delta configuration of a product to be fetched from the Dimensions repository to your selected release location.

You can use release templates that can be applied to a baseline in order to filter the configuration of items that are pertinent for a particular release, and optionally change the directory/folder path mapping.

Deploy your Changes

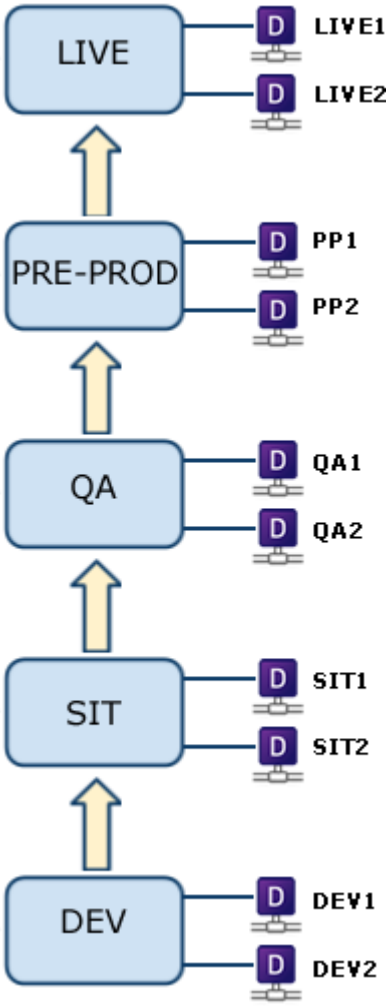
Dimensions CM has two deployment models:

- Dimensions Deployment
- Deployment Automation (DA)

Dimensions Deployment

Dimensions Deployment, the default deployment model, manages individual items, or groups of items, as they move through the development process. The change management process is defined by a series of development stages linked to the Global Stage Lifecycle (GSL). For each stage of the GSL one or more physical or logical locations on disk, known as deployment areas, can be setup to contain a snapshot of the items at that stage.

This is a sample GSL with five stages from development (DEV) to system integration testing (SIT), quality assurance (QA), pre-production (PRE-PROD) and production (LIVE), with two deployment areas assigned to each stage:



The term *deployment* refers to the process of copying item files to a deployment area when they have reached a particular stage of development. The approval process is called *promotion* and is also linked to the GSL. Deployment areas can have filters applied to determine the types of files they will contain, so that, for example, you could have

different areas for versions of the software for different operating systems.

When items are promoted to a stage, you can choose to have them automatically deployed to one or more areas associated with that stage by defining them as default areas, or you can have the option of deploying them manually later. The deployments can also be scheduled to take place at a specified time in the future.

You can promote:

- **Files**—versions of individual files can be selectively promoted to the required stage and copied to the corresponding promotion area.
- **Requests**—A request can be promoted so that the versions of the item files created in response to that request are also promoted, and optionally deployed to the promotion area when the request is promoted to that stage. You can have the child requests also automatically promoted when the parent request is promoted. This means that a request can be used to deploy a controlled "package" of changes to the designated areas when they are ready. Refactoring changes, such as the renaming or moving of files, can also be deployed using this method.
- **Baselines**—A frozen configuration of items can be promoted from one stage to another so that its item files are promoted and copied to the deployment area when the baseline is promoted.

You can configure scripts to perform chosen tasks whenever files are deployed to the area. There can be separate scripts for before and after an area is populated, and for when an error condition occurs. For example, you could run a script to tidy up the area before the files are copied to it, and you could perform a build of the software after the deployment has taken place. Dimensions maintains full audit trails for these operations.

Deployment jobs can be viewed and managed using the Deployment tab in the web client. Deployment of files can be re-scheduled to occur at specified times in the future, and can be rolled back or canceled.

For full details see the *Deployment Guide*.

Deployment Automation

Deployment Automation (SDA) automates software deployment, the process of moving software through pre-production stages to final production. Typically, each stage represents a step of higher criticality, such as testing to production.

Software deployment complexity increases with more releases to deploy, more deployment targets, more types of deployment targets, shortened deployment cycles, and changes in technology. Deployment Automation helps you meet the deployment challenge by providing tools that improve deployment speeds while simultaneously improving their reliability.

You can invoke an SDA application process during promotion instead of using Dimensions Deployment, the default CM deployment model. The benefits of using SDA include:

- Easy configuration and management of deployments.
- Processes can execute on multiple machines, for example:
  - Stop server A.
  - Move files to server B.
  - Backup DB and restart SQL Server C.
  - Start server A.
- Multiple CM servers and base databases can use one SDA server.
- Re-use of existing SDA processes.
- Automatic promotion through an SDA pipeline.
- Different component versions can be automated together.
- Single sign on between CM and SDA.

For details about using SDA with CM see the *Deployment Guide*.

For details about configuring SDA see the Documentation Centre.

Chapter 2

The Dimensions CM Process Model

In this Chapter

About the Process Model

This chapter provides a conceptual overview of the Dimensions process model and the basic components and features involved. For a more detailed description, see the Process Configuration Guide.

The Dimensions process model enables you to control the way you develop software or manage assets. The functional areas that Dimensions enables you to manage are shown below.

The Components of the Process Model

About Base Databases

The basic instance of a Dimensions process model is a *base database*. A single Dimensions CM server can connect to multiple RDBMS instances and these database instances can also be hosted on machines distinct from the Dimensions CM server, if needed. Within a single database instance, you can choose to divide up your applications being managed into what Dimensions CM calls base databases. In implementation terms, a base database is a schema, for example, in Oracle each base database is an Oracle user with their own copy of the Dimensions CM schema.

You may decide to create different base databases to separate out different applications due to security concerns or simply because the applications are following very different processes.

About Products

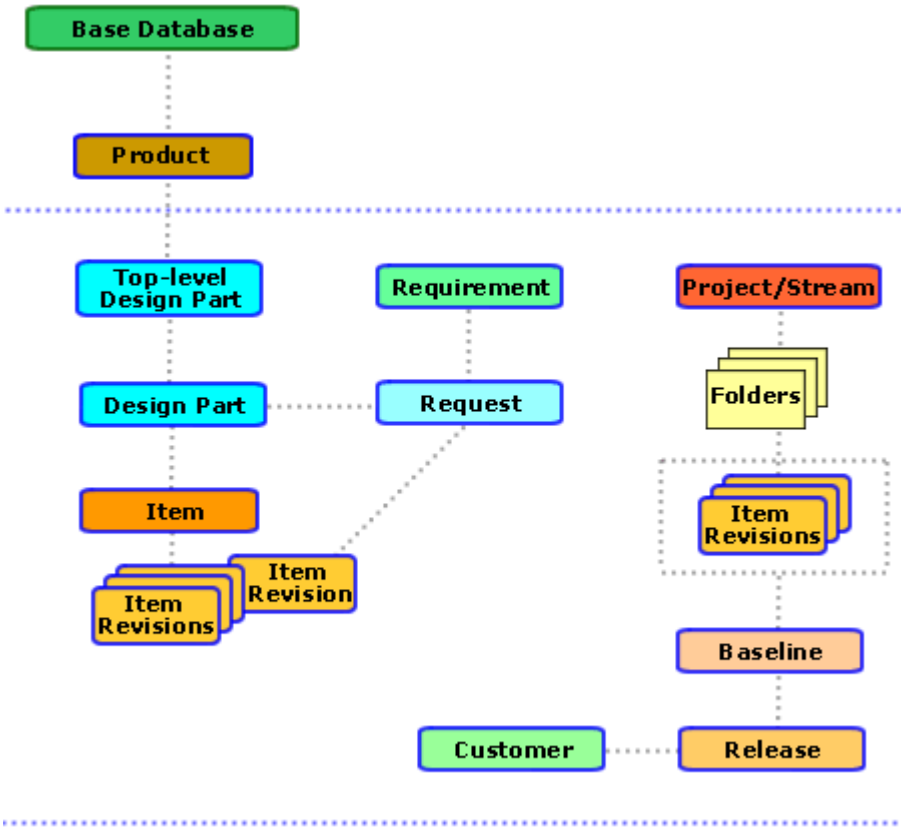
Within that base database there can be one or more products. A product is a high-level container for related development projects, for example, in your in-house environment you could create a single application called "ACME LEGER". Some aspects of the development process/workflow can then be customized on a per-product basis (for example, Dimensions CM object types, rules, attributes, and functional component breakdown).

Within a single product, many development efforts may be in progress (sometimes in parallel) and Dimensions CM calls these containers *projects* or *streams*. So, perhaps in-house, you might have a Dimensions CM project or stream called "ACME 10" and another one called "ACME 11". With respect to these projects/streams, you can then perform Dimensions CM operations such as:

- Updating your work area with changes and delivering changes back to the repository.
- Raising defects.
- Implementing requirements.
- Building and deploying software.

Dimensions CM Objects

Each product has a set of Dimensions object classes that can be configured with their own properties, such as lifecycles, attributes, or relationships with other objects. The diagram below shows these objects and the part they play in the process model.



Product: This is a major subdivision of the process model, the objects that you configure belong to the product in which they are defined.

Design part: A design part is a logical part of the functional breakdown of the product and design parts are related to one another in a hierarchy. When a product is first created it automatically has a top-level design part of the same name as the product.

Item: An item represents an asset, for example a source file, that forms a part of the implementation of your product. It belongs to, or is owned

by, a particular design part, although it could be reused by other design parts. The item itself is a logical representation of the asset.

Item revision: This is a specific version of an item. Item revisions are identified by having a version appended to the name of the item. This version consists of a *version number*, optionally preceded by a *branch*. An item revision usually has an item file associated with it. The physical file is stored in an *item library* and can be located on a different machine to the one that hosts the base database.

Project/Stream: A project or stream represents a collection of item revisions that form a strand of development of part of the product. The item revisions are related in a folder structure which reflects the organized structure in which the files belong.

A user works with a project using commands that operate on specific files, such as check out, check in, and so on, whereas when working with streams the user copies sets of files to and from the stream in the repository using commands such as update and deliver. A project can contain multiple revisions of each item and can have one or more branches associated with it. A stream contains only one branch of revisions for each item.

Baseline: A baseline consists of a snapshot of a collection of item revisions that is used to capture a particular milestone in the development of the project or stream. A tip baseline captures the latest item revisions in a project/stream at the point that it is taken. A release baseline is created by applying a *baseline template* to a project/stream or design part. The baseline template is a set of rules that you define, and you can create different baseline templates for this purpose. The template can contain rules based on the type, the lifecycle state, and other properties that determine which item revisions are included in the baseline.

Release: A release is created by applying a *release template* to a baseline in order to copy the item files for some or all of the items in the baseline to a particular location. You define release templates in a similar way to baseline templates.

Customer: A customer is a logical record of a company or organization to whom a release has been supplied, or *forwarded to*.

Request: A request represents an enhancement or fault that has been recorded against the product. It affects one or more design parts, and can affect one or more item revisions.

Requirement: A requirement is created in Dimensions RM and is associated with a Dimensions project. It is visible in Dimensions CM and you can relate requests to requirements.

Object Relationships

Dimensions CM objects can have various types of relationships that help you group, search and control these objects. For example a request can have other child requests related to it. It can affect a design part, which is the functional area in which the required change needs to be made. It can also be related to the item revisions which need to be changed (are *affected by* it) and to the new item revisions created (*in response to* the request) in order to implement the required change.

Some relationships are formed automatically as a result of operations you perform. For example, when you create a baseline, you have to specify a design part to which the baseline has an *owned by* relationship. Other relationships may be optional or user-defined.

Defining Object Types

Some classes of object have a user-defined type associated with them. Within the same product, you can create different types of these objects. This enables you to define certain properties for those objects which can be configured differently for different purposes. These object classes are:

- Design Parts
- Items
- Requests
- Projects/streams
- Baselines

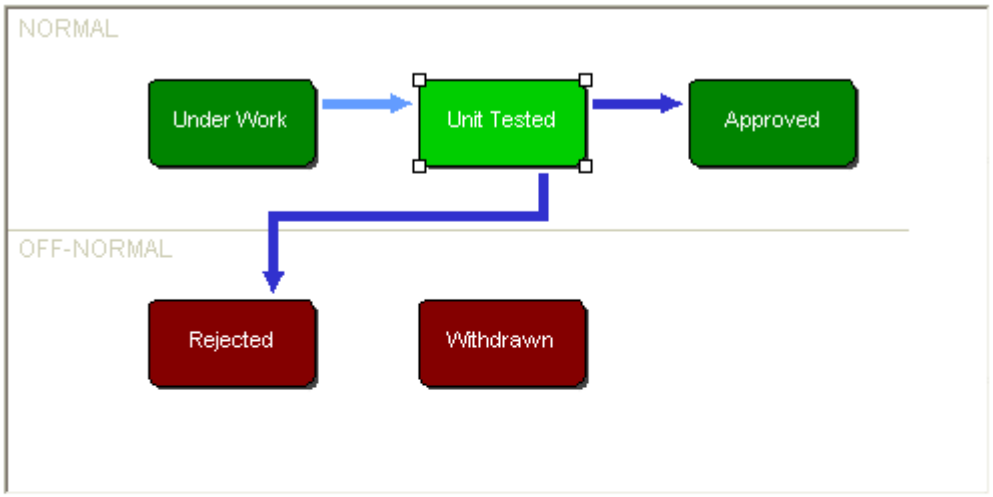
This means that you can, for example, define different types of request. You could have a request type of CR (Change Request) and another of type TDR (Test Defect Report). These types of request could follow different lifecycles for their workflow and process of approval, and have different attribute fields for the user to fill in during this process.

For further information about the Dimensions CM object types and how to define them, see the *Process Configuration Guide* Object Type Definitions chapter.

Lifecycles

A lifecycle consists of a series of possible states that an object can be in. The object can be moved to a given state from another state, which in Dimensions CM is called *actioning*. Normally an object in a particular state is only allowed to be actioned to certain other states. The allowable pairs of to and from states between which it can be actioned are called *transitions*, and only certain users can action the object through those transitions. (Some users, for administrative purposes, may be granted the privilege to action a particular object between any states, but these are the usual transitions for most users.) There are *normal* states, which follow a progressive path through the lifecycle, and *off-normal* states, which are not located on this path.

For example, the lifecycle for a source file below shows the Unit Tested state selected. Normally that state can only be reached from the Under Work state when it is actioned by a developer. The item can only be actioned from this state to Approved by a Team Leader. The only other transition possible from Unit Tested is to Rejected, which can only be made by a Team Leader.



Transitions

From	To	Roles (O/P)
UNDER WORK	UNIT TESTED	DEVELOPER P
UNIT TESTED	APPROVED	TEAM LEADER P
UNIT TESTED	REJECTED	TEAM LEADER P

Attributes

Attributes are fields used to contain information about an object. Attributes can be:

System defined: These are predefined by Dimensions CM for the object class. They are often used by Dimensions CM to maintain certain data about the object, for example Creation Date, or Current Status.

User defined: These are defined in your process model for a particular object type. For example an item of type *Source* may have an attribute called *Lines of Code*, whereas an item of type *Object* may have an attribute called *Platform*.

Attributes can have different formats, such as Text, Date, etc. They can also be *multi-value*, meaning that a field can have more than one value,

or *multi-field, multi-value* meaning that it can consist of a group of fields with more than one value.

Attribute Rules

Rules can be defined for a particular lifecycle transition with regard to an attribute. A rule can specify that for a given transition for the specified user:

- The attribute is visible to the user.
- The user can update the attribute.
- The attribute is mandatory, that is the user must update it before they can action the object.

Valid Sets

You can define sets of rules that allow users to only enter certain values for an attribute. Also, you can define a validation that cross-validates the values entered in a combination of fields, and can also automatically fill in the values of dependent fields when the user enters a value for an attribute. You define valid sets independently and than assign them to attributes as required.

Sensitive Attributes and States

When you define an attribute or a lifecycle state, you can, for security purposes, specify it as *Sensitive*. This means that:

- If an attribute is sensitive, a user is required to re-enter their password before they are allowed to update it
- If a state is sensitive, a user is required to re-enter their password before they can action the object to or from that state.

CM Rules

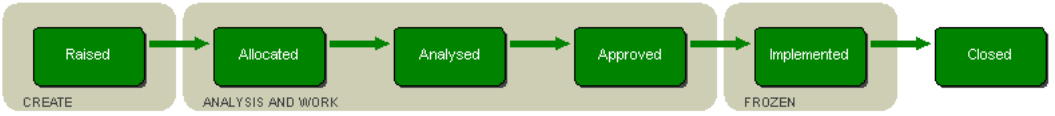
You can set up a special category of rules relating to items and/or requests called CM (Change Management) Rules.

For requests you can configure certain *phases* that determine what operations can be performed as in the example below.

Permitted operations:

Request Phase	Relate Design Parts	Relate Items As Affected	Relate Items In Response To	Relate Requests.	Update Attributes
Off Normal	⛔	⛔	⛔	⛔	✅
Held	✅	⛔	⛔	⛔	✅
Create	✅	✅	⛔	⛔	✅
Analysis	✅	✅	⛔	✅	✅
Analysis & Work	✅	✅	✅	✅	✅
Work	⛔	⛔	✅	⛔	✅
Frozen	⛔	⛔	⛔	⛔	✅

You can assign the lifecycle states for a request type to those phases, as in the example below.



These phases then determine what operations can be performed when the request is at a particular state.

You can also configure certain rules for item types. For example, you can require that a request must be related to an item revision before it can be checked out.

Upload Rules

Upload rules are used to map file name patterns to Dimensions file formats and item types. These rules determine whether files that match a certain file name pattern can be added to the database using a Dimensions client or an IDE, and if so, what item types they will be created as. Upload rules must exist in the base database before you can start adding files.

Roles & Privileges

Roles and Privileges consists of the following topics:

Users and Groups

Users are registered in Dimensions CM with a username and password. Various privileges and facilities can then be assigned to these individual users. You can also set up *groups* of users, and control access and privileges by assigning them to a group. This can make things easier to maintain, since when users join or leave a particular department they can simply be added to or removed from the appropriate group.

Differences Between Roles and Privileges

Dimensions CM provides two mechanisms for determining which functions different users can perform. One is the use of *roles*. This consists of defining a role, or using one of the Dimensions CM predefined roles, and assigning that role to one or more users. There is also another mechanism, called *privileges*. A privilege is either a particular operation that can be performed on a class of object, or an administrative function. A privilege can then be made available to users or groups according to certain *privilege rules*. The basic differences between privileges and roles can be summarized as follows:

- Roles can be assigned at product or design part level, thus you can assign a role to different users for different functional areas of your product. The ability to action objects through specific lifecycle transitions can only be assigned by means of roles, and are not defined using privileges. However, you can assign the required roles to users or groups.
- Privileges define specific operations that a user can perform. They can be assigned to specific users or groups, or can be assigned to all users, or they can apply to a user according to certain rules. They can also be assigned to a role. They cannot be restricted to particular design parts, but generally only apply to all objects in a particular class.

For details about privileges and roles and how to use them, see the chapter Privileges and Roles in the Process Configuration Guide.

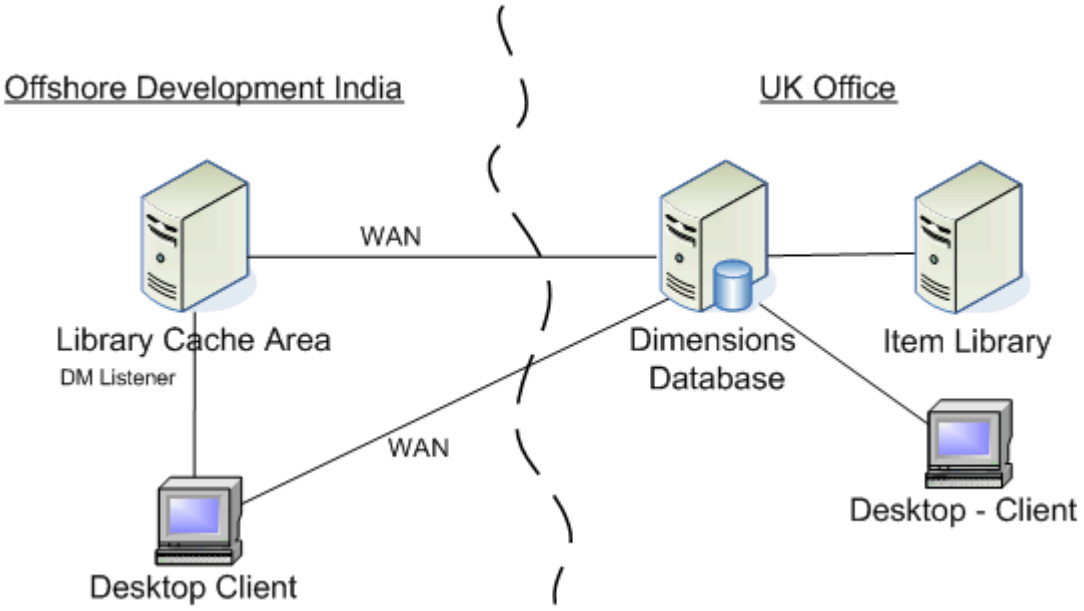
Area Management

Dimensions CM enables you to define areas. An area is a location somewhere on a network node that is used to contain item files. There are three types of area:

Work Area: an area used for development work for a particular user or a number of users, so that their file operations such as check in, check out, update, deliver, and so on, take place in the context of that area. The same work area can be assigned to more than one user.

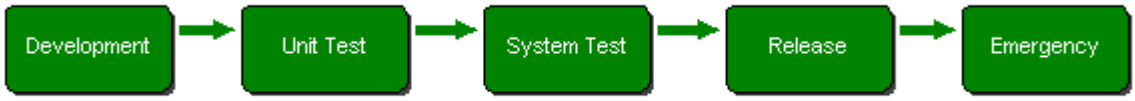
Deployment Area: an area associated with a project/stream for a particular stage in its development. It is used to contain the item files that have reached a particular stage and are ready to be system tested or to be included in a release build.

Library Cache: An area that is defined in order to contain copies of item files for a project/stream to improve the efficiency of Dimensions item operations when accessing a database over a remote network. When a user needs to access an item file, Dimensions looks in the library cache area first to see whether the file is already there before attempting to retrieve it from the remote server.



When you define an area, you identify the network node and the folder path where the area is located. You then associate the area with one or more projects or streams. Users in the Dimensions CM clients can then users can select that work area when they are working in a project/ stream as the location for the files when they are working in that project/ stream.

If the area is a deployment area, you also associate it with a stage in the *Global Stage Lifecycle*. By default, the Global Stage Lifecycle is set up with the following stages:



In Dimensions CM, the process of approving item revisions for a particular stage for their files to be copied to these areas is called *deployment*. For deployment areas, you can also define scripts to be performed in connection with deployments. There can be separate scripts for before and after an area is populated, and for when an error condition occurs. You associate these scripts with the area when you define it.

You can deploy baselines and requests, as well as individual item revisions. This enables you, for example, to deploy all the items related to a request, or a group of child requests.

Dimensions Build

Dimensions Build is a build management, execution, and monitoring tool that is part of Dimensions CM. Dimensions Build can be used by individual developers who are building their code at regular intervals, or by build managers who are executing controlled builds of source code from many developers. Dimensions Build enables you to execute builds from the Build Administration cluster in the Dimensions CM Administration Console, or from the Dimensions desktop, web, and ISPF clients. Dimensions Build supports cross-platform builds on secure remote nodes where the same source code is used for builds on multiple, heterogeneous platforms.

Dimensions Build runs on all major Dimensions CM supported platforms. It is build engine independent and integrates with many third party engines on distributed and mainframe platforms.

Benefits:

- Completely configurable sets of build stages (previously these were hard coded).
- Open architecture allowing for use of external build engines in addition to Catalyst Openmake. If the user already has a build system based on a standard engine such as Make or ANT, those systems can be re-used without major rework.
- Integrated scheduler to control when builds occur.
- Build environment integrated with the Dimensions CM engine to version manage the build environment.
- Integration with the Dimensions CM role management to control build administration and execution.
- Audit trail stored within Dimensions CM.
- A build engine for z/OS offering high performance and parallel build execution.

Dimensions CM also provides integrations for CruiseControl and Ant that may be more suitable for more users of agile development processes.

For details see the Dimensions CM Build Tools User's Guide.

Chapter 3

The Components of Dimensions CM

In this Chapter

Dimensions Architecture

Dimensions CM is a true enterprise-level product. It has a multi-tier architecture with heterogeneous platform support including Windows, UNIX, Linux, and IBM z/OS platforms.

- It has a web client, desktop client, and a Web browser-based administration console.
- It allows a local workspace to be used on different platforms.
- It allows assets to be held in any location.

Dimensions CM allows the use of distributed server and client platforms. The server employs an RDBMS database that contains the details of the process model and the metadata relating to the assets under its control. The data relating to these assets are held in *item libraries* that can be held in a location independent of this server. Client platforms can access the database across a network using the *web client*, *desktop client*, or a *command-line client*. You can also access the Dimensions CM database from within a number of IDEs for which integrations are provided. The files under development can be held locally on the client platform or can also be located on a remote, or tertiary node accessed via the network. The process model itself can be maintained via the Web browser-based *Administration Console*.

Dimensions CM Clients

The following client tools are available for accessing a Dimensions CM database.

Administration Console: This is a Web browser-based GUI client giving you access to the process modeling and other administrative functions of Dimensions CM. For more details, see the *Process Configuration Guide*

Web client: This is a Web browser-based GUI client giving you access to the Version and Change Management functions of Dimensions CM. For more details, see the *Dimensions CM User's Guide*.

Desktop client: This is a Windows-based GUI client giving you access to the Version and Change Management functions of Dimensions CM. For more details, see the *Dimensions CM User's Guide*.

Command-line clients: These allow you to access the Version and Change Management functions of Dimensions CM and the process modeling functions via the command line. For more details, see the *Command-Line Reference Guide*.

Developer Command Line: This is a simplified command-line client that enables you to work with streams in your development area. This tool provides a set of commands for tasks such as updating your work area from a stream, delivering your changes to a stream, or viewing and resolving conflicts in content.

For more details, see the *Command-Line Reference Guide*.

ISPF Client: This is an interactive client that can be used on z/OS platforms to access many of the Version and Change Management functions of Dimensions CM. For more details, see the *Dimensions for z/OS User's and Administrator's Guide*.

Other Client Tools and Plug-ins

There are a number of other tools for accessing the features of Dimensions CM. These are:

Synchronize Wizard: This enables you to compare files and folders in your working location with the corresponding folders and items in a Dimensions CM project or stream. You can update your local work area from the Dimensions CM repository, deliver your changes to the Dimensions CM repository, or in the case of projects, synchronize your work area and repository with one another. Dimensions CM creates and maintains metadata relating to these files and the corresponding items in the repository. This enables you to see the type of change that has occurred between the work area and the repository, and enables the tool to process these changes, usually without user intervention.

Dimensions CM Plug-in for Microsoft® Windows Explorer: This enables you to work with item files in Windows Explorer. You can see various fields and icons relating to the files and their corresponding items in the Dimensions projects or streams. You can perform certain actions such as deliver or update from the Explorer window without having to start a desktop client or web client session, and you can run the Synchronize Wizard to synchronize the work area with Dimensions CM.

Project Merge Tool: This enables you to compare folders containing files and subfolders in your Windows work area with project folders and

item revisions in Dimensions CM. The GUI interface indicates the type of change that has occurred in relationship to each file and informs you if there is a conflict that needs to be resolved. You can use this tool to compare and merge folders and files under development with the items in the Dimensions CM repository.

Araxis Merge: This is a file merge tool that enables you to compare the content of item files with one another line by line and merge the changes together.

For more details of these tools see, the Dimensions CM User's Guide.

IDE Integrations

Dimensions CM provides various integrations for integrated development environments (IDEs). These enable developers to use the version control or change control features of Dimensions CM whilst working within the IDE.

For more details of these, see the IDE User's Guide.

Administration Console

The Dimensions CM Administration Console enables you to define the rules governing the process model and to configure the dimensions objects involved. The basic areas of administration are:

Configuration Object Management: This is concerned with defining the behavior and characteristics of the objects involved in Dimensions CM. It includes defining different types of these objects and the attributes and lifecycles they will follow to suit your process model.

Product Administration: This involves defining the products in your base database and their design part structure, or functional breakdown. There are also other configuration options that are defined at product level, such that valid sets governing object attributes, and baseline and release templates.

Users and Roles: This allows you to register Dimensions CM users and groups of users and to define roles. You can then assign the privileges that are required to perform the various functions in Dimensions and

control which functions people can use. Other features that are defined here are email notifications, when and to whom emails are automatically sent when certain events occur, and User Interface Profiles, which determine which functions a user will have access to in the Dimensions CM clients.

Distributed Development: This section is concerned with managing how Dimensions CM operates in a distributed network environment. It includes the definition and assignment of work, deployment and library cache areas, network administration, and the Build Administration for the Dimensions CM common build component. Also you can manage Replication and Archives and Transfers, which are concerned with archiving parts of a base database and transferring Dimensions CM data from one database to another.

Database Management: This section enables you to view Connection Pooling statistics for a Dimensions CM databases, and to define reports that can be made available to users. You can also configure certain options for your database, such as whether to use projects or streams, whether or not to use requests, and optionally to use SBM as the request provider.

For details of Configuration Object Management, Product Administration, Users and Roles, and Area Definitions, see the Process Configuration Guide.

For Network Administration, Replication, Archives and Transfers, and Connection Pooling Statistics, see the System Administration Guide.

For details on Build Administration, see the Dimensions CM Build Tools User's Guide.

For details on Reports Administration, see the Reports Guide.

Dimensions Build

Overview

Dimensions Build is a build management tool. It is configured and administrated using the Build administration function of the Administration Console. For details see the *Dimensions CM Build Tools User's Guide*.

Versioning and Repeating Builds

You can create multiple versions of build configurations and repeat these builds whenever you want. Each version of a build configuration includes the following information:

- Target definitions including high-level dependencies.
- Scripts used to build each target. For z/OS you can version the scripts required for each separate step when building a target.
- Build area definitions (host, authentication details, file system location etc).

Scheduling Builds

You can schedule the execution of builds to suit your build paradigm. When you set up a scheduled build job you specify the build configuration and version that will be executed, the targets, the build area, and the start time. You can also specify the frequency at which a build reoccurs.

Monitoring Builds

You can monitor the status of builds that are currently running and view the history of completed builds. For each build event you can view the expanded script used to build the step, the output log of link and compile listings for the target, and the error log (if applicable).

Notifications

You can create and subscribe to e-mail notifications that update you about the progress of your build jobs.

Integration with Dimensions CM

Dimensions CM performs the following functions for Dimensions Build:

- Drives the population of build areas.
- Authenticates and authorizes items.
- Preserves outputs and intermediate files generated by the build engines.

- Preserves bill-of-materials and post-build dependency information to enable you to perform impact analysis and traceability.
- Records which build configuration versions were in use at the time a baseline is taken.

You can also execute builds directly from the following Dimensions clients:

- Desktop client
- Web client
- ISPF client

For more information see the User' s Guide and the Dimensions for z/OS User' s and System Administration Guide.

Difference Between Mainframe and Other Platforms

While Dimensions Build offers many cross-platform advantages, it is important to understand that there is a difference in the most commonly-expected scenarios for the mainframe versus the other platforms.

On the distributed side, as in Windows, it is expected that the user (perhaps a build manager) will have a favorite build engine such as Ant or Dimensions Build. Dimensions Build integrates with those tools and even import the target definition files such as build.xml that define the targets and dependencies. In other words, it is assumed that the distributed platform user will make use of third-party build engines.

On the mainframe side, the reverse is true. Dimensions Build incorporates a build engine, as stated previously, and it is expected that the user will make use of it.

Versioning and Repeating Builds

You can create multiple versions of build configurations and repeat these builds whenever you want. Each version of a build configuration includes the following information:

- Target definitions including high-level dependencies.
- Scripts used to build each target. For z/OS you can version the scripts required for each separate step when building a target.

- Build area definitions (host, authentication details, file system location etc).

Developer Tools

C/C++ API DTK

The Dimensions CM Developer's Toolkit (DTK) enables you to access and manipulate objects that are held within a Dimensions repository.

Java API

The Java API provides full, programmatic access to the features of Dimensions CM. The Java API allows you to create and manipulate versioning, change management, and process modeling data while under the control the Dimensions CM permissions and change management rules framework.

Scripts and Templates for Remote Job Execution

Remote job execution enables you to invoke processing on a remote node on which a Dimensions Listener is running, and to integrate Dimensions with your own systems and tools. Dimensions supports a templating language on Windows, UNIX, and z/OS (both USS and MVS) that you can use to customize the remote construction of processing.

Web Services API and ALF Events

With the Dimensions CM Web services API (Application Programming Interface) and the Dimensions CM ALF (Application Lifecycle Framework) events, you can access key Dimensions CM features from your own applications. This enables you, for example, to build your own front-end clients or create business mashups and orchestrations with Solutions Business Mashups (SBM).

Replicator

Replicator is designed specifically to support software development effectively in decentralized and distributed team environments. Replicator supports remote and distributed development of items, baselines, and requests by allowing teams of developers, located at different geographical locations, to work in parallel on the same project files while accessing their own local Dimensions repository. Behind the scenes, Replicator will ensure that items, baselines, and requests, plus their associated meta-data, are replicated from one repository to another in accordance with the replication policies set up by the project administrator. Furthermore, it will ensure that these separate repositories are always kept in synchronization with regard to any changes in these policies.

Replicator can be configured to operate across a network between a master base database and a number of subordinate base databases. Users can then work on the items, baselines, and requests and a replication can be run to synchronize the updated objects between the databases. Replication can also be done without using a network by downloading the data at one site, physically transferring this data to another site and uploading it to a base database at that site (air-gap replication).

For further details about using Replicator, see the System Administration Guide.

Dimensions CM ART

The Dimensions CM ART facilities enable you to archive the contents of a baseline and store a copy of them on designated archive media. These media are referred to as archive volumes and each has an identity known to Dimensions ART as a Volume-id. An archive, which is effectively a copy, may be required for various reasons including satisfying procedural or contractual obligations. A special Release-baseline may be taken using the *ALL rule which includes in the baseline all revisions of each relevant item of the type specified regardless of lifecycle status.

Dimensions ART also provides functions to retrieve such information back into the system and restoring their status to ONLINE. The retrieval is based on material that has been taken OFFLINE and held on an archive

volume such as a magnetic tape. Each tape is identified by a Tape No. and a "Volume-id" which is written physically on to the tape by Dimensions ART. The system permits retrieval of such items from a specified Volume-id, on a selective basis, the Tape No. only being used for mounting purposes.

For further details of Dimensions ART see the System Administration Guide.

Reports and Published Views

Dimensions CM provides a sophisticated Report Builder feature that is accessed through either the web client or desktop client. This allows you to create and modify end-user GUI client reports for all Dimensions CM object types. The formats include:

- Listing report
- Distribution table
- Pie chart
- Bar chart

You can construct sophisticated queries involving Dimensions CM objects, such as design part, baseline request, item, workset, release and product, and create and save reports based on them.

See the User's Guide for details of Dimensions CM Report Builder.

Dimensions also provides a number of Published Views. These enable you to write your own programs to extract information from the database. You can also setup your own reports, which are run using the Dimensions RUR command, by using User Reports Administration in the Administration Console.

See the Reports Guide for details of these features and a list of the Published Views available.

For details of the RUR command, see the Command-Line Reference Guide.

In addition to this, the desktop client allows you to access the Crystal Reports desktop, provided you have this product installed. See the online help in the desktop client for information on how to access this.

The Documentation Set

The Documentation Set consists of the following topics:

Books

The Dimensions Documentation Set is provided in the form of Adobe ™ Acrobat ™ Portable Document Format Files (.pdf) files.

Adobe Acrobat Readers are provided for Microsoft Windows platforms and a variety of UNIX and Linux platforms.

For information on using the online manuals, access the Acrobat Reader Online Guide.

Dimensions provides a Documentation Roadmap document that contains a list of all the PDF books together with links. You can also search the entire Dimensions documentation set from the Roadmap PDF file using the Adobe Acrobat Search feature:

- Search for a specific word or a phrase.
- You can choose to search within a single book or across an entire doc set.
- Jump to a specific section in a manual from the bookmarks or Table of Contents.

To search using Acrobat Reader:

1. In Adobe Reader, select Edit | Search.
2. In the text box, enter the word or phrase for which you want to search.
3. Select the **All PDF Documents in** option, and browse to select the folder in which you want to search. (If you have a document open that has an Adobe Catalog index attached, you can leave the In the

index named... option selected to search across all the manuals in the index.)

4 Optionally, select one or more of the additional search options, such as Whole words only and Case-Sensitive.

5 Click the Search button

Online Help

Dimensions provides online help for all its GUI clients.

The Dimensions client GUIs provide online help in HTML WebHelp format and it is viewed using the appropriate HTML browser. Context-sensitive information is provided to aid the user in the application of the specific function.

Dimensions CM for z/OS

Dimensions for z/OS provides textual online help for ISPF Panels. Context-sensitive help is provided to aid the user in the application of the specific function.

HTML online help is also provided for the SDK for Dimensions for z/OS. For more information on how to use the SDK, see the Dimensions CM *for z/OS User's and Administrator's Guide*.

Glossary

<all_local_ branches>

A replication option that specifies that all *named branches* owned by the master DB site are to be replicated to the subordinate DB sites.

<all_named_ branches>

A replication option that specifies that all *named branches* on the master DB site are to be replicated to the subordinate DB sites, except those that are owned by the subordinate DB sites.

Action

To move a Dimensions *object,* such as an *item*, *request*, or *lifecycle*, to another lifecycle *state*. Only a *user* with the appropriate *role* for a state can action the item to another state.

Administration Console

A Dimensions tool for setting up the *process model* and creating *products*. For more information, see the *Process Configuration Guide*.

Ancestor

When comparing projects, the base *project* to which other projects, called *derivatives*, are compared in the *Project Merge Tool*.

When comparing files, the base file to which other *derivative* files are compared in the default merge tool.

Archive baseline

A restricted variant of a *release baseline*. Archive baselines use a *baseline template* that includes all of the *revisions* of the *item types*.

Use the archive baseline to preserve the *product* at a milestone using the Dimensions Archive, Transfer, and Retrieval (ART) facilities. For more information on Dimensions ART, see the *Distributed Development Guide*.

Area

A Dimensions-defined location on a network node where item files are stored. Files are referenced using the same hierarchical folder structure as the items in the *project* or *stream*. See also *work area*, *deployment area*, and *library cache area*.

Attribute A property of an *object* that records important configuration information such as creation date, owner, status, description, and the user who last updated the object. See also *user-defined attributes*.

Authentication A requirement for users to confirm their password before they can perform certain actions on an *object*. The users are presented with a confirmation dialog box if they attempt to update an object's *lifecycle state* or *attribute* that is specified as *sensitive*.

Base database The basic instance of a Dimensions *process model*. Each base database has a separate database schema, and can contain one or more *products*. Products in different base databases cannot reference each other.

Base database site The location of a base database, which is specified by the physical network node name, Oracle database instance identifier, and the base database name.

Baseline A snapshot of a *design part* or a *project* or *stream* at a particular time. Baselines ensure that the design parts and *items* included in the baseline can be reliably recreated in the future. For example, you might create a baseline before starting a maintenance cycle or assigning further development activities. See also *design baseline, release baseline*, and *archive baseline*.

Baseline pedigree See *pedigree*.

Baseline replication The transfer of baselines and the items contained in the baselines from a master site to one or more subordinate sites.

Baseline template

A set of rules that determine which *items* to include or exclude in a *release baseline* or an *archive baseline* based on the item type, revision, status, and *relationships*. There are two types of baseline template:

- An item baseline template contains a list of criteria relating to item types that is used to construct the baseline.
- A request baseline template contains a list of criteria for selecting *requests* whose related items are used to construct the baseline.

Baseline templates are defined in the Administration Console. For more information, see the *Process Configuration Guide*.

Baseline type

An attribute of a baseline that determines which user-defined attributes and *lifecycle* are used for the baseline.

Baseline types are defined in the Administration Console. For more information, see the *Process Configuration Guide*.

Batch mode

A method of running a group of Dimensions commands together as a script—for example, by running a *command file*.

Branch

A chain of *item revisions* that follows its own update path. Branches allow *product* versions to be developed in parallel. For example, a software product might have separate branches for concurrent development of a major release and a maintenance version of the product. Branches can be *merged* back into the main development path.

Breakdown relationship

A relationship in which a *design part* (child) is owned by another design part (parent). Breakdown relationships determine the inherited responsibilities for the items or requests associated with a particular design part segment.

See also *usage relationship*.

Build

To use Dimensions Build to process one or more *item revisions* into some other form, which can be anything from a single item revision, such as an executable file, to an entire *product*.

Catalog list

See *primary catalog*.

Category

See *request category* and *design part category*.

Change document See *request*.

Change Manager A Dimensions-defined *role* for the *user* who has special authority and privileges in the handling of *requests*.

Changeset A changeset is a logical grouping of changes that is created automatically every time that you deliver changes to a stream or project in a repository. Each changeset creates a new version of a stream or project.

Check in To return a currently checked-out item to the *Dimensions database*.

Check out To extract an *item* from the *Dimensions database* so you can work on it. When you check out an item, Dimensions creates a new *item revision* and places it in your *working location*. The revision is locked in Dimensions CM so no one else can change it. When you finish working with it, you *check in* the item.

Client node A network client (any DFS or NET node) that requires access to the Dimensions server.

Code page Defines the method of encoding characters. A code page encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, which is defined by the network connection or set individually when running commands from the Dimensions command mode.

Command A Dimensions function followed by parameters and qualifiers. For more information, see the *Command-Line Reference*.

Command file A text file whose contents include one or more Dimensions *commands*. You can run a command file in *batch mode* using the Dimensions `CMD` command. For more information, see the *Command-Line Reference.*

Command mode A method for using Dimensions *commands*. Command mode involves entering Dimensions functions at the operating system prompt, in the Dimensions Execute Command dialog box, or by running a *command file*. For more information see the *Command-Line Reference*.

Configuration The structure of *design parts* in a *product*, or a part of that *design structure*.

Console window In the desktop client, a window that shows Dimensions *commands* and the results of those commands. You can type commands directly into the console window.

Content area In the *web client*, an area of the screen that displays lists of *objects*. For example, on the My Current Project/Stream tab, the content area shows the *items*, *requests*, and *baselines* that are assigned to you.

Content window In the *desktop client*, an area of the screen that displays lists of *objects* or object trees, depending on your current task. For example, the Request Catalog window shows all *requests* in a *product*. A content window can be opened, closed, and minimized, and you can switch between open content windows.

Customer The recipient of one or more *releases*.

Daemon A process that sits in the background listening for TCP/IP connection requests. A daemon starts another process to perform operations required by the connection request.

Database server node The node in the network where the Dimensions database is located.

Delegate

- To transfer responsibility for a *request, item*, or *baseline* from yourself to another *user* who does not currently have a *role* for the object. Delegating an object overrides the normal role assignments made through the *design structure*.
- To assign a particular role to a user.
- To assign the *owning site* of a request to another Dimensions CM replication site.

Deliver The process of updating a *project* or *stream* in the repository with the changes made to the corresponding files in the work area.

Delta release A *release* that includes only the *items* that have changed since the previous release.

Demote

To move item revisions, baselines, or requests to a lower stage in the *Global Stage Lifecycle*. When item revisions are demoted they may then be automatically or manually *rolled back*. When you demote a *request*, all the item revisions related to that request, or any of its child requests, are demoted. When you demote a *baseline*, all the item revisions in the baseline are demoted.

Deploy

To copy the item files for selected item revisions to the area(s) corresponding to their stage in the *Global Stage Lifecycle* for the *project* or *stream* to which they belong. You cannot deploy item revisions to areas corresponding to a stage that is greater than the stage to which they have been have been *promoted*. When you deploy a *request*, all the item revisions related to that request, or any of its child requests, are deployed. When you deploy a *baseline*, all the item revisions in the baseline are deployed. Deployment occurs automatically when the items are *promoted* if the corresponding areas are defined as *Deploy by default*.

Deploy by default

An option specified for a *deployment area* such that deployment is automatically scheduled to occur when item revisions are *promoted* to the corresponding stage in the *Global Stage Lifecycle* for that area when it is associated with their *project* or *stream*.

Deployment area

An *area* that is defined for a *project* or *stream* for *Deploying* item files that have reached a particular stage in the *Global Stage Lifecycle.*

Derivative

When comparing projects, one of any number of *projects* being compared to the *ancestor* in the *Project Merge Tool*.

When comparing files, one of any number of files being compared to the *ancestor*.

Design baseline

A snapshot of the current *product design structure*, or a selected portion of it, within the scope of the current *project* or *stream* or *design part*, including all the *revisions* for each *item*.

Design baselines provide managers with audit capabilities. You can compare two baselines to see how the product project/ stream has developed during the period between them. Developers can continue to modify items included in a design baseline.

Design part

A logical component of a *product*. Design parts are related groupings of *objects,* such as modules of code, specifications, and requests. Each design part represents a conceptual component of the *design structure* of the product. See also *top-level design part*.

Design part category

An *attribute* of a *design part* that indicates its general purpose. All *custom lists* of a design part have the same category. Categories are defined as part of the *process model*. For more information, see the *Process Configuration Guide*.

Design structure

The hierarchical structure that identifies the *design parts* of a *product* and the relationships between them. See also *breakdown relationship* and *usage relationship*.

Desktop client

A Microsoft Windows-based application that you use to access the *Dimensions database*. This has also been known as PC Client.

Dimensions base database

See *base database*.

Dimensions CM database

The collection of Dimensions data stored in a structured relational format.

Dimensions network

The collection of nodes that have been defined via the Administration Console or the Network Administration command line interface that form a LAN or WAN network.

Dimensions CM System Administrator

The *user* responsible for creating, deleting, and maintaining the *Dimensions database* as well as other administrative tasks.

Directory item — An entire directory/folder, including subdirectories and files, that is compressed and stored in Dimensions as a single *item*. A directory item is uncompressed when you *check out* or *get* a copy of it in your *working location*.

Display bar — The navigation area in the *desktop client* that enables you to open various *content window*s.

Distributed file system (DFS) — A type of physical network node that can host a Dimensions database and store Dimensions item libraries.

Filter — A set of search criteria that you use to locate and display matching *object*. You can save and reuse filters.

Format — See *item format.*

Format template — A text file that defines the initial content of an *item* or a *request* that is created without content. Item format templates can include *header substitution variables*.

Function — See *command*.

Get — To extract a copy of an *item revision* from the *Dimensions database* so that you can view it without checking it out. When you get an item revision, Dimensions leaves the revision in its current *state* (checked in or checked out) and creates a read-only copy in your *work area*.

Global project — The *project* that contains all of the projects/streams and items in the base database. Any structural changes made in a project or stream, such as adding, deleting, and moving files, are also reflected in the global project.

The global project is named $GENERIC:$GLOBAL and cannot be deleted. *Users* assigned to the global project can reference any item revision in any product in the base database to which they are connected.

Global Stage Lifecycle — The Global Stage Lifecycle is the lifecycle that items follow that controls which versions are included in configurations and builds of the *project* or *stream*. This lifecycle is defined for the base database. Deployment areas for a project/stream can be associated with these stages so that item files are copied to those areas when they are *deployed* to the corresponding stage.

Group

A set of users to which you can assign *privileges*. Groups provide a manageable way of granting and denying privileges to similar sets of users.

Header substitution variables

Placeholders for text in an *item* that are expanded dynamically when you *check out*, *get*, or preview an item. Header substitution variables must be placed within an item header. You can embed multiple headers anywhere in a file.

Held list

The list of *requests* that you have created but not yet submitted because you are still working on them. An example is a request that requires a yet-to-be-created screenshot.

Inbox

A list of requests, items, and baselines that are currently waiting on you for further action. An object is added to your inbox when it is actioned to a state in which you have a role.

Item

An *object* that represents the physical implementation of a *product* component. An item could be source code, an executable file, a document, or an image file. Items can also represent data that does not reside in the file system, such as hardware or database objects.

Item format

An *attribute* associated with an *item type* in the *process model* that indicates the internal structure of those items. The item format determines how *items* of that type are processed during a *build*. In the *web client*, the item format sets the MIME type for the item, which determines how item content is displayed in your web browser.

Item ID

An attribute that identifies an *item*. The item ID may be automatically generated from the *item type* and item *project filename*, or it may be typed in when the item is created.

The same item ID may be used for a "family" of items whose contents have the same origin, but may:

- Serve different purposes (different *item types*)
- Exist to meet different requirements (different *custom lists*)
- Be at a different stage of modification (different *revisions*)

Item library A directory/folder that holds the data files for one or more *item types* in the *Dimensions database*. It can be located on a different network node from the Dimensions database containing the items.

Item pedigree See *pedigree*.

Item replication A process by which Dimensions items are transferred between projects. These projects may reside in the same base database or on different databases resident locally or remotely.

Item revision A specific instance of an *item*. Whenever you modify and *check in* an item, a new item revision is created and stored in the *Dimensions database*. Each revision has a full set of *attributes*, such as modification date, reason for change, and author's name, so you can trace the history of all changes to an item. Revisions are numbered according to the *process model*.

Item specification The unique identifier for an *item*. Item specifications have the following form:

productID:*itemID*.*variant-itemType*;*revision*

See also *product ID*, *item ID*, *variant*, *item type*, and *item revision*.

Item type An *attribute* of an *item* that indicates the general purpose of that category of item. For example, item types can include source files, documentation files, specifications, or object files.

Leader An *attribute* of a *role* that gives the holder permissions and responsibilities above those of other users with the same role. The leader attribute is set up in the *Administration Console*. For more information, see the *Process Configuration Guide*.

Library Cache Area An *area* that is defined in order to contain copies of files whose items are located on a database on a remote server. The purpose is to improve Dimensions CM file get performance. When a user requests a copy of a file by using an operation such as get or check out, Dimensions CM first looks in the library cache area to see if it is already there before attempting to retrieve it from the remote item library.

Lifecycle The set of *states* and rules for transitions between states defined for a particular *object* type and *design part* in the *process model*.

Local item replication The transfer of Dimensions items from and to projects that have all been defined within the *same* Dimensions base database.

Logical node The user-defined alias of the *remote node*. A physical node can have more than one logical name. You can specify an asterisk in this field, to denote that a particular node can connect to any node in the Dimensions Network.

Login profile The login details for your connection to the *Dimensions database*, including your user ID and the database name. A login profile saves all of your connection information, except for your password.

Main catalog See *primary catalog*.

Master base database A Dimensions base database that is defined within the context of a replication configuration as the source for a replication process.

Master DB site The site upon which the master base database is located.

Master project The project in the master DB base database that acts as the source for a replication process.

Merge To view the differences and resolve the conflicts between two text *items* or projects with the result being a single item or project.

Merge configuration file A text file used to control *merge* options when starting the *Project Merge Tool* from the command line.

Merge project A set of *item revisions* in a *project*, *baseline*, or folder/directory that you want to compare or *merge*.

Merged baseline The composite of two or more *release baselines*, merged baselines, or *revised baselines* and the selected *design part*.

Named branch A valid name for a branch in a particular project. Revisions in a branch include the branch name and the revision ID, separated by a pound (#) sign. For example, the first revision in a named branch called `win` is `win#1`.

NET A type of physical network node that can host a Dimensions database, but not Dimensions item libraries.

Network object For TCP/IP this is a socket listed in the services file or its equivalent. It is used by the networking software to communicate between nodes in the network.

Node See *remote node*.

Notification An email message that is sent to *subscribed* users and groups when a particular event occurs.

Object A term used to refer collectively to these object types: *design parts*, *items*, *requests*, *project/streams*, *baselines releases*, and *customers*.

Object list A list of *objects* presented in table form in the content area of the web client.

Object tree A tree view of a hierarchical *object*. Dimensions displays the *project* structure and the *design structure* of a *product* in an object tree.

Oracle service The name of the NET8 service.

- On UNIX this is defined in the Oracle file *tnsnames.ora.*
- On Windows this is defined using the **Net Configuration Assistant** option.

Oracle SID The Oracle system identifier that identifies the database instance.

Originator The creator of a Dimensions *object*. The *role* of $ORIGINATOR is automatically assigned to the creator of a Dimensions object for the duration of its *lifecycle*. This role may be authorized to *action* the object.

Owned The relationship of an *item* to a *design part* when the item was created or moved. This relationship indicates the part of the product to which the item belongs. All *revisions* of the item have this relationship to the same design part.

Owning site The replication site that has ownership of a *request* when there is replication of requests between Dimensions base databases. The site that owns the request is able to update it, whereas other sites can only view its details.

Part Same as *design part*.

Part category Same as *design part category*.

Part change status The field in the *part specification* that identifies the modification level of the *design part*. Previous versions of the design part are Closed. Only the latest version has the Open part status.

Part ID See *part specification*.

Part specification The unique identifier for a *design part*. Design part specifications have the following form:

productID:*partID*.*variant*;*pcs*

See also *product ID*, *part ID*, *variant*, and *part change status*.

Pedigree A graphical representation of the history of an item, project/ stream, or baseline. The pedigree shows how objects are related in time and origin.

Pending list See *inbox*.

Permission See *privilege*.

Physical node The actual machine name of the *remote node*. You can specify an asterisk in this field, to denote a particular node can connect to any node in the Dimensions Network.

Placeholder item revisions Placeholder items serve to maintain the *pedigree* of replicated items for item revision "holes." They are owned by the recipient's Global project.

A placeholder item is also a term that refers to items used for files created as a result of a build by build preservation policies.

Primary catalog The storage area for active requests. Requests that become inactive may be moved to the secondary catalog by someone with the necessary privilege.

Privilege	The ability to perform certain operations, such as creating a project or actioning a request. Administration privileges are set for a base database, and product-level privileges are set for a particular product.
Privilege rule	The conditions under which a user is granted a given privilege.
Process model	A set of user-defined controls that govern the design, development, and maintenance of a product. The process model incorporates your company's change, version, process, and build policies in order to regulate and improve your development environment.
Product	The top-level *object* type that provides the context for managing development with Dimensions. Every object belongs to a product. The *design structure* of a product is modeled as a set of related *design parts*.
Product ID	An *attribute* that uniquely identifies a *product*.
Product Manager	A Dimensions-defined *role* for the *user* who has special authority and *privileges* in setting up the *process model* rules for an individual *product* and managing the development process.
Profile	See *user interface profile*.
Project	A project is a collection of item revisions that are relevant to a development activity. A project differs from a *stream* in that it can contain parallel versions of the same items.
Project filename	A name that identifies an *item* within a *project*. The project filename may or may not be the same as the actual filename of the item. Also, different projects/streams may refer to the same item using different project filenames.
Project Manager	A Dimensions-defined *role* for the *user* who has special authority and privileges in the handling of projects. The Project Manager sets up and maintains the *project folder* structure, adds or deletes *items*, sets the project options that regulate how users work within projects, and locks projects to create *baselines*.
Project/Stream folder	A folder in the *project* or *stream* structure that contains *items*.

Project Merge Tool — A *Dimensions* tool that enables you to compare or *merge* projects. When you compare projects, you view the differences between two projects. When you merge projects, you view the differences between two or more projects and resolve the conflicts, with the result being a single project.

Project pedigree — See *pedigree*.

Project root folder/ directory — The top-level folder/directory that identifies your *work area* for a *project* or *stream*. Items are checked out, checked in, and copied to the working location relative to their location in the *project/stream directory*.

Promote — To move item revisions, baselines, or requests to the next stage in the *Global Stage Lifecycle*. When item revisions are promoted they can then be automatically or manually deployed, meaning that their item files are copied to the corresponding *deployment area* for the *project* or *stream*. When you promote a *request*, all the item revisions related to that request, or any of its child requests, are promoted. When you promote a *baseline*, all the item revisions in the baseline are promoted.

Recipient DB site — A site that receives replicated items or requests from a sender DB site.

Relationship — An association between two *objects*. Each object type has a set of possible relationships with other object types. You can unrelate items, if necessary, depending on the type of relationship and the *state* the item is in.

For example, relationships record whether an item is affected by a *request*, which items are used to build another item, and which items belong to a baseline.

Some relationship types are system-defined, while others are user-defined. The *Product Manager* can set up user-defined relationship types in the *process model*.

Release — A snapshot of a *design structure* or a *baseline* that you can deliver to a customer or use in integration testing. See also *delta release*.

Release baseline

A snapshot of a *design structure* or a *project* at a particular time. You create a release baseline to define and build a test or release of the *product*. A release baseline uses a *baseline template* to select a single version of each *item* that matches the criteria in the template. Dimensions CM locks the design parts and items included in the release baseline.

Release folder/ directory

A file system directory, external to the *Dimensions database*, that contains copies of all the *items* in a *release*.

Release template

A set of rules for selecting which parts of the *design structure* and *item types* to include in a *release*, and which *release directory* to put them in.

Release templates are created in the *Administration Console*. For more information, see the *Process Configuration Guide*.

Remote item replication

The transfer of Dimensions items to a project that resides in a different base database.

Remote node

A computer on your network that contains files and directories that you want to work with in Dimensions. To access these files and directories, you must first log into the remote node from Dimensions.

Replication branch

A named branch that is replicated.

Replication configuration

A controlling definition for a specific replication. It identifies a collection of base database sites (a master and multiple subordinates).

- For an item replication, it specifies a set of named branches to be replicated.
- For a baseline replication, it specifies a set of named branches and item types to be replicated.
- For a request replication, it specifies a set of request types to be replicated.

The configuration also specifies whether a subordinate should replicate items or requests back to the master.

Request

An *object* used to report a defect, suggest an enhancement, or detail other work for a particular *product*. *Requests* can include external files (such as requirements or specification documents) as attachments.

Each *request type* has a *lifecycle* assigned to it that determines which *users* may work on the request at each step in the *process model*.

Request category

One of four groups to which *request types* are assigned. Each category has its own set of *user-defined attributes*.

Request list

A list of *requests* that you can create for your own purposes. For example, you can construct a request list of those *requests* on which you are currently working.

Request replication

A process by which *requests* are transferred remotely between Dimensions base databases.

Request type

An *attribute* of a *request* that classifies the change described, and indicates the general purpose of the request. Valid types are defined by the *Product Manager* as part of the *process model*. A request's type determines which *lifecycle* it follows.

Requirement

An object that represents a planned change to a project. Requests are created in Dimensions RM and are visible in Dimensions CM where you can create *requests* in response to them.

Revised baseline

A *release baseline* that is modified in response to *requests*. Depending on the *relationship* of *item revisions* to requests, items may be removed, replaced or added.

Revision

See *item revision*.

Role

A name, such as Developer or Reviewer, that identifies *users* with similar responsibilities. Roles are used in *lifecycles* to grant *permissions* to *action objects*. Roles are assigned to users according to the object's place in the *design structure* and the user's responsibilities.

Some roles are built into Dimensions CM, while others are user-defined. Roles are created and assigned in the *Administration Console*. For more information, see the *Process Configuration Guide*.

Roll back Rolling back a deployment of items removes the associated item files from a *deployment area* and then automatically *redeploys* the versions that were there previously. When you roll back a *request*, all the item revisions related to that request, or any of its child requests, are rolled back. When you roll back a *baseline*, all the item revisions in the baseline are rolled back.

Role section A set of *attributes* that are relevant to users with a particular *role* in an *object's lifecycle*. Role sections are not restricted to users with that role; they are simply a way of reducing the list of attributes to what the users want to see.

Root folder/directory See *project root folder/directory*.

SDP Standard Dimensions Protocol. Version 2 of the TCP/IP communications protocol is used by Dimensions Network connections.

Secondary catalog The storage area for inactive *requests*. If you are the *Change Manager*, you can move requests from the *primary catalog* to the secondary catalog in order to save space and improve the performance of the primary catalog. Once placed in the secondary catalog, a request can't be updated. Requests can be moved back to the primary catalog.

Secondary catalog list The list of all *requests* placed in the *secondary catalog* by the *Change Manager*.

Sensitive A condition of a *lifecycle state* of an object that requires *authentication* before the user can *action* the object to or from that state. An *attribute* can also be marked as sensitive, which requires authentication before it can be updated.

Merge Tool A tool that enables you to compare or *merge* text *items.*

Server node A node that can host network-accessible Dimensions item libraries. This is any DFS node.

Site The location of a Dimensions base database, specified by a network node, Oracle SID, and a Dimensions base database name.

State An approval level in a *lifecycle* through which an object must pass. States typically reflect states of approval such as development, test, and release.

Status The current *lifecycle state* of an *object*.

Stream A stream is a container in a Dimensions CM repository for a set of item revisions. Streams are used to isolate the development of features from the main code base. Typically each development team uses a child stream that has been branched from a parent mainline stream. The child stream is merged back into the mainline as required.

Streams facilitate an iterative *'copy, modify, merge, and merge'* process where developers:

- Use the *Update* operation to copy item revisions to a local work area from a repository.
- *Modify* the item revisions locally, build, and test.
- Use the *Merge* operation to resolve any conflicts with the mainline; build and test the merged item revisions.
- Use the *Deliver* operation to commit the merged item revisions to the repository.

A stream cannot contain parallel versions of the same item and you cannot use functions that operate on individual files, such as check out and check in.

Subordinate base database A Dimensions base database that is to receive replicated items or *requests*, and which may, optionally, replicate items or requests, back to the master base database.

Subordinate DB sites The sites upon which subordinate base databases are located.

Subordinate project The project in the subordinate base database into which items are to be replicated and from which, optionally, items are to be replicated back to the master base database.

Subscribe The assignment of users and groups to receive an email *notification* when specified events occur.

Substitution variable See *header substitution variables*.

Suspend To remove a *design part* or *item* from further use. The *object* remains in the *Dimensions database* for existing *configurations* and *baselines* that include that design part or item.

Synchronize Wizard A Dimensions tool that enables you to synchronize files in your working location with item revisions in a Dimensions *project* or *stream*. It compares the files and items and determines the updates necessary to put the repository and working location in step with one-another, and optionally, performs the updates automatically. I can synchronize both the work area and repository with one-another (for projects only) or it can perform an *update* or *deliver*.

Target The project or file that results after you *merge* two or more projects or files.See also *ancestor* and *derivative*.

Template A file or set of rules that controls an *object* in Dimensions. See *format template*, *baseline template*, and *release template*.

Tip baseline A snapshot of the latest set of tip revisions for a *project* or *stream.* A tip baseline contains only the latest revision of each item and does not use a *baseline template.*

Top-level design part The top-most *design part* of a *product's design structure*. It is created when the product is defined. For more information, see the *Process Configuration Guide*.

Update The process of updating a work area with the changes in the corresponding *project* or *stream* in the repository.

Upload rules The rules for uploading files from a location outside of the *Dimensions database* and saving them in a *project*. These rules determine the *design part*, *item format*, and *item type* for these new *items*. For more information, see the IDE User's Guide.

Usage relationship The relationship between *design parts* that are reused in the *design structure*. See also *breakdown relationship*.

User A member of the project team who has been authorized and given *privileges* to use Dimensions.

User area See *working location*.

User interface profile A set of features that are shown or hidden in the client tools for a particular user or group. UI profiles simplify the UI and enable users to quickly find and focus on their specific tasks. UI profiles do not enforce security. See *privileges*.

User list See *custom list*.

User-defined attributes A set of *attributes* that can be defined to collect custom information for different *object* types. User-defined attributes can be used to customize the layout of *requests* by including them in *format templates*.

Variant An alternative implementation of a *design part* or *item*, usually to meet different standards or customer requirements. Variants are identified by a field in the *part specification* of an *item specification*.

Version A general term for an *item revision*, or for the *part change status* of a *design part*.

Web client A Web-based application that enables you to use a web browser to access the *Dimensions database*.

Work area A location on your local hard drive, on a *remote node*, or on a network drive that you use to *check out*, *check in*, *get*, and add *items*. This working location is a value that you set as the default for a project you are working on.

A work area can also be a Dimensions-configured *area*, which is an area defined in the database referencing a location on disk to be used for files under development.

Index

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W