SERENA®

DIMENSIONS® CM 14.3.3

Dimensions CM Build Tools User's Guide

Serena Proprietary and Confidential Information

Publication date: May 2017

Table of Contents

Part 1 **Overview of Dimensions Build** 13

Chapter 1 **About Dimensions Build** 15

Introduction 16
Setting Up and Configuring Dimensions Build 17
Pre-Requisite Knowledge 17
Setting up a Build 18
Configuring the Build Flow 19
Dimensions Build Terminology 25
Build Areas 26
Build Area Locations 26
Build Area Security and Ownership 26
Managed Development Areas 27
Build Areas and Active Item Revisions 28
Search Paths 28
Build Roles and Privileges 29
Capturing Build Outputs 29
Preservation Rules and Policies 30
Placeholder Item Revisions 30
Auditing Build Areas 31
Templates 31
Impacted Targets 31
Support for Design Parts 32
Support for Wildcards in Build Configurations 32
Build Ordering 32
Build Ordering Prior to Dimensions CM 12.2.1 32
Build Ordering in Dimensions CM 12.2.1 (and later) 34
Mainframe Wildcard Use Cases 37
Using Item Formats to Control Build Rules 41
Using Build Item Types 41
How Does Dimensions Build Collect Outputs? 42
Build Support 43
Checking Item Revisions are at the Requested Deployment Level 43
Configuring Build Error Messages 44
Upgrading a Pre-Dimensions CM 12.1 Database Table 45
Submitting Parallel Builds through the Deployment Server Queue 45
Parallel Build Rules 46
Disabling Parallel Builds 46

Chapter 2 **Overview of the Build Administration Console User Interface** 47

About Dimensions Build 48
Build Administration Console Main Window 48

Build Management Tab 49
Build Scheduling Tab 52
Build Monitoring Tab 53
Notification Tab 55

Part 2 Configuring and Managing Dimensions Builds 57

Chapter 3 **Configuring an MVS Build Environment 59**
Pre-Requisites 60
Setting up your MVS Build Environment 60

Chapter 4 **Managing Dimensions Build Settings 61**
About Build Settings 62
Build Configuration Types 62
About Build Configuration Types 62
Viewing Build Configuration Types 64
Adding Build Configuration Types 64
Adding Options to Build Configuration Types 64
Modifying Build Configuration Types 65
Modifying Options for Build Configuration Types 65
Deleting Build Configuration Types 66
Deleting Options from Build Configuration Types 66
Build Tools 66
About Build Tools 66
About Build Options 67
Viewing Build Tools and Options 69
Adding Build Tools 69
Adding Options to Build Tools 70
Importing Build Tools 70
Modifying Build Tools 71
Modifying Build Tool Options 71
Deleting Build Tools 72
Deleting Build Tool Options 72
Build Option Groups 72
About Build Option Groups 72
About Build Options 74
Viewing Build Option Groups 74
Adding Build Option Groups 74
Adding Build Options to Build Option Groups 75
Modifying Build Option Groups 76
Modifying the Options of Build Option Groups 76
Deleting Build Option Groups 77
Deleting Options from Build Option Groups 77
Transition Rule Templates 78
About Transition Rule Templates 78
About Build Options 78
Viewing Transition Rule Templates 79
Adding Transition Rule Templates 79

Modifying Transition Rule Templates 80
Deleting Transition Rule Templates 81
Adding Input Masks to Transition Rule Templates 81
Modifying Transition Rule Template Input Masks 82
Deleting Input Masks from Transition Rule Templates 82
Adding Output Masks to Transition Rule Templates 83
Modifying Transition Rule Template Output Masks 83
Deleting Output Masks from Transition Rule Templates 83
Adding Build Options to Transition Rule Templates 84
Modifying Build Options for Transition Rule Templates 85
Deleting Build Options from Transition Rule Templates 86
Adding Build Option Groups to Transition Rule Templates. 86
Deleting Build Option Groups from Transition Rule Templates 87
Using Build Templates with Build Configurations 88
Specifying Build Templates 88
Specifying a Chain of Build Steps 89
Specifying Multiple Inputs 90
Specifying Multiple Outputs 90
Application Rule Templates 91
About Application Rule Templates 91
Viewing Application Rule Templates 92
Adding Application Rule Templates 92
Modifying Application Rule Templates 92
Deleting Application Rule Templates 93
Adding Transition Rule Templates to Application Rule Templates 93
Removing Transition Rule Templates from Application Rule Templates 94

Chapter 5 **Managing Dimensions Build Configurations 95**
About Build Configurations 96
About Build Projects 97
Basic Build Configuration Operations 98
Viewing Build Configurations. 98
Creating Build Configurations 98
Modifying Build Configurations 99
Copying Build Configurations 100
Deleting Build Configurations 100
Setting Build Configuration Properties 101
Build Areas 102
About Build Areas 102
Creating a New Work Area 102
Creating a New Deployment Area 104
Creating a New Library Cache Area 105
Modifying Build Areas 106
Deleting Build Areas 107
Sources 107
About Sources 107
Adding a Build Source from a File 107
Using Wildcard Patterns to Add Build Sources from Dimensions 108

Adding Individual Files in Dimensions as Build Sources 109
Recursive Search. 110
Modifying Build Sources 110
Deleting Build Sources 111
Build Targets. 111
About Build Targets 111
Creating Build Targets 111
Modifying Build Targets 115
Deleting Build Targets 115
Inputs 116
About Inputs. 116
Adding a New Target as an Input 116
Adding a New Source as an Input 116
Adding an Existing Target as an Input 117
Adding an Existing Source as an Input 117
Modifying Inputs 117
Deleting Inputs 118
Outputs 118
About Outputs 118
Scripts 119
About Scripts 119
Build Configuration Scripts 119
Transition Scripts 119
Reusing Scripts 120
Adding a Script 120
Viewing a Build Configuration Script 122
Viewing a Transition Script 122
Modifying a Build Configuration Script 122
Modifying a Transition Script. 123
Deleting a Build Configuration Script 125
Deleting a Transition Script 125
Build Options. 126
About Build Options 126
Creating Build Options 126
Modifying Build Options 127
Deleting Build Options 128
Creating Build Option Groups 128
Versions 128
About Versions 128
Exporting and Importing. 129
About Exporting and Importing 129
Exporting a Build Configuration 129
Importing a Build Configuration 129
Importing an Ant build.xml File 130
Importing an Openmake Build Configuration. 131
Copying a Project with its Build Configurations 131
Tips 132

Chapter 6 **Executing Builds in the Build Administration Console 133**
About Executing Builds 134
Running a Build Configuration 134
Running a Scheduled Build 136
Running a Non-Scheduled Build from the Build Scheduling Tab 136

Chapter 7 **Using Filters 137**
About Using Filters 138
What Can Be Filtered 138
Filtering Build Jobs by User Name 138
Filtering Build Jobs by Date and Time 139
Filtering Dimensions Projects 140
Editing Filter Conditions 141
Altering the Definition of a Filter Condition 141
Removing Specific Filter Conditions 141
Removing All Filter Conditions 141

Chapter 8 **Versioning 143**
About Versioning 144
Example of Rollback 144
Example of Rewrite 146
Deleting Versions Not Allowed 146

Chapter 9 **Scheduling Dimensions Build Jobs 147**
About Scheduling 148
Managing the Scheduler Service 148
Configuring the Scheduler Service 148
Filtering Dimensions Projects 149
Managing Build Job Schedules 149
Viewing Scheduled Build Jobs 149
Adding Build Job Schedules 149
Scheduling the Frequency and Range of Recurring Build Jobs 152
Modifying Build Job Schedules 153
Modifying the Frequency and Range of Recurring Build Jobs 155
Deleting Build Job Schedules 156
Checking the Results of Scheduled Build Jobs 156
Running Builds from the Build Scheduling Tab 156

Chapter 10 **Monitoring Builds in the Build Administration Console 157**
About Monitoring Builds 158
Reviewing Build Status 158
Interpreting Build Monitor Events 159
Viewing Build Details 162
Monitoring a Running Build 163
Canceling a Running Build Job 164
Monitoring Past Builds 164
Viewing Build Execution History 164
Viewing Dependencies 165

Automatic Detection of Dependencies 166

Chapter 11 **Managing Notifications 167**

About Notifications 168
Formatting Notification Templates and Events 168
Notification Templates 168
Notification Events 169
Notification Subscriptions 170
Managing the Notification Service 170
Configuring the Notification Service 170
Managing Notification Templates 171
Viewing Notification Templates 171
Adding Notification Templates 171
Editing Notification Templates 172
Copying Notification Templates 173
Deleting Notification Templates 173
Managing Notification Subscriptions 174
Filtering Dimensions Projects 174
Viewing Notification Events, Details, and Subscribers 174
Adding Notification Events 175
Editing Notification Events 176
Deleting Notification Events 177
Adding Notification Subscriptions 177
Editing Notification Subscriptions 178
Deleting Notification Subscriptions 179

Part 3 Integrating Dimensions Build with Third Party Build Engines 181

Chapter 12 **Using Dimensions CM Build with Ant 183**

About Using Dimensions Build with Ant 184
Deciding Where to Store the Ant Buildfile 184
Selecting or Creating a Dimensions Project 185
Creating or Selecting Build Areas 185
Importing Ant Buildfiles into a Build Configuration 185
Editing Details of the Imported Targets 186
Editing the Transition Script of an Imported Ant Target 187
Running the Imported Build Job 188

Part 4 Integrating Dimensions CM with Build Management Tools 189

Chapter 13 **Integrating Dimensions CM with Apache Ant 191**

Introduction 192
Registering the Dimensions Ant Task with Ant 192
Using the Dimensions Ant Task 192
Syntax and Attributes 194

Chapter 14 **Integrating the Maven SCM Dimensions CM Provider 197**
What is Maven? 198
Terminology 198
Installing and Configuring the Maven SCM Dimensions CM Provider 200
Downloading and Installing. 200
Specifying User Credentials in pom.xml 200
Specifying User Credentials in an External File 201
Goals 201
Maven SCM Plug-in Commands and Options 202
Commands 202
Options 203
Scenarios 205
Scenario 1: Updating an Existing Stream 205
Scenario 2: Checking Out Using a Copy of pom.xml 206
Scenario 3: Checking In a Modified File 207
Scenario 4: Checking In an Added File 207
Scenario 5: Checking Out and Executing Default Goals 208

Chapter 15 **Integrating Dimensions CM with Jenkins. 211**
Introduction 212
Example: Installing Jenkins. 213
Preliminary Steps 213
Installing Jenkins Plugins 213
Configuring Jenkins System Properties 214
Preliminary Steps 214
Configuring a JDK 214
Configuring Apache Ant 214
Configuring a New Build Job 214
Adding Parameters 215
Selecting the JDK 215
Adding the Dimensions CM Plugin as SCM 215
Adding a Build Step 216
Saving the Build Job 216
Testing the Build Job 216

Part 5 Appendices 217

Appendix A **Troubleshooting Dimensions Build 219**
About Troubleshooting 220
Viewing Errors Listed in Build Monitor Events 220
Specific Errors 221
Area <area name> is in use 221
Attempt to close invalid connection 221
Failed to authenticate to the build agent 222
Failed to find product-specific upload rules 222
No build areas found for the selected configuration 222
Template pbem_stop.cmd open failed 222

You do not have a role to extract item 222
Using Temporary Files to Debug Your Builds 223
Preserving Temporary Files 223
Locating Temporary Files 223
About the BRD File 224
Windows Server 2003 224

Appendix B **Dimensions Build Configuration Symbols 225**
Dimensions Build Configuration Symbols 226
DM_MAX_PBEM_RETRIES 227
DM_BUILD_OPTIMIZE 227
DM_BUILD_OPTIMIZE_EARLY_TEST 227
DM_BUILD_MSGQ_SIZE 227
DM_BUILD_MSGQ_INTERVAL 228
DM_BUILD_MSGQ_STEP 228
DM_BUILDERWS_URL 228
DM_MVS_REXEC_DELETE_JCL 228
DM_BUILD_CLEAN_TEMP_FILES 229
DM_DELIVER_RETRY_TIMEOUT 229
DM_BLD_ERROR_INVALID_REVISIONS 229
DM_BUILD_ORDERING 229
DM_MVS_TZ 230
DM_MVS_SBEM_TEMPNAME 230

Appendix C **Creating and Embedding Build Footprinting 231**
What is Build Footprinting? 232
How does Build Footprinting Work? 233
Setting Up Build Footprinting 234
Enabling Build Footprinting 234
Embedding the Footprint Report Name in the Target 235
Configuring the Footprint Report Name 235
Specifying where the Footprint Report is Checked In 235
The Footprint Report Format 236
Configuring Build Footprinting on MVS 237

Appendix D **Dimensions Build Utility Programs 239**
The Primary Build Execution Monitor 240
Running the PBEM 240
PBEM Parameters 241
Dimensions Configuration Symbols 242
Example JCL for Running the PBEM 242
Overriding the PBEM Host Name 242
The Secondary Build Execution Monitor 244
SBEM Parameters 245
Example JCL for Starting the SBEM 246
Starting the SBEM from USS 247
SBEM JCL Syntax 248
Changes to the SBEM in Dimensions CM 10.1.2 250

MDHLLNK0 251
General Parameters 251
Output Control Parameters 251
Passthrough Control Parameters 252
Example JCL 254
bldcomms 256
Using bldcomms on MVS 257
Loading Multiple Members 257

Appendix E **Dimensions Build Security 259**
Phase 1: REXEC 260
Phase 2: dmlibsrv 260
Phase 3: The PBEM 260
Jobcards 260
Phase 4: The SBEM 261
Phase 5: Output Collection 261
Scheduled Builds 261
Dimensions and Build Area Accounts 261
The Default Userid 262
Dimensions Build Area Security 262
Using Build Areas Outside Dimensions Build 262

Appendix F **The Build Configuration XML File Format 263**
Introduction 264
Qlarius Sample Application 264
XML Example #1 265
XML Example #2 268
Build Configuration XML File Structure 271
General Configuration Items 271
Targets 275
Sources 276
Build Areas 277
Transitions 279
Scripts 280
Build Options and Option Groups 281

Appendix G **Customizing a Build Server 283**
Customizing a Build Server 284

Appendix H **Load Balancing a Build Server 287**
Setting Up Load Balancing 288
Restrictions and Limitations 289

Appendix I **Advanced Build Settings 291**
Using Wait Processing 292
Using the BLD, BLDB, and REXEC Commands 292
Using Structured Information Return (SIR) 292
Using REXEC 293

Using Build Roles to Control User Builds 293
Disabling the Automatic Selection of Targets 294

Index **295**

Part 1

Overview of Dimensions Build

Part 1: Overview of Dimensions Build contains the following chapters

About Dimensions Build 15

Overview of the Build Administration Console User Interface 47

Chapter 1

About Dimensions Build

Introduction	16
Setting Up and Configuring Dimensions Build	17
Dimensions Build Terminology	25
Build Areas	26
Search Paths	28
Build Roles and Privileges	29
Capturing Build Outputs	29
Preservation Rules and Policies	30
Auditing Build Areas	31
Templates	31
Impacted Targets	31
Support for Design Parts	32
Support for Wildcards in Build Configurations	32
Build Ordering	32
Using Item Formats to Control Build Rules	41
How Does Dimensions Build Collect Outputs?	42
Checking Item Revisions are at the Requested Deployment Level	43
Configuring Build Error Messages	44
Upgrading a Pre-Dimensions CM 12.1 Database Table	45
Submitting Parallel Builds through the Deployment Server Queue	45

NOTE For z/OS mainframes, the term directory refers to one or more MVS data set qualifiers at the start of data set names.

Introduction

Dimensions Build is a build management, execution, and monitoring tool that is part of Serena® Dimensions® CM. Dimensions Build enables you to execute builds from the Build Administration Console in the Dimensions CM Administration Console, or from the Dimensions desktop, web, and ISPF clients. For information about building from these clients see the *User's Guide* and the *Dimensions for z/OS User's and Administrator's Guide*.

Supported Platforms

Dimensions Build runs on all major Dimensions CM supported platforms: Windows, Solaris, HP-UX, IBM AIX, Linux, and z/OS mainframe (MVS and USS).

Build Engines

Dimensions Build is build engine independent and integrates with third-party engines on distributed and mainframe platforms.

On the distributed side build managers can use their preferred third-party build engine such as Apache Ant or Serena ChangeMan Builder (Openmake). Dimensions Build integrates with those tools and can import Openmake target definition files and Ant XML build configuration files. For more information see page 181.

On the mainframe side Dimensions Build natively supports the Serena mainframe build utility that is installed with Dimensions for z/OS. For more information see page 239.

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:

- The target definitions including high-level dependencies.
- The templates used to build each target.
- The build area definitions (host, authentication details, file system location, etc.).

For more information see page 95.

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. For more information see page 147.

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). For more information see page 157.

Notifications

You can create and subscribe to e-mail notifications that update you about the progress of your build jobs. For more information see page 167.

Integration with Dimensions CM

Dimensions CM performs the following functions for Dimensions Build:

- Drives the population of deployment areas.
- 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.

Build Administration

Dimensions Build is administered in the Build Administration Console of the Dimensions CM Administration Console. For more information about the user interface see page 47.

Setting Up and Configuring Dimensions Build

Pre-Requisite Knowledge

Setting up, running, and analyzing a build is a complex process that should only be performed by experienced Dimensions users. To successfully use Dimensions Build you should be familiar with:

- Setting up network nodes.
- Setting up deployment and work areas.
- The Global Stage Lifecycle.
- Preservation policies and rules.
- Upload rules for item types.
- Customizing and writing build templates.
- Setting up projects and streams.
- The desktop client or the web client.

For full details see the next section.

Setting up a Build

The table below lists all the steps that you need to perform in Dimensions to set up and run a complete build. For details about each step see the chapter or document that is referenced.

Step		Dimensions Component	See this reference
1	Set up network nodes for all remote build machines.	Dimensions Administration Console	■ System Administration Guide, chapters 18 and 20 ■ Dimensions for z/OS User's and Administrator's Guide, chapter 2
2	Set up build areas (deployment and/or work areas).	Dimensions Administration Console	Process Configuration Guide, chapter 14
3	(Optional) Review the Global Stage Lifecyle and make changes as necessary.	Dimensions Administration Console	Process Configuration Guide, chapter 14
4	(Optional) Configure preservation policies for the Dimensions item types that will participate in your build.	Dimensions Administration Console	■ "Preservation Rules and Policies" on page 30 ■ Process Configuration Guide, chapter 11
5	(Optional) Configure upload rules for the Dimensions item types that will participate in your build.	Dimensions Administration Console	Process Configuration Guide, chapter 10
6	(Optional) Customize and write build templates.	Build Administration Console	Developer's Reference, chapter 8
7	Set up Dimensions projects and/or streams and add items.	Any Dimensions clients	User's Guide, chapters 8, 9, and 10
8	Add deployment areas to Dimensions projects.	Any Dimensions clients	User's Guide, chapter 9
9	(Optional) Configure general build settings.	Build Administration Console	"Managing Dimensions Build Settings" on page 61
10	Create, or import, build configurations for the Dimensions projects.	Build Administration Console	"Managing Dimensions Build Configurations" on page 95
11	Add build areas to the build configurations.	Build Administration Console	"Managing Dimensions Build Configurations" on page 95
12	Add targets to the build configurations. Add sources, scripts, and build options as required.	Build Administration Console	"Managing Dimensions Build Configurations" on page 95

(Sheet 1 of 2)

Step		Dimensions Component	See this reference
13	Test and monitor the build configurations and fix any problems that may occur in the builds. For detailed information about how you can configure each step in the build flow, see the next section	Build Administration Console, desktop client, and web client. In the desktop client you monitor builds in a web browser.	■ "Executing Builds in the Build Administration Console" on page 133 ■ "Monitoring Builds in the Build Administration Console" on page 157 ■ User's Guide, chapter 16
14	(Optional) Set up notifications.	Build Administration Console	"Managing Notifications" on page 167
15	(Optional) Set up build schedules.	Build Administration Console	"Scheduling Dimensions Build Jobs" on page 147

(Sheet 2 of 2)

Configuring the Build Flow

This section describes the flow that is executed when you run a build. The first diagram illustrates the flow on a distributed platform, the second diagram illustrates the flow on an MVS platform. The steps in the table that follows refer to both diagrams. The table also includes information about how you can configure each step in the flow.

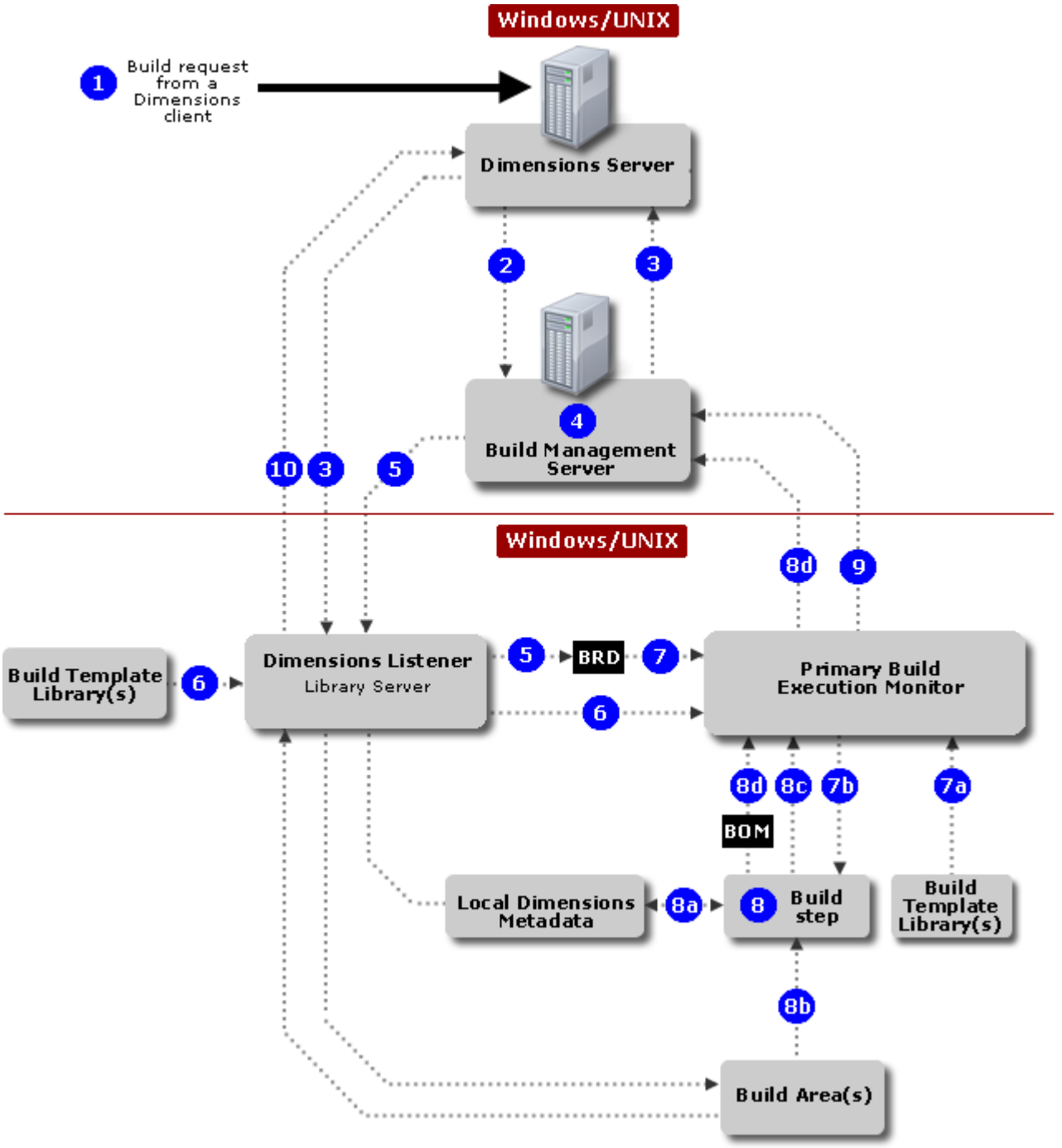


Figure 1-1. The Build Flow on Distributed Platforms

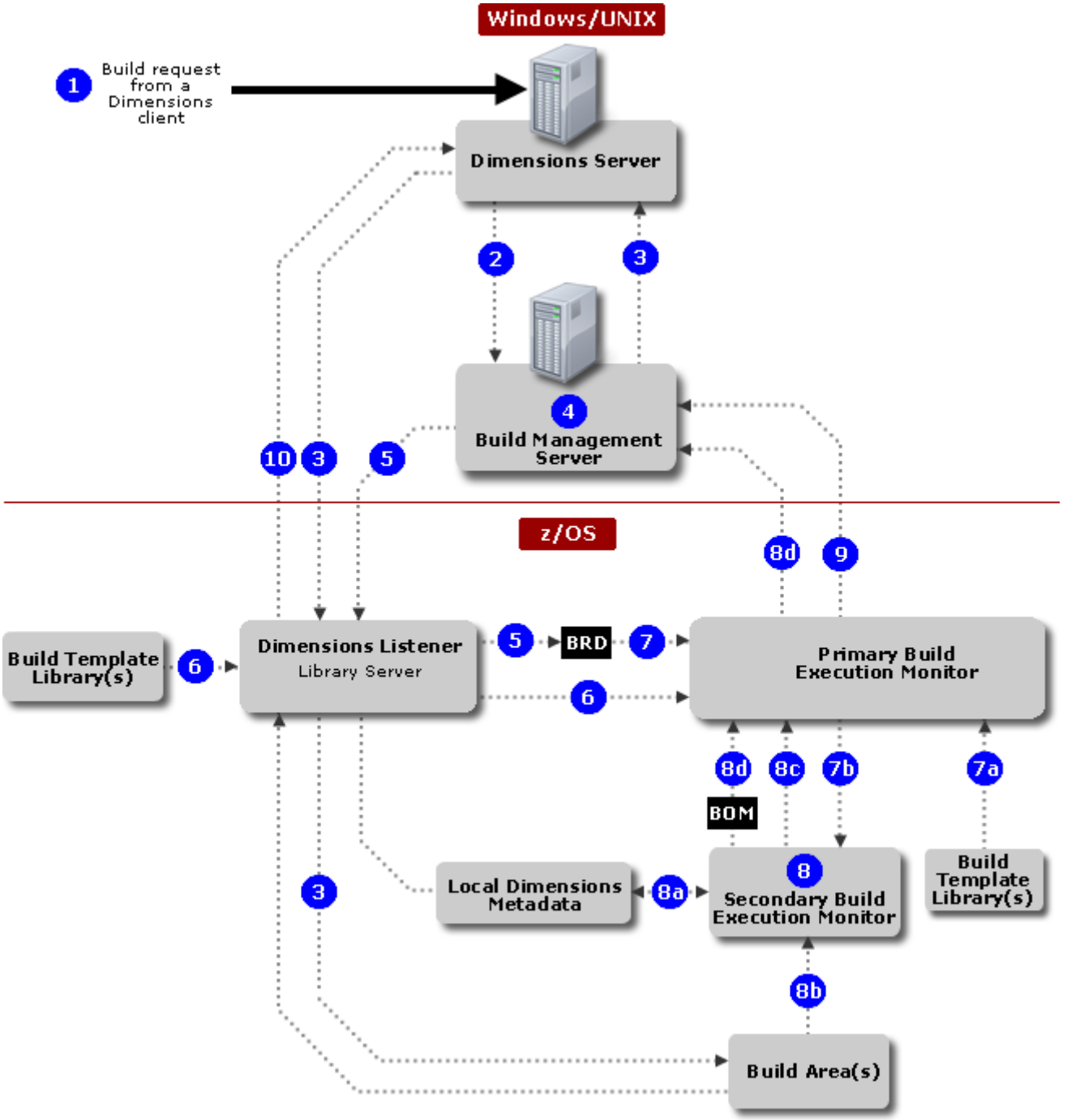


Figure 1-2. The Build Flow on MVS Platforms

Step	Description	How can I configure this step?
1	The build process begins when a Dimensions client sends a build request via the BLD or BLDB commands to the Dimensions server. The command is then submitted to the deployment server and placed in the pending queue for the build area along with all other build and deployment jobs for that area. The command waits in the queue for its turn to be executed. When its turn arrives the deployment server sends the command back to the Dimensions server.	■ For information about configuring the build commands see the *Command-Line Reference*. ■ For information about the deployment server see the *Introduction to Dimensions CM*.
2	The Dimensions server sends a SOAP build request to the Build Management Server with all the necessary information required to run the build (targets, scripts, build areas, build options, etc.).	You can configure the location of the Build Management Server in the following variables in the Dimensions configuration file dm.cfg: ■ DM_WEB_URL: specifies the URL of the default build server. ■ DM_BUILDERWS_URL: specifies the URL of a build server that is different from the default (if the default is not defined, this location is used). For more information see: ■ Appendix B, "Dimensions Build Configuration Symbols" on page 225 ■ Appendix H, "Load Balancing a Build Server" on page 287 ■ *System Administration Guide*
3	The Build Management Server contacts the Dimensions server and instructs it to transfer the required sources, using the services of the Dimensions Listener, to the build area.	Use the /POPULATE_SCOPE qualifier with the BLD command to populate the build areas. The sources that are transferred depend on the definitions in the build request and the build configuration. For more information see the *Command-Line Reference*.
4	The Build Management Server gathers all the information about the build being requested and places it in a Build Request Definition (BRD) file.	You can configure the BRD file by changing the variables in the file web.xml located in: `$DM_ROOT`\Common Tools\tomcat\5.5\webapps\bws\WEB-INF For information see: ■ Appendix G, "Customizing a Build Server" on page 283 ■ *System Administration Guide*

(Sheet 1 of 3)

Step	Description	How can I configure this step?
5	The Build Management Server transfers the BRD file to a temporary location on the build node performing the build via the Dimensions Listener. The Dimensions Listener or Agent then uses a Dimensions REXEC command to invoke an execution of the Primary Build Monitor (PBEM) on that node. The main job of the PBEM is to oversee and control the build process from start to finish.	You can specify where the BRD file is written by configuring the variable DM_TMP in the Dimensions configuration file, dm.cfg, on the build agent node. For information about DM_TMP see the *System Administration Guide.*
6	The PBEM is started via a Dimensions template. The Dimensions Listener then expands the PBEM template and invokes the resulting script (for example, .bat, .sh, and JCL) using the build template library as required.	You can modify the Dimensions templates as required. ■ For information about the Dimensions templating language and the build templates see Part 4 of the *Developer's Reference*. ■ For information about the PBEM and its command parameters see page 240.
7	**Distributed**: The PBEM parses the BRD to determine the build steps to be executed, expands the build templates **(7a)** for each step, and invokes each step **(7b)** as necessary. **Mainframe**: For each build step in the BRD the PBEM performs a process called Build Optimization. This process determines whether the given step needs to execute. For each build step that needs to execute, the build template file that is required to build the step is expanded and executed.	In each build configuration you specify the name of the required build template(s). You can configure the location the of the build templates in the variable DM_TEMPLATE_CATALOG in the Dimensions configuration file. For more information see chapter 7 in the *Developer's Reference.*

(Sheet 2 of 3)

Step	Description	How can I configure this step?
8	As each build step executes it accesses local metadata **(8a)** and the build area(s) **(8b)** to complete the build. As each build step completes, it communicates with the PBEM **(8c)**. The PBEM communicates with the Build Management Server which collects the Bill of Materials (BOM) **(8d)**, collects the build logs, and displays the current status to the user. The BOM is optional on distributed platforms. On a mainframe most of the templates run in the context of a Secondary Build Execution Monitor (SBEM). The main job of the SBEM is to: ■ Oversee and execute a given build step by expanding additional templates as required. ■ Respond to any directives that are in the build template. As a build template is prepared for execution it can initiate the Dependency Monitor to watch specified DD cards for activity and record the results. This information goes in the BOM report to be used later. The build task is then run, outputs are created in the build areas, local metadata is updated, and the final progress and log files are reported back to the PBEM on completion.	■ On MVS, metadata is defined by environment variables. For information see Chapter 2, *Installing Dimensions for z/OS*, and Appendix D, *Setting up Dimensions Metadata*, in the *Dimensions for z/OS User's and Administrator's Guide.* ■ On distributed systems, metadata is saved in the same location as the files. For more information see the *User's Guide* and the *System Administration Guide.* ■ For information about build footprinting and the BOM see "Creating and Embedding Build Footprinting" on page 231.
9	After all the build steps have completed, the PBEM communicates a final status to the Build Management Server and closes.	
10	Output collection retrieves the items from the build area back to the Dimensions repository. Output collection is based on one of the following: ■ Area scanning (no BOM is provided). ■ Using the BOM to determine which targets need to be checked back into Dimensions. Output collection is an optional step.	■ You can trigger output collection using the /CAPTURE qualifier with the BLD command. For information see the *Command-Line Reference.* ■ Upload rules and preservation policies determine what items are checked back into Dimensions and how. For information see the *Process Configuration Guide*.

(Sheet 3 of 3)

Dimensions Build Terminology

Build area The file system location used during a build to look for inputs and generate outputs. A build area definition includes the name of the logical or physical node where the build area is located, the port number running the Dimensions agent, the file system location, and the user credentials used to secure access to the build area. Build areas can be of two types, *work areas* and *deployment areas*. For details see page 26.

Build configuration Contains information about what will be built and how, including the targets, build areas, options, scripts, and dependencies for a specific platform or environment. For details see page 95.

Build configuration type The options that are required for a particular type of build configuration. For details see page 62.

Global Stage Lifecycle The Global Stage Lifecycle (GSL) is the lifecycle that items follow that controls which versions are included in the configurations and builds of a project. Item revisions are moved to the next stage in the lifecycle when they have reached the appropriate stage of approval. The GSL is defined for the base database. For details about editing the GSL see the *Process Configuration Guide*.

Build tool The logical groupings of build options for a given build engine. For example, you can configure all the possible options for the Microsoft Visual C++ compiler and add a description for each option. For details see page 66.

Build option group A logical collection of build tool options that are passed to the build engines and scripts that execute builds. Enables you to apply rules and enforce standards. For details see page 72.

Template A customizable text file containing variables and control words. Dimensions supports a common templating language and processor across Windows, UNIX, Linux, and z/OS (both USS and MVS). The templating language enables you to customize the processing of build, e-mail, and remote job execution templates on all Dimensions nodes. For details see the *Developer's Reference*.

Transition rule template Describes a generic rule for building an item and applying it to specific source items at a later time. It describes how to change or transition an item of one type (i.e., *.c) to an item of another type (for example, *obj) using a template or script. For details see page 78.

Application rule template Defines a group of transition rule templates that are used to create an application. For more details see page 91.

Primary Build Execution Monitor On mainframe platforms the Primary Batch Execution Monitor (PBEM) controls, monitors, and reports results for builds that are initiated by Dimensions Build. On distributed platforms the PBEM run templates for builds that are initiated by Dimensions Build. For details see page 240.

The Secondary Build Execution Monitor The Secondary Build Execution Monitor (SBEM) is an MVS only generalized batch execution tool. For details see page 244.

Build Areas

A build area is the physical location where a build takes place. A build area can be a Dimensions work area or a deployment area.

- A *work area* is an area on your hard drive or a remote node where Dimensions CM performs all file operations such as check in and check out. All operations are relative to the folder that is referenced by your work area or project root folder. A work area can also be shared by a group of users.
- A *deployment area* contains items that have reached a particular stage of development in the Global Stage Lifecycle. When items are promoted to a stage in the GSL you can also move them automatically to the deployment areas associated with that stage. For more information about deployment see the *Introduction to Dimensions CM*.

Build Area Locations

You can associate one or more build areas with each build configuration. You can locate build areas on:

- Your current machine.
- A Dimensions application server machine.
- A remote node, using the normal Dimensions remote node identification mechanism.

A Dimensions agent is required on each machine hosting a build area, for details see the *Installation Guide for Windows*.

On a mainframe machine you can have multiple logical MVS and USS nodes, each running a Dimensions agent. For information about setting up mainframe physical and logical nodes see the *Dimensions for z/OS User's and Administrator's Guide*.

For information about setting up physical and logical nodes all other platforms see the *Process Configuration Guide*.

LPARs

On mainframes there are frequently multiple LPARs where each can:

- Run one or more listeners.
- Have one or more logical MVS and/or USS nodes, implemented by one or more listeners for the related LPAR.

IMPORTANT! A single IP address cannot connect to a collection of listeners over multiple LPARs.

Manipulation of storage by a listener is only dependent on the physical accessibility of that storage by the LPAR that the listener is executing on. There are also catalog structure and security issues associated with using an LPAR to manage arbitrary storage.

Build Area Security and Ownership

When you setup a build area you must specify a Dimensions user ID and password to be able to connect to the area during a build. Dimensions Build does not use the credentials

of the current logged in Dimensions user in case these are not available during a build, for example, when a scheduled build runs at night. You can also specify that the build area password is requested when a build is launched. If the user ID cannot be authenticated on a build area, no request is made for a password and the build does not start.

Each build area that you set up can have a different owner and be on a different machine. However, all the build areas used in a Dimensions project must be on the same network node, or accessible from their network nodes via NFS shares (on UNIX) or network drives (on Windows).

Build operations are performed in the security context of the user who owns the build area. For example, if your UNIT TEST build area is owned by Jim, items are written to that area in the security context of user Jim, regardless of which user name you used to log in to Dimensions. The builds also run as the build area owner, Jim. However the `-lo` parameter remains set to the Dimensions user name you used to log in, and enables you to view the Dimensions Build logs.

The user name submitted to Dimensions Build via the –lo log owner parameter is always the name of the user who logged into Dimensions.

NOTE

- When you create a build area you must ensure that the build area owner has permission to read and write to any files in the directory.
- If you do not specify a build area owner the build area is populated using the credentials of the current user. Serena recommends that you always specify a build area owner.

Build area user credentials are determined by Dimensions in the following order:

1. The credentials of the build area owner.
2. The user credentials specified with the last `AUTH` command for the remote node.
3. The current user's log in credentials. If the user logged in using LDAP credentials, they will probably not be valid operating system user credentials.
4. If none of the above credentials authenticate, you will be prompted with a log in dialog box to enter credentials for the build area node.

Builds at the DEVELOPMENT stage in the current project root directory are submitted and performed in the security context of the user running the Dimensions client. If your project root directory contains a node name, normal Dimensions security applies; the client tries your Dimensions credentials and if authentication fails, prompts you for the correct user credentials.

For more information about build security see Appendix E, "Dimensions Build Security" on page 259.

Managed Development Areas

Managed development areas are deployment areas that are assigned to the first stage in the global stage lifecycle, typically the DEVELOPMENT stage. You can use managed development build areas as integration build areas for source code changes done by a development team when the code is not ready for deployment to a stage such as UNIT TEST or SYSTEM TEST.

Build areas are maintained the same as all other areas. For example, if you create an item revision it is automatically deployed to all managed development areas associated with the first stage. If you deploy an item revision from the DEVELOPMENT to UNIT TEST stages, it is removed from all managed DEVELOPMENT areas.

Build Areas and Active Item Revisions

The areas associated with a stage in the global stage lifecycle only contain active item revisions. An item revision at a stage is only active when there are no newer revisions of the same item at that stage, or any later stage.

- If the item revision becomes the active revision at the new stage, a copy of the item revision is automatically placed in the area(s) mapped to the new stage.
- If the item revision was the active revision at the old stage, the item revision is deleted from the area(s) mapped to the old stage.

For example, if you deploy foo.c from UNIT TEST to SYSTEM TEST:

- A copy of foo.c is placed in the areas associated with the SYSTEM TEST stage.
- foo.c is deleted from the areas associated with the UNIT TEST stage.

Search Paths

A search path is a collection of areas on a single logical node, with the first element of the search path being the area the build is to be performed in. All elements of the search path must be deployment areas except the first. The areas appear in the same order as the Global Stage Lifecycle (GSL), starting with either the GSL level of the deployment area being used for the build,

If you are using a build engine that supports search paths, such as Openmake or the Serena mainframe build engine, set up search paths in Dimensions Build to do the following:

1. Attach your deployment areas to your Dimensions project (for details see the *Process Configuration Guide*).
2. Attach your deployment areas to your build configuration, see "Managing Dimensions Build Configurations" on page 95.
3. Set the DMPATH variable in your build template to list the search paths. The Openmake and mainframe templates supplied with Dimensions will honor those search paths.

For details about integrating with Openmake see page 289.

For details about the Serena mainframe build utility see page 239.

For details about build templates see the *Developer's Reference*.

Build Roles and Privileges

The table below lists the Dimensions privileges that are required to administer and launch builds.

Build Function	Privilege Required
Build administration	Manage build configurations
Build items Build projects	Build from a project
Build requests	Build from a project
Build baselines	Build from a baseline

For more information about privileges and how to define them see the *Process Configuration Guide* or the Administration Console help.

Capturing Build Outputs

Dimensions enables you to capture build outputs and check them into Dimensions. This functionality is also referred to as a closed-loop build. If you choose to capture build outputs the following actions occur in the final stages of a build after all relevant job steps have been completed:

- Dimensions Build captures the build targets defined in the build configuration.
- By default, build targets are checked into the Dimensions project from where you initiated the build. You can choose to put the targets in a different project. The default Dimensions upload rules automatically derive the file format and item type of the outputs (see the *Process Configuration Guide* for details).

  The upload rules for MVS build outputs stored in a PDS/E or PDS, such as MERVK.VT.LOAD(SUBB), are defined in terms of the data set low level qualifier, converted to lower case, for example, `%load`. By default, listing data sets are held per module. The upload rules for generated listings, such as MAL.UT.SUBB.LLISTINGS, are also defined in terms of the data set low level qualifier, converted to lower case, for example, `%llisting`. Listing data sets can also be placed in libraries (PDS/E or PDS) for which the upload rules are defined the same as for other build outputs stored in a PDS/E or PDS.
- Preservation rules specify whether to preserve each build output as a normal, placeholder, external item revision, or none. See below for details.
- The build outputs are created in Dimensions at the initial lifecycle state and the initial stage, for example, Development.
- The Dimensions server creates relationships between every captured target and the source items used to build the target. The server also creates a post-build bill of materials report showing all the components (revisions) that went into the build. This report is stored in Dimensions and any e-mail notifications that have been set up are sent to subscribers.

- The Dimensions server relates the captured outputs to the requests that you optionally specified when you initiated the build.
- If the build used inputs that are not under the control of Dimensions, this information is stored in the Dimensions repository and in the made-of report. For example, if you are using Openmake the search path includes locations that are not defined as Dimensions build areas.

For details about capturing build outputs and viewing items derived from closed-loop builds see the *User's Guide* or the online help for the desktop and web clients.

Preservation Rules and Policies

If you choose to preserve build outputs each built target is mapped by upload rules to a particular Dimensions item type. The rules in a preservation policy define whether each built target is preserved as a normal, external, or placeholder item revision. By default, all built targets are preserved as normal Dimensions item revisions.

Placeholder Item Revisions

Placeholder item revisions represent versioned files that have no content in a Dimensions item library; a placeholder item revision has no storage at all. Typically, you use placeholder item revisions to represent intermediate targets and made-of relationships, avoiding the overhead in Dimensions of preserving every build output generated as a result of a build job.

Physically, a placeholder item revision is represented as a zero-byte file in the item library, and can be created for any build output inside or outside a build area. You can deploy placeholder item revisions between stages. However, since a placeholder item revision does not refer to any files, no build areas are updated; instead, deployment is reduced to a stage change. The AUDIT command ignores placeholder item revisions.

You cannot fetch or extract placeholder item revisions. If you use the UI command in a Dimensions client to revise a placeholder item revision, a normal item revision is created. You can only create placeholder item revisions from build output collections.

For details about creating preservation rules and policies see the *Process Configuration Guide*.

NOTE The preservation policy for a build area is the same as the first stage of the GSL.

Auditing Build Areas

The AUDIT command produces an audit report for the build and deployment areas associated with a Dimensions project. The report compares the files in the Dimensions repository with those in the build areas that you select. You can also repair the build area so that it contains all the item revisions from the project that are at the build stage you selected.

NOTE You cannot audit work areas.

For information about auditing build areas see the *User's Guide* or the online help for the desktop and web clients. For information about the AUDIT command see the *Command-Line Reference*.

Templates

A template is a customizable text file containing variables and control words. In Dimensions, templates are used to execute builds and execute general commands via remote job execution. You can also attach scripts and templates to deployment areas. For details see the *Developer's Reference.*

In addition, you can use templates to construct the body of e-mail messages used for notifications, for details see the *Process Configuration Guide*.

Impacted Targets

Dimensions enables you to view the targets impacted by items, a request, or a project. This is useful when you want to check what targets will be built before you launch a build or what targets will be affected if you change a source file. For details see the *User's Guide* or the online help for the desktop and web clients.

Support for Design Parts

When you add a target to a build configuration, or modify an existing target, you can optionally specify a design part to be related to the target.

The following rules are used in descending order to determine how a design part is assigned to a collected item.

1. If a previous revision of the collected item already exists, the design part for the new revision is the same and is not changed.
2. If the target item rule in the build configuration has a design part specified, that design part is used for the item.
3. The design part from the first input to the transition that has a valid part is used.
4. The design part in the TGT file is used (Openmake projects only).
5. Regular upload rules are used.

Support for Wildcards in Build Configurations

When you add build sources from Dimensions CM you can add individual files or wildcard patterns representing multiple files, for example, COBOL(*). For details about using search paths with wildcards see page 28.

The following options enhance the use of wildcards in build configurations:

- **Item Formats**: When you add a build source or pattern you can optionally specify an item format in the Build Type. This enables you to create multiple rules using the same pattern but have a unique item format or build type to keep each rule unique. For more details see page 41.
- **Expanding Wildcards**: When you add or modify a script for a target you can select the Expand Wildcards check box. This enables you to:
  - Expand all the wildcard inputs and outputs for a transition at build time.
  - Create an instance of the transition for each item that matches the wildcard.

Build Ordering

Build Ordering Prior to Dimensions CM 12.2.1

Prior to Dimensions CM 12.2.1 build ordering works as described below.

A build configuration contains a series of build rules that are executed to build your software. The order in which the rules are executed may be important, for example, you want component X to be built before component Y. Use the following methods to set up build ordering in a build configuration:

- **Implicit Ordering**

  When one target rule (the intermediate target) is used as the input to another target rule (the final target) in a build configuration, the intermediate target must be built before the final target. For example:

```
LOAD(*)
  OBJ(*)
    COBOL(*)
```

  In this example, all of the OBJ(*) build rules execute before the LOAD(*) build rules. This is called implicit ordering as it is determined automatically because of the nesting of build rules in the configuration.

- **Explicit Ordering**

  Optionally, you can assign a numeric value to a target rule using the Build Order field. This value specifies the order in which the rule is built relative to the other rules in the same configuration. For example:

```
LOAD(*) - build order 2
  OBJ(*)
    COBOL(*)

LOAD(*) - build order 1
  OBJ(*)
    PLI(*)
```

  In this example, all the PLI OBJs are created first, followed by the PLI LOADs, then the COBOL OBJs, and finally the COBOL LOADs. Notice how implicit ordering is done within the rules of explicit ordering.

  A numeric build order can only be applied to final target rules in the configuration. If a build order is assigned to an intermediate target, the value cannot be higher than its parent, but it is ignored. Implicit and explicit ordering cannot conflict with each other.

  A value of 0 in a build order indicates no order so the build rules will execute in a random order. By default, rules with no order build before any rules with an assigned order. However, you can change this by changing the value of `force.unordered.transition.execute.later` in web.xml to 'T' (true). This causes items with no build order to run after those with an assigned order.

  NOTE web.xml is located in:

  `$DM_ROOT\Common Tools\tomcat\5.5\webapps\bws\WEB-INF`

  Build order values do not need to be sequential (1, 2, 3, 4…) and can have gaps between the values (5, 10, 15, 20…). You should leave gaps in build orders so that you can insert new build rules later without having to reorder the entire build configuration.

Build Ordering in Dimensions CM 12.2.1 (and later)

In Dimensions CM 12.2.1 and later the default behavior of build ordering has changed. The main benefits of the new behavior are:

- More efficient builds.
- Allows builds that include steps that fail to proceed much further.

Major and Minor Build Order Steps

Build steps can now optionally have major and minor orders. A major order is the number assigned to a group of build steps. Minor orders group the build steps in a major order. Both major and minor orders specify the sequence in which the build steps are processed.

For example, assume that you have the following build steps: S1, S2, S3, S4, S5, S6 and S7. Build steps S1, S2, and S3 are grouped together and have a major order of 1. Build steps S4, S5, S6, and S7 are grouped together and have a major order of 2. All build steps with a major order of 1 are processed before those with a major order of 2.

S1 has a minor order of 1, S2 has a minor order of 2, and S3 has minor order of 3. The build orders for this group are:

S1 (1,1)

S2 (1,2)

S3 (1,3)

Build step S1 is processed first followed by S2 and then S3.

S4 and S5 both have a minor order of 1, and S6 and S7 both have a minor order of 2. The build orders for this group are:

S4 (2,1)

S5 (2,1)

S6 (2,2)

S7 (2,2)

Build steps S4 and S5 are processed first followed by S6 and S7. Both S4 and S5, and S6 and S7, can be built in parallel.

Notes

- The major build order is set by the build administrator when adding targets to a build, for details see page 111.
- Build ordering enables you to create complex configurations of build steps.
- The PBEM stops reading the Build Request Document when the build order changes. If there is a combination of major and minor orders the PBEM then consumes less resources and occupies less memory, which is useful for very large builds.

- If a build step fails you can execute subsequent build steps using the variable DM_ON_ERROR. Therefore, for large builds you can complete the build, and fix the failed steps later, without stopping the entire build process. This makes large build processes much more robust.
- Specifying a build order on other than the last of a chain of build rules has no effect and is ignored.
- Do not mix wildcard and explicit rules in a major build order. This combination is not supported by Serena.

Build Order Example

The minor build order is determined by the target type. The minor build order steps are arranged so that all entities that need to be built as in input to a type are built *before* any of the objects of that type. A type can be an extension of a target.

For example:

- Rule 1: `LOAD(*) <<- OBJ(*) <<- ASM(*) [F1]`
- Rule 2: `LOAD(*) <<- ASM(*) [F2]`

where:

- <<- this transition uses Dimensions CM wildcard expansion.
- [ ] is a build type that restricts the rule selection.

The following major build order steps are processed:

- `ASM(S1) or ASM/S1.ASM format (=build type) = F1`
- `ASM(S2) or ASM/S2.ASM format (=build type) = F2`

Using the above rules:

- All OBJ steps will precede any LOAD steps.
- In rule 1 the target types are OBJ and then LOAD.
- In rule 2 the target type is LOAD.

The minor build order steps are:

- Minor order 1:

```
construct OBJ(S1)
   from ASM(S1)
```

- Minor order 2:

```
construct LOAD(S2)
   from ASM(S2)
construct LOAD(S1)
   from ASM(S1)
```

Notes

- Both steps in minor build order 2 may be run in parallel.
- Minor build order 2 will not start until all the steps in minor build order 1 are complete.
- This major build order will not be complete until all its minor build order steps are complete.

Build Order Variables

The system uses the following build ordering variables:

Variable	Description and Value
DMORDERING	Specifies if the 12.2.1 (and later) build ordering functionality is enabled. If not, build processing uses the pre-12.2.1 functionality. To globally enable or disable build ordering using the `dm.cfg` variable DM_BUILD_DMORDERING. Values: YES and NO Default: YES
DM_ORDER_ZERO_FIRST	Specifies that build orders with a value of zero are processed first. Values: YES/NO Default: YES (if DMORDERING is enabled)
DM_ON_ERROR	Specifies if processing continues when errors are encountered in a group of build steps. If you specify PUSHON, processing continues. Values: PUSHON and GIVEUP Default: PUSHON
DMORDMIN	If DMORDERING is enabled you will see these variables in the BRD file.
DMORDMAJ	

Mainframe Wildcard Use Cases

The following use cases demonstrate the use of wildcards in mainframe build configurations.

Use Case #1

- One source per target.
- Two transition steps: source => intermediate target => final target
- The target name is the same as the patterned COBOL source name.
- The build order is dictated by nested build pattern dependencies.
- No build type property is specified.

Defined Sources

Source Name	Build Type
COBOL(CBATA)	COBOL
COBOL(CBATB)	COBOL
COBOL(CBATC)	COBOL

Build Rules

Build Pattern	Script	Properties
LOAD(*) OBJECT(*) COBOL(*)	SCRIPTA SCRIPTB	Build Type: (none) Build Order: (none)

Generated Steps (in order):

COBOL(CBATA) => OBJECT(CBATA)

COBOL(CBATB) => OBJECT(CBATB)

COBOL(CBATC) => OBJECT(CBATC)

OBJECT(CBATA) => LOAD(CBATA)

OBJECT(CBATB) => LOAD(CBATB)

OBJECT(CBATC) => LOAD(CBATC)

Use Case #2

- One source per target.
- Multiple transition steps:
  - Source => final target
  - Source => copybook
- The target name is the same as the patterned source names.

- The build order is set by the build order property.
- The build type property dictates which source is used.

Defined Sources

Source Name	Build Type
COBOL(CUIA)	COBOLCICS
COBOL(CUIB)	COBOLCICS
COBOL(CBATA)	COBOL
COBOL(CBATB)	COBOL
BMS(CUIAMP)	CICSMAP
BMS(CUIBMP)	CICSMAP

Build Rules

Build Pattern	Script	Properties
LOAD(*) COBOL(*)	SCRIPTA	Build Type: COBOLCICS Build Order: 2
LOAD(*) COBOL(*)	SCRIPTB	Build Type: COBOL Build Order: 2
LOAD(*) BMS(*)	SCRIPTC	Build Type: CICSMAP Build Order: 1
COPY(*) BMS(*)	SCRIPTD	Build Type: CICSMAP Build Order: 1

Generated Steps (in order)

BMS(CUIAMP) => COPY(CUIAMP)

BMS(CUIBMP) => COPY(CUIBMP)

BMS(CUIAMP) => LOAD(CUIAMP)

BMS(CUIBMP) => LOAD(CUIBMP)

COBOL(CBATA) => LOAD(CBATA) (via SCRIPTB)

COBOL(CBATB) => LOAD(CBATB) (via SCRIPTB)

COBOL(CUIA) => LOAD(CUIA) (via SCRIPTA)

COBOL(CUIB) => LOAD(CUIB) (via SCRIPTA)

Use Case #3

- One source per target.
- One transition step: source => final target.
- The target name is the same as the patterned COBOL source name.

- The build order is dictated by nested build pattern dependencies.
- No build type property is specified.

Defined Sources

Source Name	Build Type
COBOL(CPAYA)	COBOL
COBOL(CPAYB)	COBOL
COBOL(CPAYC)	COBOL
COBOL(CBATA)	COBOL
COBOL(CBATB)	COBOL
COBOL(CBATC)	COBOL

Build Rules

Build Pattern	Script	Properties
LOAD(CPAY*) COBOL(CPAY*)	SCRIPTA	Build Type: (none) Build Order: (none)
LOAD(CBAT*) COBOL(CBAT*)	SCRIPTB	Build Type: (none) Build Order: (none)

Generated Steps (in order)

COBOL(CPAYA) => LOAD(CPAYA) (via SCRIPTA)

COBOL(CPAYB) => LOAD(CPAYB) (via SCRIPTA)

COBOL(CPAYC) => LOAD(CPAYC) (via SCRIPTA)

COBOL(CBATA) => LOAD(CBATA) (via SCRIPTB)

COBOL(CBATB) => LOAD(CBATB) (via SCRIPTB)

COBOL(CBATC) => LOAD(CBATC) (via SCRIPTB)

Use Case #4

- One source per target.
- Two independent transition steps:
  - source => intermediate target
  - intermediate target => final target
- The target name is the same as the patterned source name.
- The build order is set by the build order property.
- No build type property is specified.

Defined Sources

Source Name	Build Type
COBOL(CBATA)	COBOL
COBOL(CBATB)	COBOL
ASM(ABATA)	ASM
PLI(PBATA)	PLI
SYSLIN(CBATA)	LINK
SYSLIN(CBATB)	LINK
SYSLIN(ABATA)	LINK
SYSLIN(PBATA)	LINK

Build Rules

Build Pattern	Script	Properties
LOAD(*) SYSLIN(*)	SCRIPTA	Build Type: (none) Build Order: 2
OBJECT(*) COBOL(*)	SCRIPTB	Build Type: COBOL Build Order: 1
OBJECT(*) ASM(*)	SCRIPTC	Build Type: ASM Build Order: 1
OBJECT(*) PLI(*)	SCRIPTD	Build Type: PLI Build Order: 1

Generated Steps (in order)

COBOL(CBATA) => OBJECT(CBATA)

COBOL(CBATB) => OBJECT(CBATB)

ASM(ABATA) => OBJECT(ABATA)

PLI(PBATA) => OBJECT(PBATA)

SYSLIN(CBATA) => LOAD(CBATA)

SYSLIN(CBATB) => LOAD(CBATB)

SYSLIN(ABATA) => LOAD(ABATA)

SYSLIN(PBATA) => LOAD(PBATA)

Using Item Formats to Control Build Rules

When you add a build source to a build configuration you can specify a build type item format. This enables you to add an additional level of granularity and subdivide a large group of sources with the same extension or item type into smaller subsets based on their unique item format. Wildcards are then scoped using a name match and the build type item format.

For example, COBOL source code can be BATCH, CICS, DB2, IMS, or a combination of multiples. They all have a .COBOL extension and are the same item type. However, if you define an item format for each one, you can specify a unique build type item format when adding build sources. For example, you could have the following COBOL build type item formats:

- COBOL_CICS
- COBOL_DBS

If you specify a build type item format, the source is scoped by both the item type (e.g., COBOL(*) and the build type item format (e.g., COBOL_CICS).

For information about specifying Dimensions item formats see the chapter *Data Formats and MIME Types* in the *Process Configuration Guide*.

Using Build Item Types

To use build type item formats do the following:

1 In the *Data Formats & MIME Types* section of the *Configuration Object Management* cluster of the Administration Console define a new data format type for each build type item format.

2 When you create new items in Dimensions, in the Item Format field specify the build type item format.

3 To modify an existing item's format, use the /FORMAT qualifier in the UIA (Update Item Attributes) command.

4 When you are add or modify build sources, in the Build Type field enter the unique item format.

How Does Dimensions Build Collect Outputs?

When a build is launched on a build node, the PBEM (Primary Build Execution Monitor) sends SOAP (Simple Object Access Protocol) messages to the build server at predefined intervals. These messages notify the build server when the build has started and finished, and when the build of an individual target or build step has also started and finished. These messages include logging information from the build step, such as std out and std error, and the Bill of Materials (BOM) report from an individual target or build step, if it is available. During a build, start and finish messages are issued that contain supporting logs and a BOM report for each target that is built.

When a request is made to collect targets from a build, the build server processes each step/target completion message and BOM report to determine which files need to be collected back into Dimensions. The build server perform this task this immediately so that outputs are being collected while the build is running, spreading the network, appsrv, and libsrv load across the build process.

For each target item in a BOM, the build server determines the filename of the item and any secondary files, such as listings, that need to be collected. The build server combines these with attributes from the configuration, such as the design part for the target, and makes a request to the appsrv to store these files in the repository. The appsrv determines whether the file being preserved is a new or existing item, compiles the information required to complete the process, and stores the item in the repository. The appsrv also creates "made of" records in the database for impact analysis and build target calculation by the Dimensions clients.

For build steps that do not produce a BOM report, mainly on distributed platforms, the appsrv collection logic searches the build area matching the file patterns that are provided by the build configuration. All items that match the search patterns are placed in the repository and "made of" records are created.

After the appsrv has stored the items, it sends the item specifications and filenames to the build server. This information is stored in a cache on the build server and used to supplement metadata deficiencies from the BOM data in subsequent build steps. Because output collection is running asynchronously from the build, a final target that has a dependency on an intermediate target created earlier in the same build job may not have the proper metadata information for the intermediate target. This is because the output collection, and the updating of the local metadata, may not have run by the time the final target was being built. The missing metadata information is added by the build server using the local cache it maintains throughout the build of items that have been processed during this build job.

The above process is executed repeatedly until all outputs are collected for a given build job.

Build Support

When an output collection is invoked for each completed build step the target files are collected into a temporary changeset. When the build is complete this changeset contains all of the collected targets for the current build request. When the build server is notified by the PBEM that the build is complete, it notifies the Dimensions server that the temporary changeset can be merged into the stream or project. When the merge occurs request relationships are created. If an error occurs during processing, the temporary changeset is not deleted and a message is returned to the build server log. You can use the DBT (Deliver Build Targets) command to reattempt the delivery at a later time.

Redelivering a Failed Delivery

If a delivery fails, use the DBT command to reattempt the delivery. For details, see the *Command-Line Reference*.

Contents of Temporary Changeset

When a build is complete, the temporary changeset that was used contains all the build targets. Using the "made-of" relationships as a guide, only the build targets used during the delivery are added to the XML file because the source files should already exist in the stream. If these relationships are incorrect or not complete, set the following variable in dm.cfg to TRUE to force all items from the project (sources and targets) to be added to the delivery XML:

```
DM_BLD_MERGE_ALL_ITEMS
```

Checking Item Revisions are at the Requested Deployment Level

You can start a build from a single source or from a set of sources, for example, all the sources related to a request. For a work area, you can populate the area prior to the build starting. For deployment builds, the area should already be populated by Dimensions during deployment so that you can only build the versions that already exist in the area.

In previous releases of Dimensions, the population of a work area was always from the tip version of the sources in the project and not the version you actually chose to build. This was a problem as you were not actually building what you requested. For deployment areas you could not control what was in the area. If you chose a revision of a source that was not at the deployment level that you going to build, you were not notified and ended up building whatever level of source happened to be in the deployment area at the time. This was not necessarily the version that you selected.

Dimensions Build now works as follows:

- For work areas, Dimensions pushes the source revision that you selected or the revisions that are related to the request. Therefore you can build exactly what you requested and not the tip.
- For deployment areas, Dimensions checks that the item revision of the source at the requested deployment level is actually there. This is not a physical check on the build area, rather a verification on the server about what should be there. If it is not, Dimensions displays an error message during the build launch notifying you that the

version you requested to build is not currently in the area that you have requested to build it in. For information about configuring the build error messages, see below.

Configuring Build Error Messages

A build will fail if one or more of the following conditions are met:

- The source file is missing from the search path.
- The source file has a higher revision number than the one that was requested.
- The source file has a lower revision number than the one that was requested.
- The source file is missing from the project/stream.
- The item specification of the source file is wrong.

You can control these errors as follows:

- (Default) To turn off all error messages, display warnings instead, and allow the build to proceed, set the symbol `DM_BLD_ERROR_INVALID_REVISIONS` in the Dimensions configuration file to 'N':

  ```
  DM_BLD_ERROR_INVALID_REVISIONS N
  ```

  Note: If you do not set the symbol the behavior is the same.

- To turn on all error messages and stop the build, set the symbol `DM_BLD_ERROR_INVALID_REVISIONS` in the Dimensions configuration file to 'Y':

  ```
  DM_BLD_ERROR_INVALID_REVISIONS Y
  ```

- To apply more control over which errors stop a build and which do not, specify the appropriate hexadecimal number in the symbol `DM_BLD_GETSRC_FUNCTION_OR` in the Dimensions configuration file:

Warning	Hexadecimal Number	Behavior
NOFAIL_MISSING	0x00100000	Do not fail if the source file is missing from the search path.
NOFAIL_NEWER	0x00200000	Do not fail if the source file has a higher revision number than the one that was requested
NOFAIL_OLDER	0x00400000	Do not fail if the source file has a lower revision number than the one that was requested.
NOFAIL_NOITEM	0x00800000	Do not fail if the source file is missing from the project/stream.
NOFAIL_SPEC	0x01000000	Do not fail if the item specification of the source file is wrong

  For example, if the only warning that you require is `NOFAIL_NEWER`, specify the value 00200000:

  ```
  DM_BLD_GETSRC_FUNCTION_OR 00200000
  ```

You can create compound selections by adding together hexadecimal numbers. This example turns the NOFAIL_NEWER and NOFAIL_OLDER errors into warnings:

```
DM_BLD_GETSRC_FUNCTION_OR 00600000
```

NOTE If you use DM_BLD_GETSRC_FUNCTION_OR it overrides the settings in DM_BLD_ERROR_INVALID_REVISIONS.

For more details about the symbol DM_BLD_ERROR_INVALID_REVISIONS see page 229.

Upgrading a Pre-Dimensions CM 12.1 Database Table

Before an upgraded pre-Dimensions CM 12.1 build system can be run in Dimensions CM 14.x you need to run the utility build_upgrade_nolists.exe. For details see the *Installation Guide for Windows.*

Submitting Parallel Builds through the Deployment Server Queue

In Dimensions CM releases prior to 12.1, build jobs ran on demand. If two builds were submitted at the same time, they ran in parallel with the customer responsible for avoiding clashes between the builds.

Dimensions 12.1 introduced the deployment server. Builds, like any other job that affected a deployment area, were queued, and ran in sequence one at a time.

In Dimensions 12.2.1 and later this strict sequential queuing has been relaxed to allow for parallel build jobs, while preserving the benefits of the deployment queue. This improves the throughput of build jobs in a highly stressed system. Build jobs take longer to run than deployment jobs therefore running multiple build jobs at the same time greatly increases the throughput of the deployment server queue.

Parallel builds affects build jobs submitted through the deployment server. These are the build jobs that you can see in the web client Deployment view tab. Currently the only type of build that is not launched in this way is a work area build, which does not use any deployment areas to supplement the search path. You can run these types of builds in parallel, as in previous releases.

CAUTION! If you have independent teams coding changes to different parts of the system, and the changes overlap, parallel build jobs may cause confusion. The build system does not currently detect overlap between jobs automatically, and you will have to enforce this through working practices. If you are unsure you should disable parallel builds (see below).

Parallel Build Rules

The following rules describe how parallel builds work:

1. Build jobs submitted from the same user client session are always run sequentially. This is essential for the reliable execution of batch scripts, which normally expect their commands to be run strictly sequentially. This rule applies to the client session, not the user ID. The same user can have multiple client sessions.
2. Deployment jobs always run on their own. They do not run at the same time as other deployment or build jobs. This creates a quiet-point in the deployment log, which allows future rollbacks to be computed reliably.
3. Build jobs can run in parallel with other build jobs, even in the same area subject to rule 1.
4. Where there is an outstanding queue of work containing both build and deployment jobs that have not yet started, these jobs are automatically re-ordered to optimize the chances of the build jobs running in parallel. Reordering is designed to produce sequences of multiple build jobs and where possible allow them to run together, therefore improving the throughput of the system. For example:

   If we represent a deployment job as "D", and a build job as "B", reordering is designed to take a queue of work such as "DB DB DB DB DB", and reorder it to produce "DDDDD BBBBB". This latter sequence allows the build jobs an opportunity to run together as all the deployment jobs have already been executed.

   Deployment jobs and build jobs are never run out of sequence. Deploy1 is always run before Deploy2 and Build1 is always run before Build2.

Disabling Parallel Builds

Parallel builds are now the default. You can disable them by adding the following line to the %DM_ROOT%\dm.cfg file on the server:

DISABLE_CONCURRENT_BUILD_DISPATCH 1

If you set this flag the behavior will be the same as for Dimensions 12.1 and 12.2: all build and deployment jobs will be single thread, running one job at a time in the sequence in which they were submitted.

Chapter 2

Overview of the Build Administration Console User Interface

About Dimensions Build	48
Build Administration Console Main Window	48
Build Management Tab	49
Build Scheduling Tab	52
Build Monitoring Tab	53
Notification Tab	55

About Dimensions Build

Dimensions Build is a build management, execution, and monitoring tool that is part of the Dimensions CM Administration Console. For an overview of Dimensions Build see page 15.

Invocation Dimensions Administration Console | Distributed Development | Build Administration

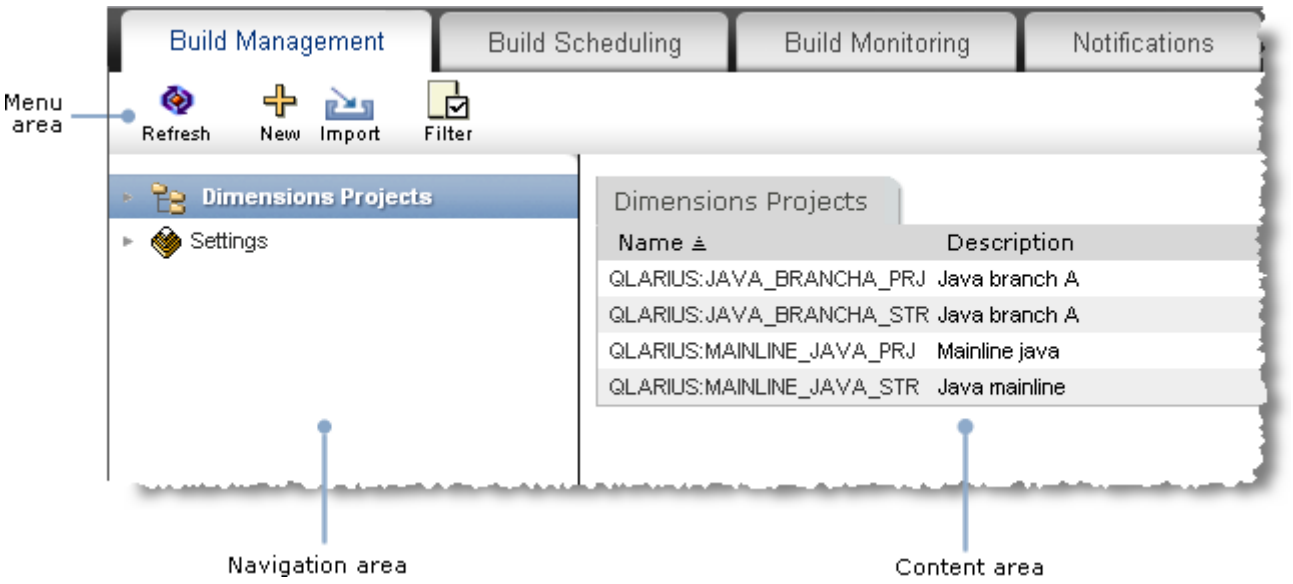
Build Administration Console Main Window

The Build Administration Console main window has the following tabs:

- **Build Management:** Enables you to manage build configurations and settings, see page 49.
- **Build Scheduling:** Enables you to set up build schedules, see page 52.
- **Build Monitoring:** Enables you to monitor the progress of your builds and view the history of previous builds, see page 53.
- **Notifications:** Enables you setup emails that notify you about the progress of your builds, see page 55.

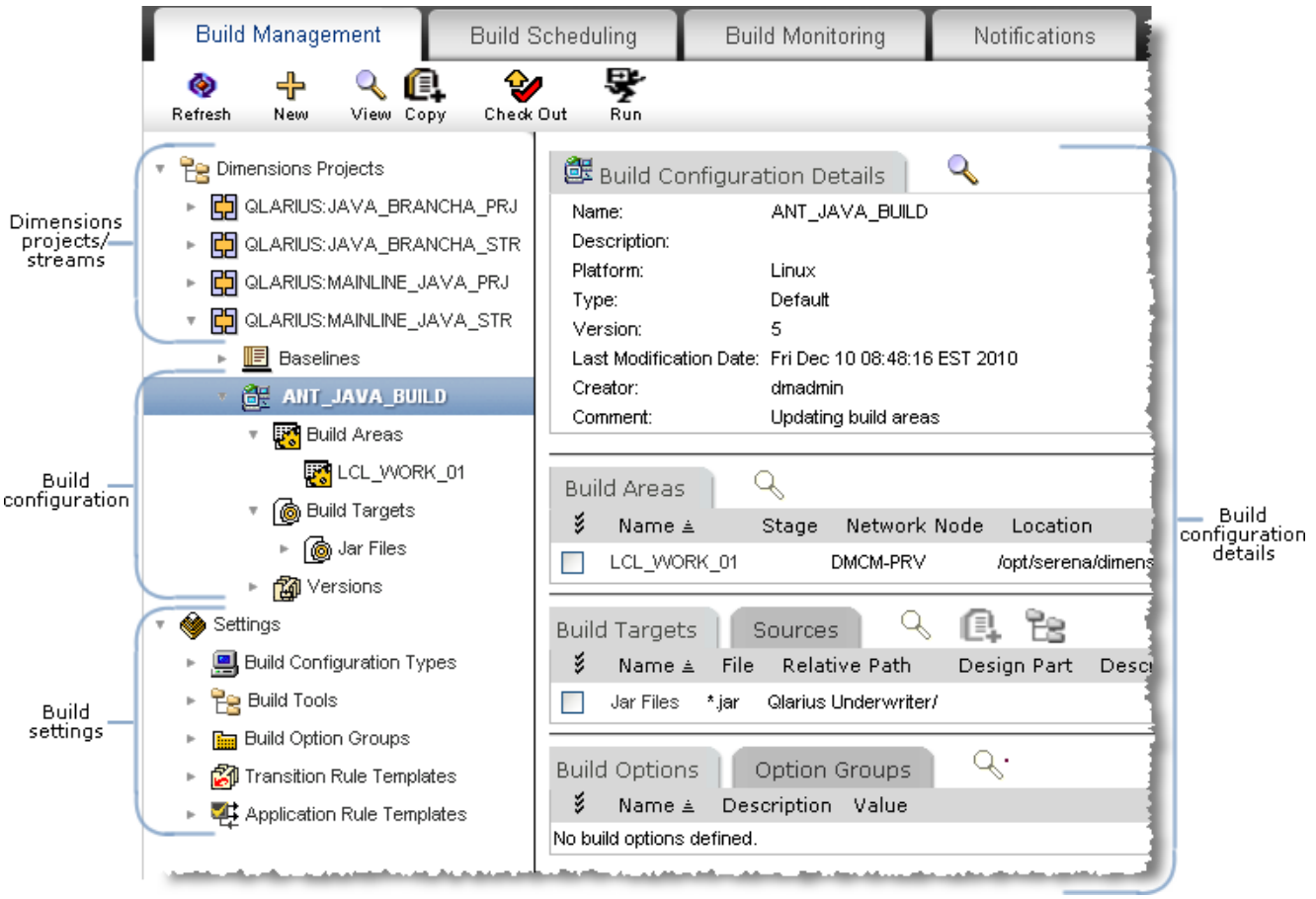
Each tab has the following areas:

- **Menu area:** Displays a toolbar to help you carry out various build tasks. The buttons that are displayed depend on the object that you select in the navigation or content panes.
- **Navigation area:** Allows you to view and select build objects.
- **Content pane:** Displays details about the build object that you have selected in the navigation pane.



Build Management Tab

The Build Management tab has the following areas:



Dimensions Projects

When you are working in the Dimensions Projects section of the Build Management tab, the menu area has the following buttons. The buttons that are displayed depend on the object that you select in the navigation or content panes.

Button	Description
	**Refresh**: refreshes the screen.
	**New**: creates one of the following new objects: ■ Build area. ■ Build target.
	**New**: creates a new build configuration.
	**Import**: imports a build configuration from an XML file.
	**Edit/View**: displays the selected build configuration and enables you to edit it.
	**Edit**: edits the object selected in the content pane.
	**Delete**: deletes the object selected in the content pane.
	**Filter**: (when you select the Dimensions Projects node in the navigation tree) filters the Dimensions projects that are displayed in the navigation and content panes.
	**Copy**: copies the selected build configuration or build target.
	**Checkin**: checks in the selected build configuration.
	**Checkout**: checks out the selected build configuration.
	**Undo Checkout**: undoes the check out of the selected build configuration.
	**Run**: builds target(s) in the selected build configuration.

Button	Description
	**View**: displays details of the selected checked in build configuration.
	**Rollback**: rolls back your current build configuration to the version you selected in the Versions node of the navigation pane.
	**Rewrite**: overwrites your current build configuration with the version you selected in the Versions node of the navigation pane.

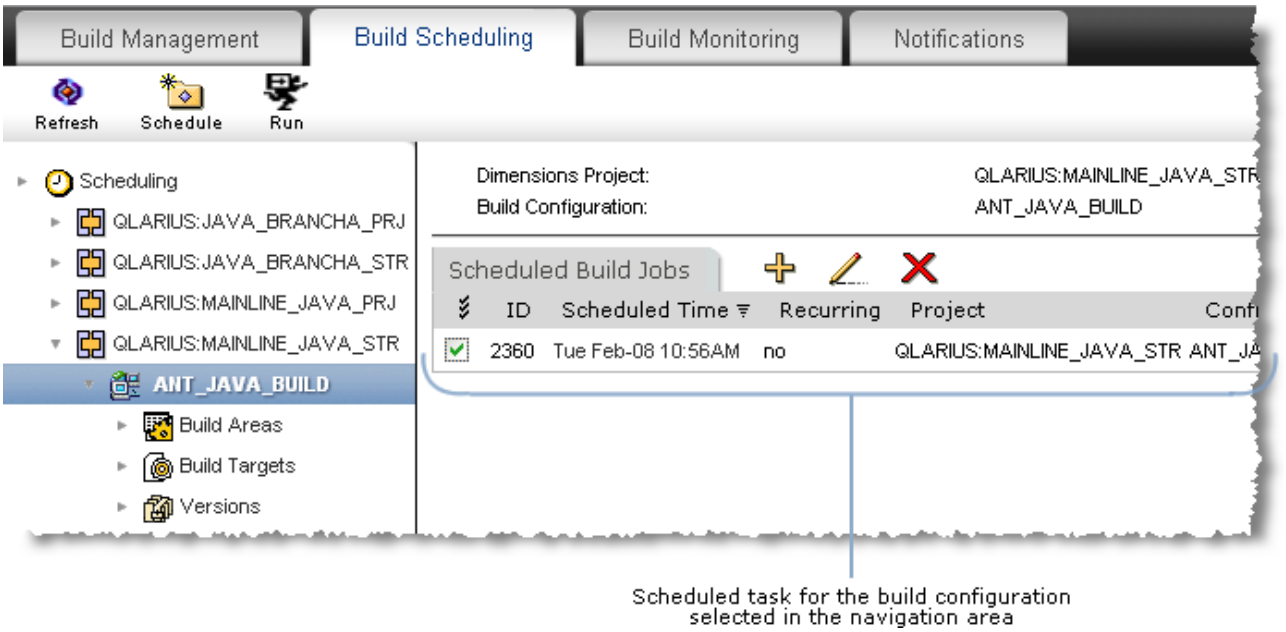
Settings

When you are working in the Settings section of the Build Management tab, the menu area has the following buttons. The buttons that are displayed depend on the object that you select in the navigation or content panes.

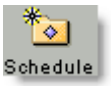
Button	Description
	**Refresh**: refreshes the screen.
	**Import**: (when you select the Build Tools node in the navigation tree) imports a build tool from an external file.
	**New**: creates one of the following new objects: ■ Build configuration type. ■ Build tool. ■ Build option group. ■ Transition rule template. ■ Application rule template.
	**Edit**: edits the object selected in the navigation or content panes.
	**Delete**: deletes the object selected in the content pane.

Build Scheduling Tab

The Build Scheduling tab has the following areas:

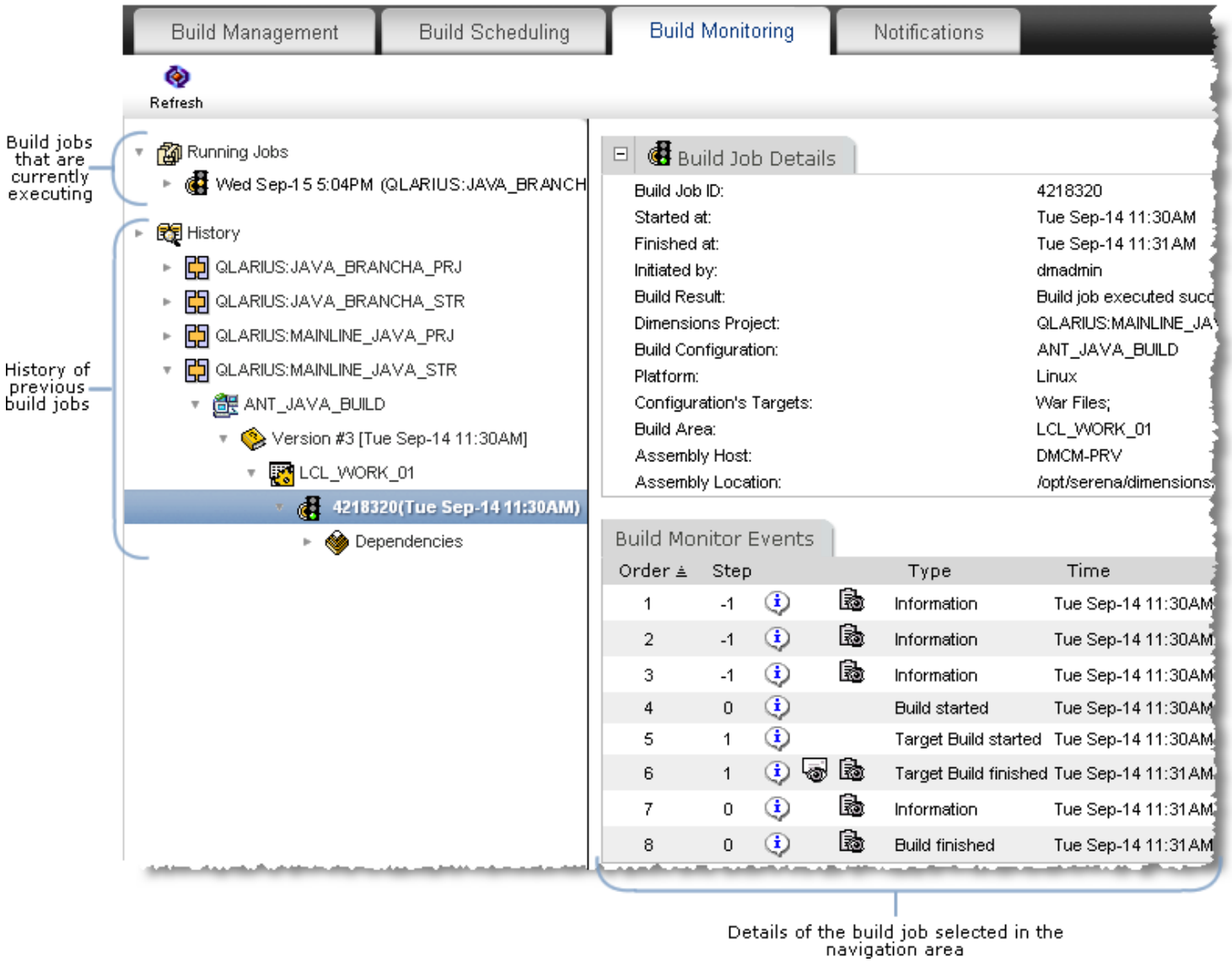


The menu area has the following buttons. The buttons that are displayed depend on the object that you select in the navigation or content panes

Button	Description
	**Refresh**: refreshes the screen.
	**Schedule**: creates a new build execution schedule.
	**Scheduler Service**: creates or modifies a scheduler service.
	**Filter**: (when you select the Scheduling node in the navigation tree) filters the Dimensions projects that are displayed in the navigation and content panes.
	**Run**: (when you select a build configuration in the navigation tree) enables you to build the selected build object.

Build Monitoring Tab

The Build Monitoring tab has the following areas:

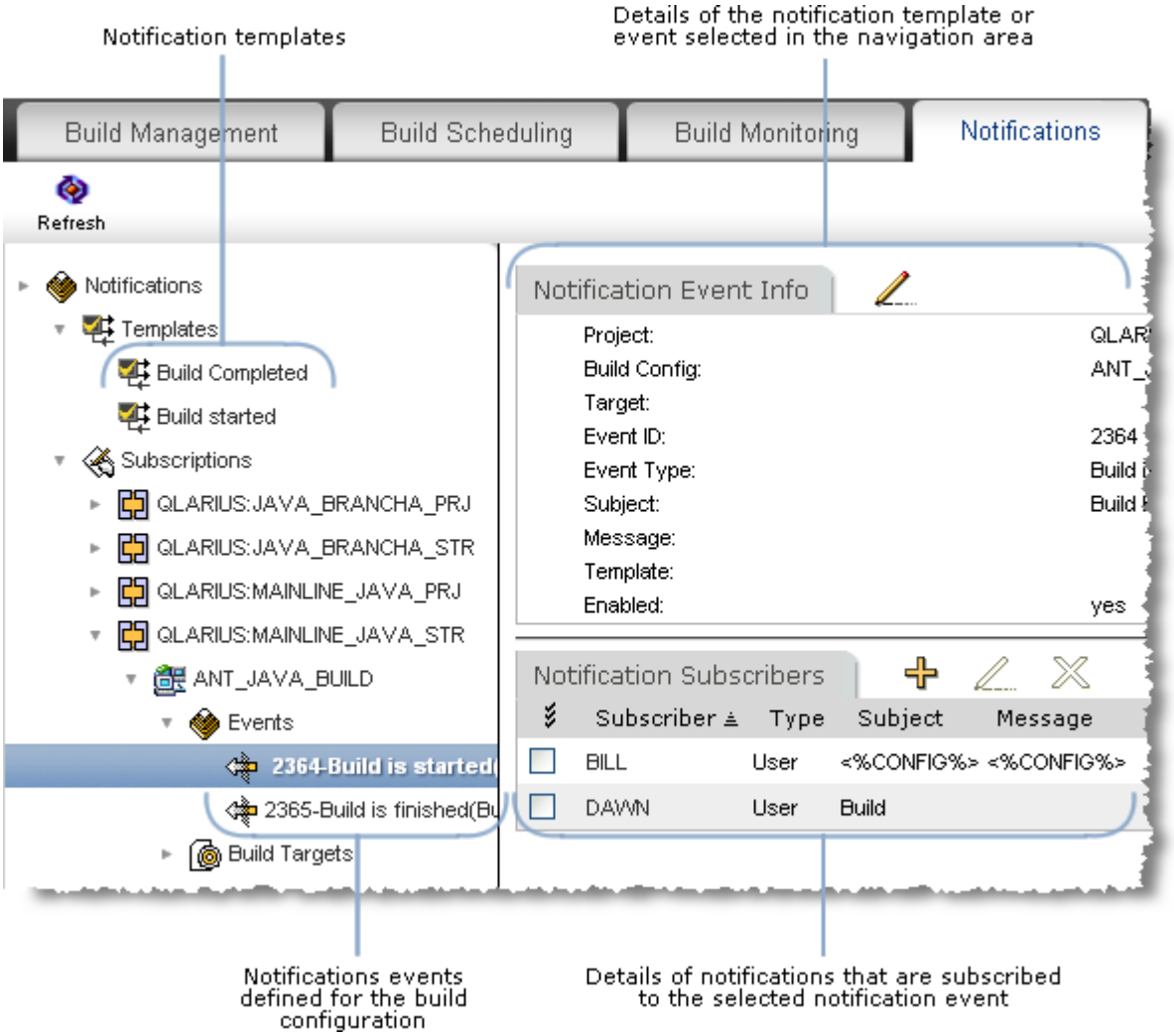


The menu area has the following buttons. The buttons that are displayed depend on the object that you select in the navigation or content panes.

Button	Description
	**Refresh**: refreshes the screen.
	**Cancel**: (for the Running Jobs node only) cancels the build job that is currently executing.
	**Rebuild**: rebuilds the build job that you select in the navigation or content panes.
	**Filter**: filters the build jobs that are displayed in the navigation and content panes.

Notification Tab

The Notification tab has the following areas:



The menu area has the following buttons. The buttons that are displayed depend on the object that you select in the navigation or content panes.

Button	Description
	**Refresh**: refreshes the screen.
	**Notification**: creates or modifies a notification service.
	**New**: creates a new template or notification event.

Button	Description
	**Copy**: copies the selected notification template.
	**Filter**: filters the build jobs that are displayed in the navigation and content panes.

Part 2

Configuring and Managing Dimensions Builds

Part 2: Configuring and Managing Dimensions Builds contains the following chapters

Configuring an MVS Build Environment	59
Managing Dimensions Build Settings	61
Managing Dimensions Build Configurations	95
Executing Builds in the Build Administration Console	133
Using Filters	137
Versioning	143
Scheduling Dimensions Build Jobs	147
Monitoring Builds in the Build Administration Console	157
Managing Notifications	167

Chapter 3

Configuring an MVS Build Environment

Pre-Requisites 60

Setting up your MVS Build Environment 60

Pre-Requisites

Use the Dimensions Administration Console to create definitions for the following remote z/OS mainframe network nodes:

- A physical network node for the z/OS mainframe where the Dimensions listener resides.
- A logical network node and connection for the USS file system.
- A logical network node and connection for the MVS file system.

For details see the *Dimensions for z/OS User's and Administrator's Guide*

Before setting up and running builds, check that you have access to all remote nodes where you will be running builds.

Setting up your MVS Build Environment

Setting up builds on z/OS mainframes can be a complicated and lengthy process. Serena recommends that you follow the steps below to set up your z/OS build environment:

1 Check that the following software is installed and configured:

- A Dimensions server on any supported platform. For details see the *Installation Guide for Windows* or the *Installation Guide for UNIX*.
- Dimensions for z/OS: check that you can run the Dimensions for z/OS ISPF client, or perform a remote checkout from the desktop client to the Dimensions node. For details see the *Dimensions for z/OS User's and Administrator's Guide*.

2 On your z/OS mainframe do the following:

- Pre-allocate the build areas that Dimensions will use to ensure that data sets are allocated properly (DCB, sizes, etc.) for your application and project. As a general rule, the last qualifier in the data set name indicates its type, for example, COBOL, COPY, H, and C. Build areas are typically allocated for each stage. Optionally, you can allocate a managed Development build area.

  NOTE Any managed build areas that you set up should be allocated under a security protected ID that the average Dimensions user does not have write access to. This precaution is necessary to protect the integrity of the data contained in the build areas.

- Create data sets for targets and intermediate targets, for example, .LOAD and .OBJ respectively.
- (Optional) Set up a user build area where you will do development work; allocate a collection of data sets with a common HLQ ('directory') as required for your build.

3 (Optional) If you are adding items from a PC to a z/OS mainframe and want automatic allocation of data sets, check that the Dimensions configuration file on the z/OS mainframe, MDH.V1010.MDHPARM(MDHTDCFG), has the correct defaults to create data sets by LLQ (low level qualifier). For details see the *Dimensions for z/OS User's and Administrator's Guide*. Serena recommends that you perform step 2 above to ensure proper data set sizing for each item type.

Chapter 4

Managing Dimensions Build Settings

About Build Settings	62
Build Configuration Types	62
Build Tools	66
Build Option Groups	72
Transition Rule Templates	78
Using Build Templates with Build Configurations	88
Application Rule Templates	91

About Build Settings

Dimensions Build enables you to configure the following build settings:

- **Build Configuration Types**

  Build configuration types specify options that are required for a particular type of build configuration. For more detail see the next section.

- **Build Tools**

  Build tools are logical groupings of build options for a given build engine. For more details see page 66.

- **Build Option Groups**

  Build option groups are logical collections of build tool options that are passed to the build engines and scripts that execute builds. For more details see page 72.

- **Transition Rule Templates**

  Transition rule templates describe generic rules for building items. For more details see page 78.

- **Application Rule Templates**

  Application rule templates define groups of transition rule templates that are used to create applications. For more detail see page 91.

Build Configuration Types

About Build Configuration Types

Build configuration types enable you to specify options that are required for a particular type of build configuration. For example, for the Openmake build configuration type you must specify the URL to the Openmake build monitor as follows:

Option: `DMOMSERVER` (Openmake server path)

Value: `http://localhost:58080`

TIP When you add a build configuration to a Dimensions project you have to specify a build configuration type. To specify a build configuration type that is not one of the Dimensions Build default types, you must add it first.

Dimensions Build includes the following build configuration types:

- Openmake: for Openmake builds.
- Default: for all other builds.

NOTE You cannot modify or delete the base properties of the default build configuration types, or add new options. However you can modify the default options. To add new options you can add build options to your build configuration, for details see "Build Options" on page 126.

About Build Configuration Type Options

Build configuration type options have the following properties:

- **Name**: the name of the option.
- **Description**: the description of the option.
- **Level**: must be one of the following:
  - **Build Configuration**: an option that is specific to the entire build configuration type, for example, Openmake project name or Openmake search path.
  - **Build Area**: an option that is specific to a build area, for example, a path to license file, a license manager host, or a Dimensions root directory.
  - **Build Transition**: an option that is specific to a script that builds targets, for example, debug or trace.
- **Default Value**: the default value of the option.
- **Required**: specifies if the option is mandatory.

About the Openmake Build Configuration Type

The Openmake build configuration type that is included with Dimensions Build has the following default options and values:

Option Name	Description	Level	Default Value	Required
DMBLDPROJ	Openmake project name	Build configuration	None	Yes
DMOMINSTALL	Openmake installation path	Build Area		Yes
DMOMSERVER	Openmake server path	Build Area	http://localhost:58080	Yes
DMPUBLIC	Public build job	Build configuration	True	Yes
DMSEARCHPATH	Openmake search path	Build configuration		Yes
DM_LICENSE	Path to the license file or license manager host.	Build area	@localhost	No
DM_OM_OSPLATFORM	Openmake platform	Build configuration	Java	Yes

The variables listed above are inputs to Openmake build templates. For details about the templating language and the Openmake build templates see the *Developer's Reference*.

Viewing Build Configuration Types

Purpose Follow this procedure to view a list of all the build configuration types currently defined.

To view build configuration types:

In the navigation pane of the Build Management tab click Settings and click Build Configuration Types. The content pane lists all the build configuration types that are currently defined.

Adding Build Configuration Types

Purpose Follow this procedure to add a new build configuration type.

To add a build configuration type:

1. In the navigation pane of the Build Management tab click Settings and select Build Configuration Types.
2. In the content pane click New Object.

   The Add New Build Configuration Type dialog box appears.
3. For **Name** type the name of the build configuration type.
4. For **Code** type a description of the build configuration type.
5. Click OK. The new build configuration type is added to the list of build configuration types in the content pane.

Adding Options to Build Configuration Types

Purpose Follow this procedure to add a new option to a build configuration type.

NOTE You cannot add new options to the default build configuration types.

To add an option to a build configuration type:

1. In the navigation pane of the Build Management tab click Settings, click Build Configuration Types, and select the build configuration type where you want to add an option. The content pane refreshes and displays details of the build configuration type.
2. In the Build Configuration Type Options section click New Object.

   The Add New Build Configuration Type dialog box appears.
3. For **Name** type the name of the build configuration type option.
4. For **Description** type a description of the build configuration type option.
5. From the **Level** list select the level that is appropriate to the option.
6. For **Default Value** type a default value for the option.
7. To make this option mandatory, select the **Required** check box.
8. Click OK. The new option is added to the list of build configuration type options in the content pane.

Modifying Build Configuration Types

Purpose Follow this procedure to modify an existing build configuration type.

NOTE You cannot modify the properties of the default build configuration types.

To modify a build configuration type:

1. In the navigation pane of the Build Management tab click Settings and click Build Configuration Types.
2. In the content pane select the build configuration type that you want to modify and click Edit Object.

   The Edit Build Configuration Type dialog box appears.
3. Modify the name if required.
4. Modify the Code (description) if required.
5. Click OK.

Modifying Options for Build Configuration Types

Purpose Follow this procedure to modify the existing options for a build configuration type.

To modify the options for a build configuration type:

1. In the navigation pane of the Build Management tab click Settings and click Build Configuration Types.
2. In the navigation or content panes select the build configuration type containing the option that you want to modify. The content pane refreshes and displays the options defined for the build configuration type.
3. Select the option that you want to modify and click Edit Selected Object.

   The Edit Build Configuration Type Option dialog box appears.
4. For **Name** modify the name of the build configuration type option.
5. For **Description** modify the description of the build configuration type option.
6. From the **Level** list select the level that is appropriate to the option.
7. For **Default Value** modify the default value for the option.
8. To make this option mandatory, select the **Required** check box.
9. Click OK.

Deleting Build Configuration Types

Purpose Follow this procedure to delete build configuration types.

NOTE You cannot delete the default build configuration types.

To delete build configuration types:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the content pane select the check box next to each build configuration type that you want to delete.
3. Click Delete. To confirm that you want to delete the build configuration types click Yes.

Deleting Options from Build Configuration Types

Purpose Follow this procedure to delete options from a build configuration type.

NOTE You cannot delete the default build configuration type options.

To delete options from a build configuration type:

1. In the navigation pane of the Build Management tab click Settings and click Build Configuration Types.
2. In the content or navigation panes select the build configuration type containing the options that you want to delete.
3. In the Build Configuration Type Options section of the content pane select the check box next to each option that you want to delete.
4. Click Delete. To confirm that you want to delete the options click Yes.

Build Tools

About Build Tools

Build tools are logical groupings of build options for a given build engine. For example, you can configure all the possible options for the Microsoft Visual C++ compiler and add a description for each option. When you set up a build configuration you can then choose from a meaningful list of possible build options for the compiler. This helps to avoid mistakes when manually entering options each time, and eliminates the need to check the meaning of option syntax in the compiler reference guides.

Build tools also enable you to enforce standards across all builds by managing the build options supported by the build tools used in build scripts.

You can import a build tool from an external XML files, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BuildTools author="Mickey Mouse" comments="yadda yadda" >
 <BuildTool name="C++ Compiler" description="C++ Compiler tool">
     <Option name="debug" description="blah blah">
         <Value description="option value description">-debug:1</Value>
         <Value description="option value description">-debug:2</Value>
         <Value description="option value description">-debug:3</Value>
         <Value description="option value description">-debug:4</Value>
     </Option>
     <Option name="#define" description="example of the option without
    values - freeform option value"/>
 </BuildTool>
 <BuildTool name="C++ Linker" description="C++ Linker tool">
     <Option name="debug" description="some option description here">
         <Value description="option value description">-debug:1</Value>
         <Value description="option value description">-debug:2</Value>
         <Value description="option value description">-debug:3</Value>
         <Value description="option value description">-debug:4</Value>
     </Option>
 </BuildTool>
</BuildTools>
```

About Build Options

Build options can be a single value or an array.

- Single value options appear in the BRD (Build Request Definition) file with the name and value for the attribute.
- Array value options appear in the BRD with the name and an array of values (except for the attribute).

For details of the templating language build options that you can add to build tools, see *Dimensions Build Predefined Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

Defining User Attributes in Build Options

If you define a build option value on a transition as `%option_name%`, Dimensions Build looks up the user attribute assigned to that transition input in Dimensions and passes the attribute value as the value for that option. You can define user attributes for single-value and multi-value attributes. Multi-line and block attributes are not supported.

Examples:

- A source item has a single-value user attribute named `"source_opts"` with a value of `"LIST,XREF,SOURCE"`. In the build options for a transition, you define an option called `"COMPILE_OPTS"` and set the value to `"%source_opts%"`. The BRD file will have an option called `"COMPILE_OPTS"` with a single value of `"LIST,XREF,SOURCE"` that you can reference in a build template.
- A source item has a multi-value user attribute named `"source_opts"` with the values `"LIST"`, `"XREF"`, and `"SOURCE"`. In the build options for a transition, you define an option called `"COMPILE_OPTS"` and set the value to `"%source_opts%"`. The BRD will have an option called `"COMPILE_OPTS"` with multiple values of `"LIST"`, `"XREF"`, and `"SOURCE"` that you can reference in a build template.

Build Tool Examples

- **IBM COBOL compiler v1.4**

Build Option	Description	Possible Values and Descriptions
INTDATE	Specifies the starting date for integer dates used with date intrinsic functions.	INTDATE(ANSI): Uses the Standard COBOL 85 starting date. Day 1 is Jan 1, 1601. INTDATE(LILIAN): Uses the Language Environment Lilian starting date. Day 1 is Oct 15, 1582
MAP	Produces a listing of the items defined in the DATA DIVISION.	MAP: produces a listing. NOMAP: does not produce a listing.

- **Microsoft C++ Compiler**

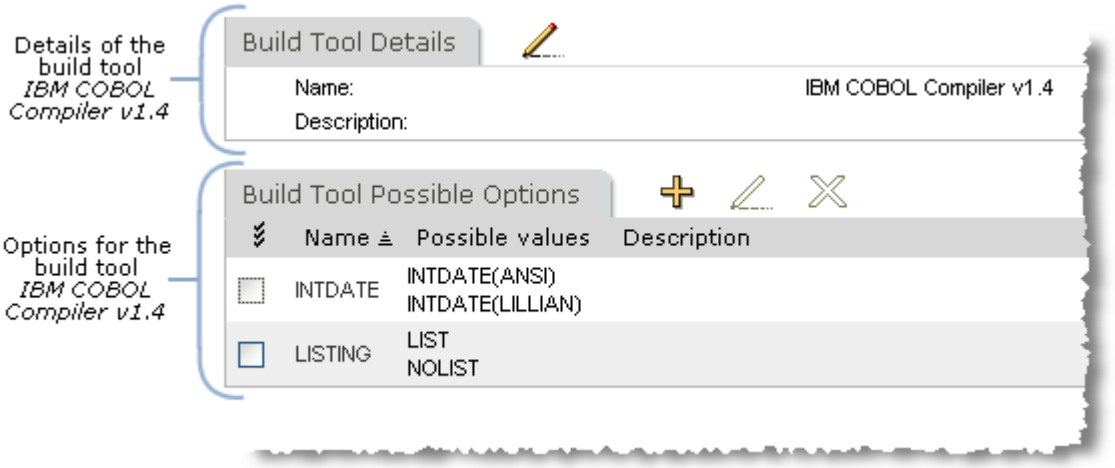
Build Option	Description	Possible Value
INCLUDE	Specifies an INCLUDE directory.	C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include
LINK	Passes one or more linker options to the linker.	/COMMENT:"For Payroll Project" / DEBUG

Viewing Build Tools and Options

Purpose Follow this procedure to view a list of build tools and their options.

To view build tools and options:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools. The content pane lists all the build tools that are currently defined.
2. To view a build tool's options, in the navigation or content pane click the build tool. The content pane refreshes and lists all the options defined for the build tool.



Adding Build Tools

Purpose Follow this procedure to add a new build tool.

To add a build tool:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the content pane, on the Build Tools tab, click New Object.

   The Add New Build Tool dialog box appears.
3. For **Name** type the name of the build tool.
4. For **Description** optionally type a description of the build tool.
5. Click OK. The new build tool is added to the list of build tools in the content pane.

Adding Options to Build Tools

Purpose Follow this procedure to add a new option to a build tool.

To add an option to a build tool:

1. In the navigation pane of the Build Management tab click Settings and select Build Tools.
2. In the content or navigation panes click the build tool where you want to add a build option. The content pane refreshes and displays the details of the build tool.
3. On the Build Tool Possible Options tab click New Object.

   The Add New Build Tool Option dialog box appears.
4. For **Option Name** type a name for the option.
5. For **Option Description** type a description of the option.
6. For **Possible Values** type a value for the option. Press the Tab key.
7. For **Descriptions of Values** optionally type a description of the value.
8. To add additional values to this option repeat steps 6 and 7.
9. Click OK. The build options and values are added to the Build Tool Possible Options tab in the content pane.

Importing Build Tools

Purpose Follow this procedure to import a build tool from an external XML file.

To import a build tool:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the content pane click Import Build Tools.

   The Import Build Tools from External File dialog box appears.
3. Do one of the following:
   - Type the path to the file.
   - Click Browse, navigate to the file, select it, and click Open.
4. Click OK. The build tool is imported into Dimensions Build and added to the list in the content pane.

Modifying Build Tools

Purpose Follow this procedure to modify the details of an existing build tool.

To modify a build tool:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the navigation or content panes select the build tool that you want to modify. The content pane refreshes and displays the details of the build tool.
3. On the Build Tools Details tab click Edit Build Tool Details.

   The Edit Build Tool dialog box appears.
4. Modify the build tool name if required.
5. Modify the build tool description if required.
6. Click OK. The details of the build tool are updated in the content pane.

Modifying Build Tool Options

Purpose Follow this procedure to modify the options for an existing build tool.

To modify build tool options:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the navigation or content panes select the build tool containing the options that you want to modify. The content pane refreshes and displays the details of the build tool.
3. In the Build Tool Possible Options section of the content pane, click the build option that you want to modify. The Edit Build Tool Option dialog box appears.
4. Modify the option name if required.
5. Modify the option description if required.
6. Modify the option values and their descriptions if required. To add a new option, in the **Possible Values** field type a value for the option. In the **Descriptions of Values** field optionally type a description of the new value.
7. Click OK.

Deleting Build Tools

Purpose Follow this procedure to delete build tools. When you delete build tools their build options are also deleted.

To delete build tools:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the content pane select the check box next to the build tools that you want to delete.
3. Click Delete. To confirm that you want to delete the build options click Yes.

Deleting Build Tool Options

Purpose Follow this procedure to delete build options from a build tool. This procedure does not delete the parent build tool.

To delete build tool options:

1. In the navigation pane of the Build Management tab click Settings and click Build Tools.
2. In the navigation or content panes select the build tool that contains the build options that you want to delete.
3. On the Build Tool Possible Options tab select the check boxes of all the build options that you want to delete.
4. Click Delete. To confirm that you want to delete the build tool options click Yes.

Build Option Groups

About Build Option Groups

Build option groups are logical collections of build tool options that are passed to the build engines and scripts that execute builds, and enable you to apply rules and enforce standards. Examples of option groups are Compile_Options and Link_Options.

- The names of build option groups do not appear in the BRD file, only the options and values that are in the groups.
- You can modify build option groups without checking out the build configuration in which they are used. The revised options and values will appear in the BRD.

About Build Option Properties

For each build option group you specify a preferred build tool and build options. Build options have the following properties:

- **Name**: the name of the option. Multiple build options can share the same name.
- **Value Type**: is one of the following:
  - Option Rule: enables you to select a build tool, and then specify options and values for that build tool.
  - Freeform Text: enables you to specify an option that is tailored to your requirements.
- (For Option Rule only) **Build Tool**: specifies the preferred build tool (does not have to be the preferred build tool specified for the build option group).
- (For Option Rule only) **Option Rule**: specifies an option rule for the build tool you select.
- **Value**:
  - If you selected Option Rule, specifies a value for the option rule.
  - If you selected Freeform text, specifies the value of the build option.

Build Option Group Examples

- **Common CICS COBOL Options**

  Preferred build tool: IBM CICS Preprocessor V2R3

  Build options:

Build Option	Value Type	Build Tool	Option Rule	Value
COBOL_OPTS	Option Rule	IBM COBOL Compiler v1.4	SOURCE	S
CICS_OPTS	Option Rule	IBM CICS Preprocessor V2R3	APOST	QUOTE

- **Common COBOL Options**

  Preferred build tool: IBM COBOL Compiler v1.4

  Build options:

Build Option	Value Type	Build Tool	Option Rule	Value
COBOL_OPTS	Option Rule	IBM COBOL Compiler v1.4	SOURCE	S
COBOL_OPTS	Option Rule	IBM COBOL Compiler v1.4	INTDATE	INTDATE (LILLIAN)

About Build Options

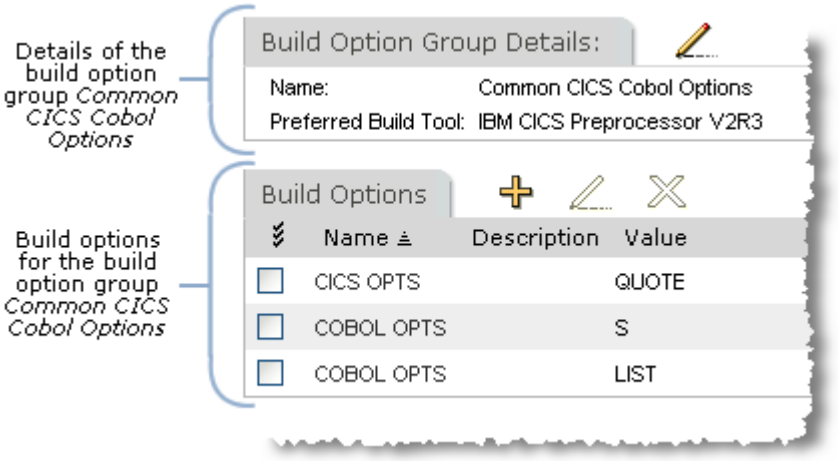
For details of the build options that you can add to build option groups, see *Dimensions Build Predefined Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

Viewing Build Option Groups

Purpose Follow this procedure to view a list of build option groups and their options.

To view build option groups and options:

1. In the navigation pane of the Build Management tab click Settings and click Build Option Groups. The content pane lists all the build option groups that are currently defined.
2. To view a build option group's build options, in the **Name** column in the content pane click the build option group name. The content pane refreshes and lists all the build options defined for the build option group.



Adding Build Option Groups

Purpose Follow this procedure to add a new build option group.

To add a build option group:

1. In the navigation pane of the Build Management tab click Settings and click Build Option Groups.
2. In the content or menu areas click New Object.

   The Add New Build Option Group dialog box appears.
3. From the **Option Group Build Tool** list select a preferred build tool for the option group.

4 For **Option Group Name** type a name for the option group. Use meaningful names that reflect the context and usage of the option group, for example, *Common CICS COBOL Options* and *VC++ Compilers*.

5 Click OK. The new option group is added to the list of build option groups in the content pane.

Adding Build Options to Build Option Groups

Purpose Follow this procedure to add a new build option to an existing build option group.

To add a build option to a build option group:

1 In the navigation pane of the Build Management tab click Settings and click Build Option Groups.

2 In the navigation or content pane click the build option group where you want to add a build option. The content pane refreshes and displays the details of the build option group.

3 In the Build Options section of the content pane click New Object.

The Add New Build Option Group Option dialog box appears.

4 For **Name** type a name for the option.

5 From the **Value Type** list select one of the following:

- Option Rule: enables you to select a build tool, and then specify options and values.
- Freeform Text: enables you to specify an option that is tailored to your specific requirements.

6 If you selected Option Rule from the Value Type list, do the following:

a From the **Build Tool** list select a build tool.

b From the **Option Rule** list select an option rule for the build tool. The list displays the possible option rules for the build tool that you selected from the Build Tool list.

c From the **Value** list select a value for the option rule. The list displays the possible values for the option rule that you selected from the Option Rule list. If the option rule does not have a default value, type your own.

7 If you selected Freeform Text from the Value Type list, in the **Value** field type the content of the build option.

8 Click OK. The new option is added to the list of build options for that build option group.

Modifying Build Option Groups

Purpose Follow this procedure to modify the details of an existing build option group.

To modify the details of build option groups:

1. In the navigation pane of the Build Management tab click Settings and click Build Option Groups.
2. In the navigation or content panes select the build option group that you want to modify. The content pane refreshes and displays the details of the option group.
3. In the Option Group Details section of the content pane click Edit Option Group Details.

   The Edit Option Group Details dialog box appears.
4. From the **Option Group Build Tool** list select a preferred build tool for the option group.
5. For **Option Group Name** modify the name of the option group. Use meaningful names that reflect the context and usage of the option group, for example, *Common CICS COBOL Options* and *VC++ Compilers*.
6. Click OK.

Modifying the Options of Build Option Groups

Purpose Follow this procedure to modify the build options of an existing build option group.

To modify the options of a build option group:

1. In the navigation pane of the Build Management tab click Settings and click Build Option Groups.
2. In the navigation or content panes select the build option group containing the build options that you want to modify. The content pane refreshes and displays the details of the build option group.
3. In the Build Options section of the content pane click the build option that you want to modify. The Edit Build Option Group Option dialog box appears.
4. For **Name** modify the name of the option.
5. From the **Value Type** list select one of the following:
   - Option Rule: enables you to select a different build tool, and then specify options and values.
   - Freeform Text: enables you to specify an option that is tailored to your requirements.

6 If you selected Option Rule, do the following:

   a From the **Build Tool** list select a different build tool.

   b From the **Option Rule** list select an option rule for the build tool. The list displays the possible option rules for the build tool that you selected from the Build Tool list.

   c From the **Value** list select a value for the option rule. The list displays the possible values for the option rule that you selected from the Option Rule list. If the option rule does not have a default value, type your own.

7 If you selected Freeform Text, in the **Value** field modify the existing option or type the content of a new value.

8 Click OK.

Deleting Build Option Groups

Purpose Follow this procedure to delete build option groups and their associated build options.

To delete build option groups:

1 In the navigation pane of the Build Management tab click Settings and click Build Option Groups.

2 In the content pane select the check boxes next to the build option groups that you want to delete.

3 In the content or menu areas click Delete. To confirm that you want to delete the build option groups click Yes.

Deleting Options from Build Option Groups

Purpose Follow this procedure to delete build options from a build option group.

To delete options from a build option group:

1 In the navigation pane of the Build Management tab click Settings and click Build Option Groups.

2 In the navigation or content panes click the build option group containing the build options that you want to delete.

3 In the Build Options section of the content pane select the check boxes next to the builds option that you want to delete.

4 In the content or menu areas click Delete. To confirm that you want to delete the build options click Yes.

Transition Rule Templates

About Transition Rule Templates

A transition rule template describes a generic rule for building an item and applying it to specific source items at a later time. It describes how to change or transition an item of one file type (i.e., *.c) to an item of another file type (i.e., *obj) using a template or script.

A transition rule template is comprised of:

- The generic input mask(s) to the transition.
- The build tool used to perform the transition.
- The template or script used to execute the transition.
- The build options and option groups used during the transition (compile, link, etc.).
- The generic output mask of the transition.

NOTE Transition rule templates are not the same as build templates. For details about build templates and the templating language see the *Developer's Reference*.

Transition Rule Template Examples

- **ASM Compile**
  - Preferred build tool: IBM High Level Assembler
  - Script/template: TEMPLATE(MDHBASC0)
  - Input mask: ASM(*)
  - Output mask: OBJ(*)
  - Build options: DMMAXRC(4)
- **General Mainframe Program Link**
  - Preferred build tool: IBM Binder v1.4
  - Script/template: TEMPLATE(LINKLE)
  - Input masks: OBJ(*), SYSLIN(*)
  - Output mask: LOAD(*)
  - Build options: LINK_OPTS AMODE(31), LINK_OPTS CASE(MIXED)

About Build Options

For details of the build options that you can add to transition rule templates, see *Dimensions Build Predefined Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

Viewing Transition Rule Templates

Purpose Follow this procedure to view a list of transition rule templates and their properties.

To view transition rule templates:

1 In the navigation pane of the Build Management tab click Settings and click Build Tools. The navigation and content panes list all the transition rule templates that are currently defined.

2 To view the properties of a transition rule template, in the navigation or content panes click the transition rule template. The content pane refreshes and displays the details of the transition rule template in the following tabs:

- Inputs
- Outputs
- Build Options
- Option Groups

To view the details of a specific property click the appropriate tab.

Adding Transition Rule Templates

Purpose Follow this procedure to add a new transition rule template.

To add a transition rule template:

1 In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.

2 In the content or menu areas click New Object.

The Add New Transition Rule Template dialog box appears.

3 For **Name** type the name of the new transition rule template.

4 From the **Preferred Build Tool** list select the build tool that will be used to execute the transition. If you do not want to specify a build tool, select Undefined.

5 From the **Select the script type** list select one of the following:

- *Input script content manually*

  In the **Script Content** field type the content of the script to be executed when this transition rule is run. To wrap the text select the **Word Wrap** check box.

- *Use Dimensions-controlled file as a script*

  Choose a file from the Dimensions repository (see step 6 below).

  NOTE If you are using a work area the file will be pushed to the build area. If you are using a deployment area, Dimensions Build assumes that the script has already been deployed to that area.

- *Use a file in the build area as a script*

  Type the name of a script or template located in your build area.

- *Same as build configuration global script*

When you use a transition that has this script type the target is built using the main script defined for the parent build configuration. However, you can specify build options for the transition that are different to those defined in the main script. Therefore the build options that are used are those defined in the transition and not the main script.

For details about adding build options to transition rule templates see page 84.

6 If you selected *Use Dimensions-controlled file as a script*, do the following:

a Click Browse. The Select File from Dimensions dialog box appears.

b In the left pane, expand Projects or Baselines and browse for the directory containing the file.

c In the right pane select the file.

a Click OK. The file is added to the Add New Transition Rule Template dialog box.

7 Click OK. The new transition rule template is added to the list of transition rule templates in the content pane.

Modifying Transition Rule Templates

Purpose Follow this procedure to modify the details of an existing transition rule template.

To modify a transition rule template:

1 In the navigation pane of the Build Management tab click Settings and click Transition Rule Templates.

2 In the content pane select the transition rule template that you want to modify, and click Edit.

The Edit Transition Rule Template dialog box appears.

3 For **Name** modify the name of the transition rule template.

4 From the **Preferred Build Tool** list select the build tool used to execute the transition. If you do not want to specify a build tool, select Undefined.

5 From the **Select the script type** list select one of the following:

- *Input script content manually*

  In the **Script Content** field type or modify the content of the script to be executed when this transition rule is run. To wrap the text select the **Word Wrap** check box.

- *Use Dimensions-controlled file as a script*

  Choose a file from the Dimensions repository (see step 6 below).

  NOTE If you are using a work area the file will be pushed to the build area. If you are using a deployment area, Dimensions Build assumes that the script has already been deployed to that area.

- *Use a file in the build area as a script*

  Type or modify the name of a script or template in your build area.

- *Same as build configuration global script*

  When you use a transition that has this script type the target is built using the main script defined for the parent build configuration. However, you can specify build options for the transition that are different to those defined in the main script. Therefore the build options that are used are those defined in the transition and not the main script.

  For details about adding build options to transition rule templates see page 84.

6 If you selected *Use Dimensions-controlled file as a script*, do the following:

a Click Browse. The Select File from Dimensions dialog box appears.

b In the left pane, expand Projects or Baselines and browse for the directory containing the file.

c In the right pane select the file.

a Click OK. The file is added to the Add New Transition Rule Template dialog box.

7 Click OK. The details of the transition rule template are updated in the content pane.

Deleting Transition Rule Templates

Purpose Follow this procedure to delete transition rule templates.

To delete transition rule templates:

1 In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.

2 In the content pane select the check box next to each transition rule template that you want to delete.

3 In the content or menu areas click Delete.

4 To confirm that you want to delete the transition rule templates click Yes.

Adding Input Masks to Transition Rule Templates

Purpose Follow this procedure to add a new generic input mask (file type) to a transition rule template, for example, ASM(*).

To add an input mask to a transition rule template:

1 In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.

2 In the navigation or content panes select the transition rule template to which you want to add a new input mask. The content pane refreshes and displays the details of the transition rule template.

3 In the content pane select the Inputs tab and click New Object.

The Add New Transition Rule Template Input Mask dialog box appears.

4 For **Mask Value** type a value for a new generic file type.

5 Click OK. The new mask value is added to the list on the Inputs tab.

Modifying Transition Rule Template Input Masks

Purpose Follow this procedure to modify the value of an existing generic input mask for a transition rule template.

To modify a transition rule template input mask:

1 In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.

2 In the navigation or content panes select the transition rule template containing the input mask that you want to modify. The content pane refreshes and displays the details of the transition rule template.

3 In the content pane select the Inputs tab, select the input mask, and click Edit.

The Edit Transition Rule Template Input Mask dialog box appears.

4 For **Mask Value** modify the value of the generic file type.

5 Click OK. The modified mask value is updated in the list on the Inputs tab.

Deleting Input Masks from Transition Rule Templates

Purpose Follow this procedure to delete input masks from a transition rule template.

To delete input masks from a transition rule template:

1 In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.

2 In the navigation or content panes click the transition rule template containing the input masks that you want to delete.

3 In the content pane, click the Inputs tab.

4 Select the check box next to each input mask that you want to delete.

5 Click Delete.

6 To confirm that you want to delete the input masks click Yes.

Adding Output Masks to Transition Rule Templates

Purpose Follow this procedure to add a new generic output mask (file type) to a transition rule template, for example, OBJ(*).

To add an output mask to a transition rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.
2. In the navigation or content panes select the transition rule template to which you want to add a new input mask. The content pane refreshes and displays the details of the transition rule template.
3. In the content pane select the Outputs tab and click New Object.

   The Add New Transition Rule Template Output Mask dialog box appears.
4. For **Mask Value** type a value for a new generic file type.
5. Click OK. The new mask value is added to the list on the Outputs tab.

Modifying Transition Rule Template Output Masks

Purpose Follow this procedure to modify the value of an existing generic output mask for a transition rule template.

To modify a transition rule template output mask:

1. In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.
2. In the navigation or content panes select the transition rule template containing the output mask that you want to modify. The content pane refreshes and displays the details of the transition rule template.
3. In the content pane click the Outputs tab, select the output mask, and click Edit.

   The Edit Transition Rule Template Output Mask dialog box appears.
4. For **Mask Value** modify the value of the generic file type.
5. Click OK. The modified mask value is updated in the list on the Outputs tab.

Deleting Output Masks from Transition Rule Templates

Purpose Follow this procedure to delete output masks from a transition rule template.

To delete output masks from a transition rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.
2. In the navigation or content panes click the transition rule template containing the output masks that you want to delete.
3. In the content pane, click the Outputs tab.

4 Select the check box next to each output mask that you want to delete.

5 Click Delete.

6 To confirm that you want to delete the output masks click Yes.

Adding Build Options to Transition Rule Templates

Purpose Follow this procedure to add a new build option to a transition rule template.

To add a build option to a transition rule template:

1 In the navigation pane of the Build Management tab click Settings and click Transition Rule Template.

2 In the navigation or content pane click the transition rule template where you want to add a build option. The content pane refreshes and displays the details of the transition rule template.

3 In the Build Options section of the content pane click New Object.

The Add New Transition Rule Template Option dialog box appears.

4 For **Name** type a name for the option.

5 From the **Value Type** list select one of the following:

- *Freeform Text*

  In the **Value** field type an option that is tailored to your specific requirements.

- *Option Rule*

  Select a build tool and then select an option rule and a value (see step 6).

- *Option Link*

  Select a build option group and then select an option (see step 7).

6 If you selected Option Rule from the Value Type list, do the following:

a From the **Build Tool** list select a build tool. If you do not want to specify a build tool select *Undefined*.

b (Not applicable if you selected *Undefined)* From the **Option Rule** list select an option rule for the build tool. The list displays the possible option rules for the build tool that you selected from the Build Tool list.

c (Not applicable if you selected *Undefined)* From the **Value** list select a value for the option rule. The list displays the possible values for the option rule that you selected from the Option Rule list. If the option rule does not have a default value, type your own.

7 If you selected Option Link from the Value Type list, do the following:

a From the **Group** list select a build option group.

b From the **Option** list select an option for the build option group. The list displays the possible options for the build option group that you selected from the Group list.

8 Click OK. The new build option is added to the list of build options for the transition rule template.

Modifying Build Options for Transition Rule Templates

Purpose Follow this procedure to modify a build option for a transition rule template.

To modify a build option for a transition rule template:

1 In the navigation pane of the Build Management tab click Settings and click Transition Rule Template.

2 In the navigation or content pane click the transition rule template containing the build option that you want to modify. The content pane refreshes and displays the details of the transition rule template.

3 In the Build Options section of the content pane click the build option that you want to modify. The Edit Transition Rule Template Option dialog box appears.

4 For **Name** modify the name of the option.

5 From the **Value Type** list select one of the following:

- *Freeform Text*

  In the **Value** field type an option that is tailored to your specific requirements, or modify the existing option.

- *Option Rule*

  Select a different build tool and then select an option rule and a value (see step 6).

- *Option Link*

  Select a different build option group and then select an option (see step 7).

6 If you selected Option Rule from the Value Type list, do the following:

a From the **Build Tool** list select a different build tool. If you do not want to specify a build tool select *Undefined*.

b (Not applicable if you selected *Undefined)* From the **Option Rule** list select an option rule for the build tool. The list displays the possible option rules for the build tool that you selected from the Build Tool list.

c (Not applicable if you selected *Undefined)* From the **Value** list select a value for the option rule. The list displays the possible values for the option rule that you selected from the Option Rule list. If the option rule does not have a default value, type your own.

7 If you selected Option Link from the Value Type list, do the following:

a From the **Group** list select a different build option group.

b From the **Option** list select an option for the build option group. The list displays the possible options for the build option group that you selected from the Group list.

8 Click OK. The modified build option is updated in the list of build options for the transition rule template.

Deleting Build Options from Transition Rule Templates

Purpose Follow this procedure to delete build options from a transition rule template.

To delete build options from a transition rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.
2. In the navigation or content panes click the transition rule template containing the build options that you want to delete.
3. In the content pane, click the Build Options tab.
4. Select the check box next to each build option that you want to delete.
5. Click Delete.
6. To confirm that you want to delete the build options click Yes.

Adding Build Option Groups to Transition Rule Templates

Purpose Follow this procedure to add a build option group to a transition rule template. You can only add build option groups that are already defined. For details about adding new build option groups see page 74.

To add a build option group to a transition rule template:

1. In the navigation pane of the Build Management tab click Settings and click Transition Rule Template.
2. In the navigation or content pane click the transition rule template where you want to add a build option group. The content pane refreshes and displays the details of the transition rule template.
3. In the content pane select the Option Groups tab and click New Object.

   The Select Transition Rule Template Build Option Group dialog box appears.
4. Select a build option group from the list.
5. Click OK. The build option group is added to the list of build option groups for the transition rule template.

Deleting Build Option Groups from Transition Rule Templates

Purpose Follow this procedure to delete build option groups from a transition rule template.

To delete build option groups from a transition rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Transition Rule Templates.
2. In the navigation or content panes click the transition rule template containing the build options that you want to delete.
3. In the content pane, click the Option Groups tab.
4. Select the check box next to each build option group that you want to delete.
5. Click Delete.
6. To confirm that you want to delete the build option groups click Yes.

Using Build Templates with Build Configurations

This section describes how to use build templates with build configurations.

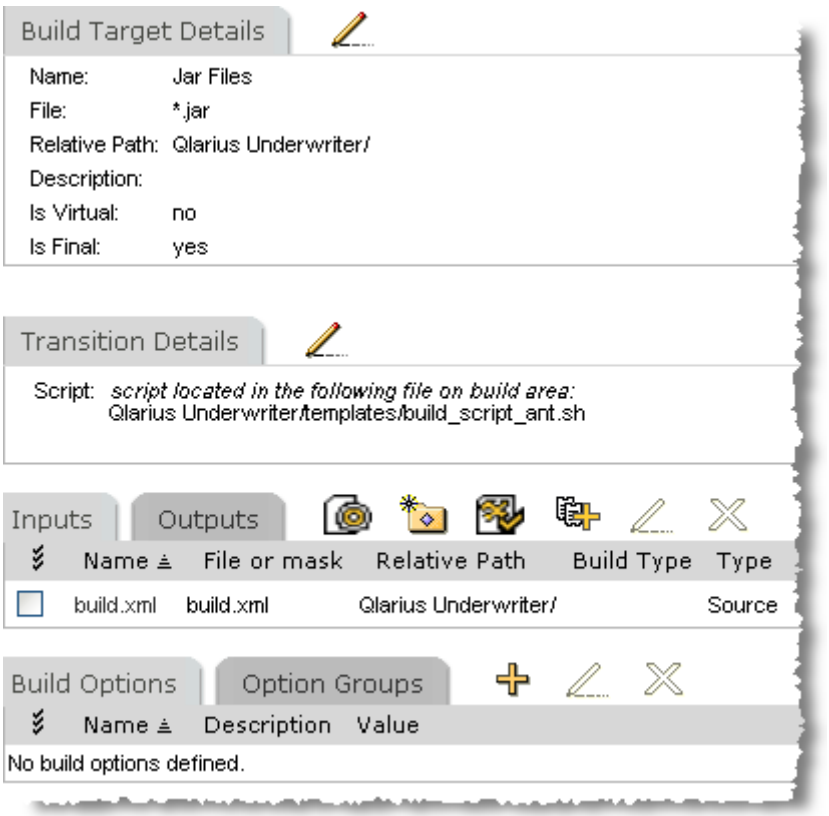
Specifying Build Templates

To specify the build template that executes during a build step, do the following:

1. In the navigation pane, select a build configuration in a project or stream and check it out.
2. In the content pane, on the Build Targets tab, click New.

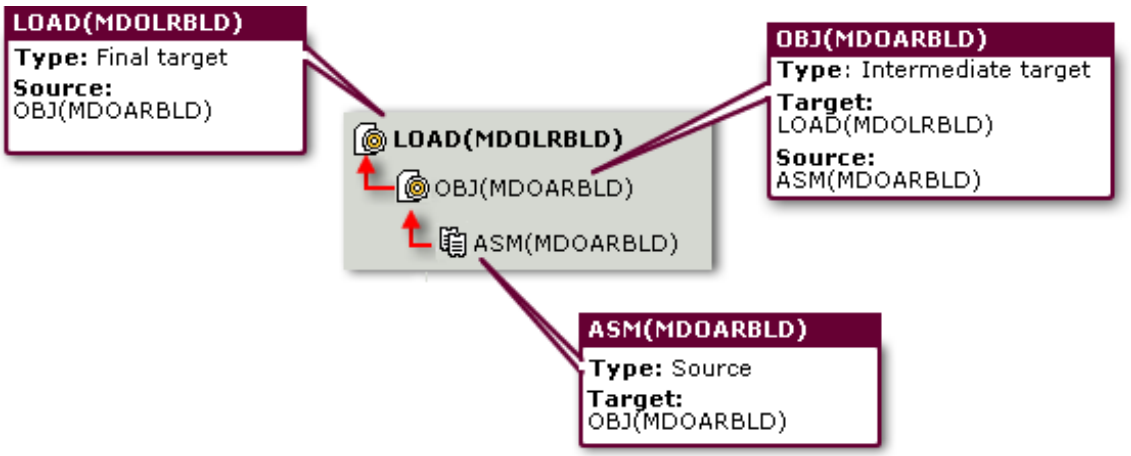
   The Create New Target Wizard appears.
3. On the Target page define the target and click Next.
4. On the Scripts page, from the Select Source list select **Use a file in the build area as a script**.
5. For a Dimensions CM controlled build, to locate the template in a search path, click Browse and navigate to the file, or type the full path and name of the template.
6. Complete the rest of the wizard.
7. In the navigation pane expand the Build Targets node for the build configuration and select the target.

   The content pane displays the details of the build target including the build template, the input, and the target:



Specifying a Chain of Build Steps

You can specify a chain of build steps that creates a final target by adding the target from one step as the input to another step. The diagram below illustrates a simple chain of build steps with a single output, a single intermediate target, and a single source. The source item is the input to the intermediate target and the intermediate target is the input to the final target.

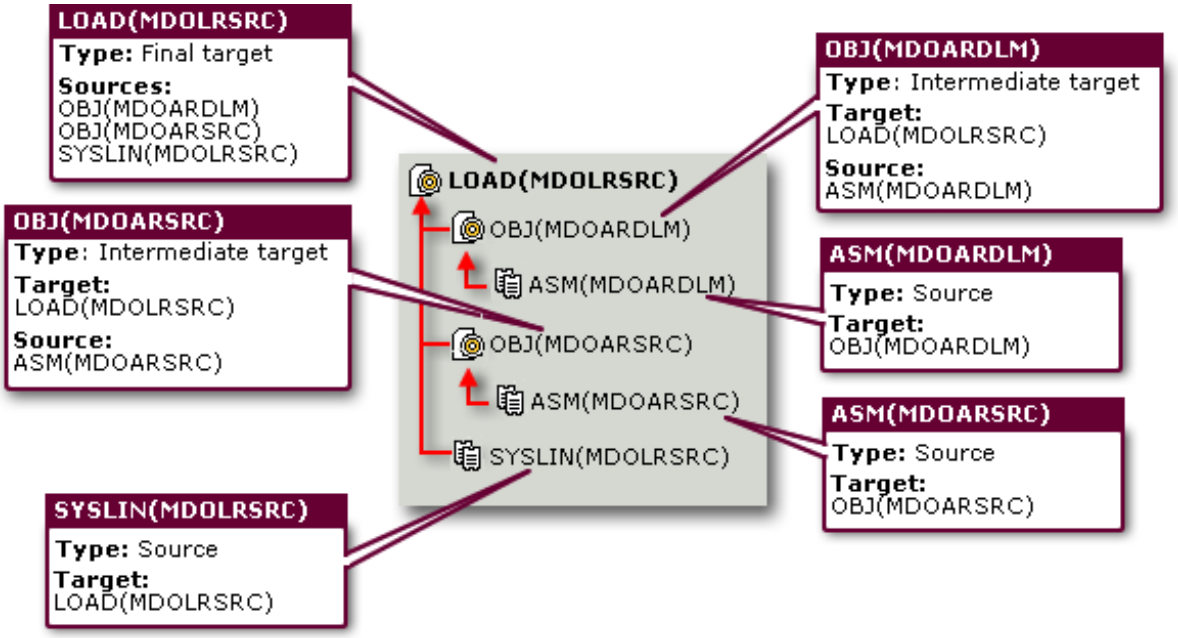


To specify a chain of build steps:

1. Open the Create New Target Wizard. On the Target page define the target and select the **Is Final** check box. Populate the Script and Options pages as required.
2. On the Build Plan page select an action, for example **Add new target** or **Add new source**. Click Next and follow the instructions in the wizard.
3. When the Build Plan page appears select the build object where you want to add an action to the chain and repeat step 2.
4. Repeat steps 2 and 3 until you have completed your chain of build steps.
5. Click Finish. The navigation and content panes update and display your build chain.

Specifying Multiple Inputs

When you create a build target for a build configuration you can also define multiple inputs to a final target. The procedure is the same as the one described in the previous section. The example below shows a final target built from three source modules:



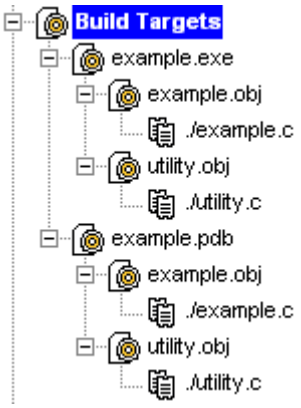
Specifying Multiple Outputs

When you create a build target for a build configuration you can define multiple outputs using the same inputs. During the build only one step is executed and only one template is expanded.

Do the following:

1. Create the first output as normal using the Create New Target Wizard.
2. To create the next output open the Create New Target Wizard.
3. On the Targets page, from the **Reuse script from another target** list, select the first target. This option repeats the configuration of the first target; the scripts, options, targets, and sources used to build the first target will also be used to build the current target. Click Finish.
4. To add more outputs repeat step 3.

The example below shows a debug database file, `example.pdb`, created from the link step. Note that two final targets are produced from the link, but the compiles and the link only occur once.



Application Rule Templates

About Application Rule Templates

An application rule template defines a group of transition rule templates that are used to create an application.

NOTE Application rule templates are not the same as build templates. For details about build templates and the templating language see the *Developer's Reference*.

Application Rule Template Examples

- **CICS COBOL Program w/Assembler**

Input Mask	Transition Rule Template	Output Mask
COBOL()	COBOL Compile	OBJ()
COBOL()	CICS COBOL Compile	OBJ()
ASM()	Assembler Compile	OBJ()
OBJ()	General Program Link	LOAD()

- **MS .NET Executable**

Input Mask	Transition Rule Template	Output Mask
stdafx.cpp	MS C++ Compilation Creating PCH	stdafx.obj,stdafx.pch
.cpp,.pch	MS C++ Compilation Using PCH	*.obj
*.obj	MS C/C++ Linkage	*.exe

Viewing Application Rule Templates

Purpose Follow this procedure to view a list of all the application rule templates currently defined.

To view application rule template:

1. In the navigation pane of the Build Management tab click Settings and click Application Rule Templates. The content pane lists all the application rule templates that are currently defined.
2. To view the properties of an application rule template, in the navigation or content panes click the application rule template. The content pane refreshes and lists all of the current properties.

Adding Application Rule Templates

Purpose Follow this procedure to add a new application rule template.

To add an application rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Application Rule Templates.
2. In the content or menu areas click New Object.

   The Add New Application Rule Template dialog box appears.
3. For **Name** type a name for the new application rule template
4. For **Description** type a description for the new application rule template.
5. Click OK. The new application rule template is added to the list of application rule templates in the content pane.

Modifying Application Rule Templates

Purpose Follow this procedure to modify an application rule template.

To modify an application rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Application Rule Templates.
2. In the content pane select the application rule template that you want to modify and click Edit.

   The Edit Application Rule Template dialog box appears.
3. For **Name** modify the name of the application rule template.
4. For **Description** modify the description of the application rule template.
5. Click OK. The details of the application rule template are updated in the content pane.

Deleting Application Rule Templates

Purpose Follow this procedure to delete application rule templates.

To delete application rule templates:

1. In the navigation pane of the Build Management tab click Settings, and click Application Rule Templates.
2. In the content pane select the check box next to each application rule template that you want to delete, and click Delete.
3. To confirm that you want to delete the application rule templates click Yes.

Adding Transition Rule Templates to Application Rule Templates

Purpose Follow this procedure to add transition rule templates to an application rule template.

NOTE If you update a transition rule template its details are also updated in any application rule template where it is included.

To add transition rule templates to an application rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Application Rule Template.
2. In the content or menu areas select the application rule template where you want to add a transition rule template and click New Object.

   The Assign Target Transition Rule Templates dialog box appears.
3. From the **Select Transition Rule Templates** list select one or more transition rule templates.
4. Click OK. The transition rule templates are added to the content pane.

Removing Transition Rule Templates from Application Rule Templates

Purpose Follow this procedure to remove transition rule templates from application rule templates. This action does not delete the transition rule templates from Dimensions Build.

To remove transition rule templates from an application rule template:

1. In the navigation pane of the Build Management tab click Settings, and click Application Rule Templates.
2. In the navigation or content panes select the application rule template containing the transition rule templates that you want to remove.
3. In the content pane, select the check box next to each transition rule template that you want to remove.
4. Click Delete.
5. To confirm that you want to remove the transition rule templates click Yes.

Chapter 5

Managing Dimensions Build Configurations

About Build Configurations	96
About Build Projects	97
Basic Build Configuration Operations	98
Build Areas	102
Sources	107
Build Targets	111
Inputs	116
Outputs	118
Scripts	119
Build Options	126
Versions	128
Exporting and Importing	129
Copying a Project with its Build Configurations	131
Tips	132

About Build Configurations

A build configuration captures information about what will be built, and how it will be built. Build configurations include the following:

- Build Areas

  A build area defines where the files related to a build will be checked out to, or delivered to. Although build areas are created in the Dimensions Administration Console (and referred to there as file areas), each build configuration must specify one or more build areas. Most builds will use work areas or deployment areas.

- Build Targets

  A build target defines the end-product of a build. Build targets include the specifications for inputs and outputs. Build targets can be real, such as an executable file, or virtual, such as "clean" or "all".

- Sources

  A source can be a source code file, an include file, another target, and so on. A source file must be imported as a source and then defined as an input to a target.

- Build Options

  A build option defines elective settings that may or may not be part of the build, such as a debug switch. Individual options may be collected into option groups. For details of the build options see *Dimensions Build Predefined Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

- Scripts

  A script defines the specific commands necessary to execute the build. Scripts indicate which build tool is to be used, which build options, and so on. You can define scripts to execute before the build, during the build, and after the build.

- Versions

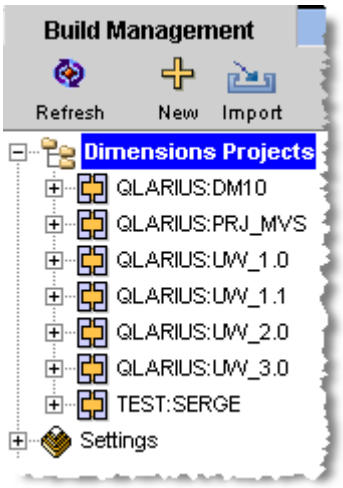
  A version is a previously-checked-in build configuration. Versions are numbered, time- and date-stamped, and include associations with build areas and targets.

  Build configurations can be exported to an XML file (and later re-imported).

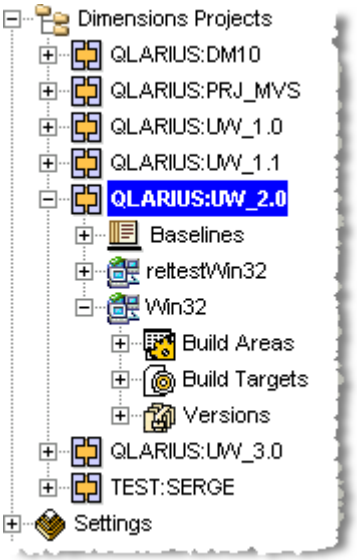
About Build Projects

Dimensions CM organizes build configurations underneath Dimensions projects; that is, projects defined in Dimensions itself. Part of the process of creating a build configuration is to select a Dimensions project.

The navigation pane of the Build Management tab shows the Dimensions projects in use.



If no projects have been associated with a build configuration yet, no Dimensions projects will appear in the tree. Expanding a specific project reveals the build configurations, and their components, underneath:



Similar, but not identical, project and configuration trees appear under the Scheduling, Monitoring, and Notifications tabs. Since those functions are primarily described in sections dedicated to those tasks, this section will describe the behavior underneath the Build Management tab.

Basic Build Configuration Operations

Viewing Build Configurations

To view the existing build configurations, use the following procedure.

To view build configurations:

1. In the Build Administration Console, click the Build Management tab.
2. Expand the Dimensions Projects tree.
3. Select a project. The Build Configurations tab appears, displaying the available build configurations for that project.

Creating Build Configurations

To create a new build configuration, use the following procedure.

To create a new build configuration:

1. If your build configuration requires the creation of a new work area or deployment area, create the area first as described in Build Areas on page 102.
2. In the Build Administration Console, click the Build Management tab.
3. Click the New Object icon (shaped like a plus sign). The Add New Build Configuration dialog box appears.

   NOTE After you create at least one build configuration, associated Dimensions projects appear in the navigation tree. Once Dimensions projects appear, you can select one before clicking the New Object icon. The selected project will appear in the Parent Project field.
4. Use the Properties tab to set the project details and time out details, as described in Setting Build Configuration Properties on page 101.
5. If you wish to create or select a script to be executed before the build is launched (but after source files have been transferred to the build area), use the Pre-Script tab. See Adding a Script on page 120.
6. If you have multiple targets that can use the same script, create or select a main script and then set each target to **Use build configuration main script**. Note that any transition scripts for individual targets will take precedence over the main script. See Adding a Script on page 120.
7. If you wish to create or select a script to be executed after the build is launched, use the Post-Script tab. See Adding a Script on page 120.
8. If you wish to create or select a script to be executed before the build is launched (and before source files are transferred to the build area), use the Clean-Up Script tab. This script is executed if you select **Execute clean-up script before build** when running a build. See Adding a Script on page 120.

Note that creating a build configuration does not require you to add the build target or source files for the project. That can be done later.

Modifying Build Configurations

To modify an existing build configuration, there are two procedures, depending on whether you want to modify the properties and scripts, or the build areas, targets, sources, and options.

To modify the properties or scripts of an existing build configuration:

1. In the Build Administration Console, select the Build Management tab if it is not selected already.
2. Expand the Dimensions Projects tree until you can see the build configuration that you want to modify.
3. Select the build configuration. The Build Configuration Details, Build Areas, and other related sections appear in the content pane.
4. Check out the build configuration if it is not checked out already.
5. Click the Edit/View icon.

   The Edit Build Configuration dialog box appears.
6. Use the Properties tab to edit the project details and time out details, as described in Setting Build Configuration Properties on page 101.
7. Use the Pre-Script tab to edit or specify the details of the script, if any, that is to be executed before the build is launched (but after source files have been transferred to the build area). See Modifying a Build Configuration Script on page 122.
8. Use the Main Script tab to edit or specify the details of the main script. See Modifying a Build Configuration Script on page 122.
9. Use the Post-Script tab to edit or specify the details of the script, if any, that is to be executed after the build is launched. See Modifying a Build Configuration Script on page 122.
10. Use the Clean-Up Script tab to edit or specify the details of the script, if any, that is to be executed before the build is launched (and before any source files are transferred into the build area). See Modifying a Build Configuration Script on page 122.

To modify the build configuration details, build areas, targets, sources, build options, or build option groups associated with a build configuration:

1. In the Build Administration Console, click the Build Management tab.
2. Expand the Dimensions Projects tree until you can see the build configuration that you want to modify.
3. Check out the build configuration if it is not checked out already.
4. Select the project containing the build configuration. The Build Configurations Details section appears, along with the tabs for Build Areas, Build Targets, and Build Options.
5. To edit the project details and time out details, click the pencil icon next to the Build Configuration Details section title. Make the desired changes as described in Setting Build Configuration Properties on page 101.
6. To change a build area used by the build configuration, select the build area and click the pencil icon next to the Build Areas section title. Make the desired changes as described in Modifying Build Areas on page 106.

7 To change a build target used by the build configuration, select the build target and click the pencil icon next to the Build Targets section title. Make the desired changes as described in Modifying Build Targets on page 115.

8 To change a build source used by the build configuration, click the Sources tab, then select the check box next to the build source to be edited and click the pencil icon next to the Build Targets section title. Make the desired changes as described in Modifying Build Sources on page 110.

9 To change a build option used by the build configuration, select the build option and click the pencil icon next to the Build Options section title. Make the desired changes as described in Modifying Build Options on page 127.

10 To change a build option group used by the build configuration, click the Option Groups tab, then select the check box next to the build option group to be edited and click the pencil icon next to the Option Groups section title. Make the desired changes as described in Modifying Build Option Groups on page 76.

Copying Build Configurations

Copying an existing build configuration and then modifying the copy is an efficient way of creating a new build configuration, or even multiple new build configurations.

To copy a build configuration:

1 In the Build Administration Console, click the Build Management tab.

2 Expand the Dimensions Projects tree until you can see the build configuration that you want to copy.

3 Select the build configuration that you want to copy.

4 Click the Copy icon.

The **Create new copy of the build configuration** dialog box appears.

5 Enter the name of the new build configuration.

6 Select the project to receive the new build configuration.

7 Click OK. A confirmation dialog briefly displays, and the new build configuration appears in the Build Configurations section.

Deleting Build Configurations

If you want to delete a build configuration, you can do so using the following procedure.

To delete a build configuration:

1 In the Build Administration Console, click the Build Management tab.

2 Expand the Dimensions Projects tree until you can see the build configuration that you want to delete.

3 Select the project containing the build configuration. The Build Configurations section appears.

4 Select the check box next to the build configuration that you want to delete. If there is more than one in the same project, you can select them all.

5 Click the Delete icon (shaped like an X). The Delete Build Configurations dialog box appears.

6 Click Yes to confirm the deletion.

Setting Build Configuration Properties

The Properties tab makes it possible for you to edit the details of a build configuration.

To create or edit the information on the Properties tab:

1 Edit the desired properties from the list below:

- Parent Project

  Select the Dimensions project to associate with the build configuration.

- Name

  Enter a name for the build configuration.

- Description

  Enter a description for the build configuration.

- Project Relative Path

  Enter a path that is relative to the Dimensions project root directory. This restricts the build to the sub-directory that you specify. For example, if the project root directory is `C:\Projects\Qlarius` and you specified a relative path of `Java` the build will be restricted to `C:\Projects\Qlarius\Java`.

- Project for Targets

  Select a Dimensions project to receive the target, or accept the default value of **the same as Parent Project**.

- Platform

  Specify the platform (Win32, Linux, and so on) for the build configuration.

- Launch Timeout

  Select Default, None, or Custom. This value represents an interval in seconds before Dimensions CM will abandon a build job that is stalled in its launch attempt. If you select **Custom**, an additional field appears, letting you enter a value in seconds.

  NOTE On MVS, the usual value is None and you should be careful when selecting any other value. An incorrect configuration could result in a days-long delay.

- Execution Timeout

  Select Default, None, or Custom. This value represents an interval in seconds before Dimensions CM will abandon a build job that is stalled during the execution of the build script. If you select **Custom**, an additional field appears, letting you enter a value in seconds.

- Type

  Select Default or Openmake.

- Reset Type

  Useful only if **Type** is changed from Default to Openmake. If selected, this check box causes settings required for an Openmake build configuration to be restored.

2 Click OK to accept the dialog box.

3 Click the Check In icon to check in the build configuration.

Build Areas

About Build Areas

Every build configuration in Dimensions CM must be associated with at least one build area. The three types of build areas are:

- Work areas

  Use this type of area to work on files checked out from Dimensions. Defining a work area indicates that you want to have files automatically placed there.

- Deployment areas

  Use this type of area to receive files that have been promoted to a chosen state in the Global Stage Lifecycle. For example, when a set of files is ready for System Test, they can be copied to an area that Quality Assurance uses.

  NOTE

  - Deployment scripts should be placed in the directory `$DM_ROOT/templates` on the node that hosts the area. Do not place the deployment scripts in the area itself.
  - Build areas are created in the Dimensions Administration Console, although they are referred to there as file areas. See the *Process Configuration Guide* for more information about file areas.

Creating a New Work Area

To create a new work area:

1 Log in to the Dimensions Administration Console.

2 Under the Distributed Development heading, click **Area Definitions**. The Area Definitions page appears.

3 Click the New icon. The pop-up menu displays the types of area that you can create:

- Work area
- Deployment area
- Library cache area

4 Select **Work Area**. The New Work Area dialog box appears.

5 Fill out the fields as follows:

- **ID**

  The identifier for the build area.

- **Description**

  Enter a description for the build area.

- **Network Node**

  The name of the machine that Dimensions CM is to use when executing the build.

- **Directory**

  The name of the directory that Dimensions CM is to use as the base directory; relative path names will be evaluated based on this directory.

- **Node User ID**

  The user ID that Dimensions CM is to use when executing the build.

  NOTE

  - On Windows 200x platforms, the user should have administrator privileges. On Windows XP, or on UNIX, USS, or MVS, administrator privileges are not required.
  - You can also enter a credential set. For details see the *System Administration Guide*.

- **Node Password**

  The user password that Dimensions CM is to use when executing the build.

  NOTE Not required if you are using a credential set.

- **Area Owner**

  This is the user ID that has permission to edit the build area.

- **Available Users/Groups** and **Users/Groups granted permission to access this area**

  Select entries from the Available Users/Groups and use the **>>** and **<<** links to move the selected entries to the list of users/groups with permission to access the build area.

6 When you have finished filling out the dialog box, click **OK**. A Results dialog box briefly confirms the creation of the new area. The new area also appears in the File Areas List.

Creating a New Deployment Area

To create a new deployment area:

1. Log in to the Dimensions Administration Console.
2. In the upper right-hand corner, click on Area Definitions. The Area Definitions page appears.
3. Click the New icon. The pop-up menu displays the types of area that you can create:
   - Work area
   - Deployment area
   - Library cache area
4. Select Deployment area. The New Deployment Area dialog box appears:
5. Fill out the fields as follows:
   - **ID**

     The identifier for the build area.
   - **Description**

     Enter a description for the build area.
   - **Network Node**

     The name of the machine that Dimensions CM is to use when executing the build.
   - **Directory**

     The name of the directory that Dimensions CM is to use as the base directory; relative path names will be evaluated based on this directory.
   - **Node User ID**

     The user ID that Dimensions CM is to use when executing the build.

     NOTE

     - On Windows 200x platforms, the user should have administrator privileges. On Windows XP, or on UNIX, USS, or MVS, administrator privileges are not required.
     - You can also enter a credential set. For details see the *System Administration Guide*.
   - **Node Password**

     The user password that Dimensions CM is to use when executing the build. This user should have administrator privileges.

     NOTE Not required if you are using a credential set.
   - **Stage ID**

     When items reach the lifecycle stage specified here, This is the lifecycle stage, named in Dimensions, that will trigger the movement of files.
   - **Area Owner**

     This is the user ID that has permission to edit the build area.

- **Filter**

  This is a template that determines which files will be taken up by the build. For example, a filter defined as "*.java" would only accept Java source files.

- **Pre-event Transfer Script**

  This is the name of a script to be executed before the files are transferred to the deployment area.

- **Post-event Transfer Script**

  This is the name of a script to be executed after the files are transferred to the deployment area.

- **Fail-event Transfer Script**

  This is the name of a script to be executed if the operation to transfer the files to the deployment area fails.

- **Is this area currently offline**

  If selected, this check box will cause the area to be excluded from file transfer operations. (You might want to do this to avoid overwriting a specific directory.)

6 When you have finished filling out the dialog box, click OK. A Results dialog box briefly confirms the creation of the new area. The new area also appears in the File Areas List.

Creating a New Library Cache Area

To create a new library cache area:

1 Log in to the Dimensions Administration Console.

2 In the upper right-hand corner, click Area Definitions. The Area Definitions page appears.

3 Click the New icon. The pop-up menu displays the types of area that you can create:

- Work area
- Deployment area
- Library cache area

4 Select Library cache area. The New Library Cache Area dialog box appears:

5 Fill out the fields as follows:

- **ID**

  The identifier for the build area.

- **Description**

  Enter a description for the build area.

- **Network Node**

  The name of the machine that Dimensions CM is to use when executing the build.

- **Directory**

  The name of the directory that Dimensions CM is to use as the base directory; relative path names will be evaluated based on this directory.

- **Node User ID**

  The user ID that Dimensions CM is to use when executing the build.

  NOTE On Windows 200x platforms, the user should have administrator privileges. On Windows XP, or on UNIX, USS, or MVS, administrator privileges are not required.

- **Node Password**

  The user password that Dimensions CM is to use when executing the build.

- **Area Owner**

  This is the user ID that has permission to edit the build area.

- **Is this area currently offline**

  If selected, this check box will cause the area to be excluded from file transfer operations. (You might want to do this to avoid overwriting a specific directory.)

6 When you have finished filling out the dialog box, click OK. A Results dialog box briefly confirms the creation of the new area. The new area also appears in the File Areas List.

Modifying Build Areas

You can modify a build area after it has been created. However, once a build area is associated with a build configuration, you cannot modify the following properties:

- Area ID
- Stage ID
- Network node
- Directory

To modify a build area:

1 From the Dimensions Administration Console, click Area Definitions.

2 Click the name of the build area you want to modify. The Edit Build Area dialog box appears.

3 Modify the details that you want to change.

4 Click OK to accept the dialog box.

Deleting Build Areas

If you have the privileges to create a build area, you can also delete the build area.

NOTE Once the build area has been included in a checked-in build configuration, that build area can no longer be deleted. If the build configuration has not been checked in, you can detach the build area and then delete. If the build configuration has been checked in, you can also delete the entire build configuration.

To delete a build area:

1. From the Dimensions Administration Console, click Area Definitions.
2. Place a check mark next to the build area you want to delete.
3. Click the Delete icon. The Delete Area dialog box appears.
4. Click OK to confirm the deletion. Dimensions displays a brief confirmation dialog box, and refreshes the list of build areas.

Sources

About Sources

Sources are files or other items needed to produce a target. To define a target such as an executable file, you specify which source files, include files, and other items are needed to create the target.

You can define sources at any time after a build configuration has been created.

You can add build sources from a file on disk, or from a file already archived in Dimensions.

Adding a Build Source from a File

To add a build source from a file:

1. In the navigation pane of the Build Management tab, locate and select the build configuration to which you want to add a source.
2. Make sure the build configuration is checked out.
3. Expand the build configuration until the Build Targets entry is visible in the project tree. This will also cause the Sources section to appear in the content pane.
4. In the content pane, click the Sources title to make it active.
5. Click the New Object icon (shaped like a plus sign). The Add New Source dialog box appears.
6. For **File Name or Mask** enter the filename of the source file, or the mask specifying wildcards.

7 For **Relative path** enter a path to the source that is relative to the project relative path. This restricts the path to the sub-directory that you specify. For example, if the project relative path is `C:\Projects\Qlarius\Java` and you specify a relative path of Utilities, the path to the source will be restricted to `C:\Projects\Qlarius\Java\Utilities`.

8 For **Build Type** enter an item format. For details about using item formats to control build rules see page 41.

9 Click OK. The new source appears in the Sources section.

Using Wildcard Patterns to Add Build Sources from Dimensions

NOTE You cannot use the procedure described below on MVS, see Support for Wildcards in Build Configurations on page 32.

When you add a build source from Dimensions, you can add individual files or wildcard patterns representing multiple files. An example of a wildcard pattern is *.java. When you are adding many files, it is easier to use wildcard patterns.

To use wildcard patterns to add build sources from Dimensions:

1 In the navigation pane of the Build Management tab, locate and select the build configuration to which you want to add a source.

2 Make sure the build configuration is checked out.

3 Expand the build configuration until the Build Targets entry is visible in the project tree. This will also cause the Sources section to appear in the content pane.

4 In the content pane, click the Sources title to make it active.

5 Click the Add Sources from Dimensions icon. The Select Dimensions Source(s) dialog box appears.

6 Expand the directory tree on the left. The folders displayed correspond to the folder structure for the associated Dimensions project. As you select each folder, a wildcard pattern appears in the Patterns section in the content pane.

7 Select the desired wildcard pattern. Notice that the pattern appears at the bottom of the dialog box.

NOTE To select the entire contents of a directory, select the pattern "*". This pattern appears as long as there is at least one file in the directory.

8 Repeat with other folders until you have selected all desired wildcard patterns.

9 If you also need to select individual files, use the procedure described in Adding Individual Files in Dimensions as Build Sources on page 109. You can switch back and forth between the two methods.

10 Click OK. The selected wildcard patterns and items appear in the Sources area of the Build Management tab.

NOTE By default, the Recursive Search check box is selected when Use File Patterns is selected. This means that a search will search throughout a directory tree. See the section on Recursive Search, on page 110, for more information.

Adding Individual Files in Dimensions as Build Sources

In some situations you may want to add individual files as sources. To do this, deselect the Use File Patterns check box in the Select Dimensions Sources dialog box.

To add individual files in Dimensions as build sources:

1. In the navigation pane of the Build Management tab, locate and select the build configuration to which you want to add a source.
2. Make sure the build configuration is checked out.
3. Expand the build configuration until the Build Targets entry is visible in the project tree. This will also cause the Sources section to appear in the content pane.
4. In the content pane, click the Sources title to make it active.
5. Click the Add Sources from Dimensions icon. The Select Dimensions Source(s) dialog box appears.
6. Deselect the Use File Patterns check box. The Files section is displayed in the content pane. When you select a folder, the individual file listings appear.
7. Select the desired files. Notice that the file specifications appear at the bottom of the dialog box.
8. If you also need to select files using a wildcard pattern, use the procedure described in Using Wildcard Patterns to Add Build Sources from Dimensions on page 108. You can switch back and forth between the two methods.
9. Click OK. The selected items and wildcard patterns appear in the Sources area of the Build Management tab.

Recursive Search

In the Select Dimensions Sources dialog box, there is a check box for Recursive Search. By default, it is selected whenever Use File Patterns is selected:

When Recursive Search is on, Dimensions CM will generate wildcard patterns that will match an entire directory tree, or a file anywhere in the directory tree. For example:

```
**/*@src/
```

If Recursive Search is turned off, Dimensions CM will generate wildcard patterns that will search only in the current directory:

```
*@src/
```

Recursive Search is also disabled if you deselect Use File Patterns.

Modifying Build Sources

To modify an existing build source:

1. In the navigation pane of the Build Management tab, locate and select the build configuration containing the source you want to modify.
2. Make sure the build configuration is checked out.
3. Expand the build configuration until the Build Targets entry is visible in the project tree. This will also cause the Sources section to appear in the content pane.
4. In the content pane, click the Sources title to make it active.
5. Click the Edit Selected Object icon (shaped like a pencil). The Edit Source dialog box appears.
6. For **File Name or Mask** revise the filename of the source file, or the mask specifying wildcards.
7. For **Relative path** enter or revise a path to the source that is relative to the project relative path. This restricts the path to the sub-directory that you specify. For example, if the project relative path is `C:\Projects\Qlarius\Java` and you specify a relative path of Utilities, the path to the source will be restricted to `C:\Projects\Qlarius\Java\Utilities`.
8. For **Build Type** enter an item format. For details about using item formats to control build rules see page 41.
9. Click OK. The edited source appears in the Sources section.

Deleting Build Sources

To delete an existing build source or sources:

1. Locate the build configuration containing the source you want to delete.
2. Make sure the build configuration is checked out.
3. Expand the build configuration until the Build Targets entry is visible in the project tree. This will also cause the Sources section to appear in the content pane.
4. Select the Build Targets heading or the name of the build configuration itself. Detail information for the build targets appears in the content pane.
5. Click the Sources title to make it active.
6. Place a check mark next to the source or sources you want to delete.
7. Click the Delete icon (shaped like a red X). A confirmation dialog box asks you to confirm the deletion.
8. Click Yes to confirm the deletion; click Cancel to leave the sources as is.

Build Targets

About Build Targets

A build target is the desired end product of the build job, such as an executable file. An executable file is an example of a **real** build target.

Build targets can also be **virtual**. A virtual target is a target that represents an abstract idea, such as "all". A target of "all" would mean to build all targets, not an executable file named "all".

Each target is associated with a build configuration. You must create a build configuration before attempting to create a target. Remember to also have the build configuration checked out before attempting to modify it.

Creating Build Targets

When you create a new build target, you invoke the Create New Target Wizard. This wizard has four pages:

- Target page

  Contains information about the target itself.

- Script page

  Contains information about the script used to build the target.

- Options page

  Contains information about build options and build option groups, if any.

- Build Plan page

  Offers a chance to add a source file, attach an existing source file, define or attach dependent targets, or to go back and alter details on any of the three previous pages.

The remainder of this section describes the process of creating a new target, but does not go into detail for the Script and Options pages. Those pages are described in the Scripts section and the Build Options section.

To create a build target:

1. From the Build Administration Console, select the build configuration that will receive the new target. The Build Targets section appears on the right side of the application window.
2. Click the New object icon (shaped like a plus sign). The Create New Target wizard appears. By default, the Target page appears first.
3. Fill out the Target page and the other three wizard pages as described in the following sections.

Fill out the Target page

On this page, you enter the identification details for the target.

1. For Name, enter a name for the target.
2. For File, enter the filename for the target. This must be the exact filename of the target on the build node.
3. For Relative Path, enter the path relative to the build area.
4. For Description, enter a description for the target.
5. Select the Is Virtual check box if the target is a virtual target.
6. Select the Is Final check box if the target is not an intermediate target; that is, the target will not be used as the input to a subsequent build operation.
7. For Design Part, to relate a design part to the target, enter a design part name or click Browse and select one from the tree.
8. Click Next to reach the Scripts page.

Fill out the Scripts page

On this page, you select the source for the script, and enter the text of the script if needed.

1 For **Build Order** enter a numerical value that specifies the order in which the transition is built.

- A value of 0 indicates no order.
- Transitions are ordered in the BRD (Build Request Definition) file from 1 to n regardless of where they appear in the tree in Dimensions Build. For example, all transitions with a value of 1 will be built before those with a value of 2, and so on.
- This functionality allows you to manually set the order of build steps for unrelated items that Dimensions Build cannot order automatically.
- The build order of a nested transition cannot be higher than its parent. For example, assume that you have the transition COBOL(*) to OBJECT(*) to LOAD(*). OBJECT(*) can not have a build order of 2 if LOAD(*) has a build order of 1 as it does not make sense to build the LOAD before the OBJECT.

2 Select the **Expand Wildcards** check box to:

- Expand all the wildcard inputs and outputs for this transition at build time.
- Create an instance of the transition for each item that matches the wildcard.

For more details see Support for Wildcards in Build Configurations on page 32.

NOTE

- The default state of this check box is determined by the platform of the build configuration. For an MVS platform the default is 'expand' (checked). For all other platforms, the default is 'no expand' (not checked).
- If you have multiple targets in a build configuration you can use both settings (expand and not expand).

3 For **Select the source for script content when building the target**, choose from one of the following four choices:

- **input script content manually**
- **Use a Dimensions-controlled file as a script**
- **Use a file in the build area as a script**
- **Use the build configuration main script**

Enter the text for the script, or select one of the other three choices that designate a script that has already been created.

For a detailed description of all four choices, see the Scripts section on page 119.

4 Click Next to reach the Options page.

Fill out the Options page

On this page, you define build options, or attach previously-defined option groups.

Build options and option groups are not required, but can be attached here if desired.

- If you do not want to define an option or attach an option group, be sure Done is selected, then click Next to reach the Build Plan page.
- If you want to define a new build option:
  - **a** Select **Add new option** and then click Next. The Add New Option page appears.
  - **b** Fill out the Add New Option page as described in Creating Build Options on page 126.
  - **c** Click Next. You are returned to the Options tab.
- If you want to attach a build option group:
  - **a** Click the Option Groups tab on the Options page.
  - **b** Select **Add new option group** and then click Next. The Add Option Group page appears.
  - **c** Select the desired option group and then click Next. You are returned to the Option Groups tab.

For a detailed description of options, see the Build Options section. For a discussion of option groups, see Build Option Groups on page 72.

When you are done with the Options page, click Next to reach the Build Plan page.

Fill out the Build Plan page

On this page, you define the inputs to the target. Inputs can be source files or other targets, either existing or yet to be defined.

You can also go back and modify an existing target or source file.

- If you want to add a source file:
  - **a** Select **Add new source** and Click Next. The Add New Source page appears.
  - **b** Enter the filename or mask (example: *.java).
  - **c** Enter the path to the file or files, relative to the build area. For the current directory, use ".\".
  - **d** For **Build Type** enter an item format. For details about using item formats to control build rules see page 41.
  - **e** Click Next. You are returned to the Build Plan page.
- If you want to add a target that the displayed target uses as an input:
  - **a** Select **Add new target**, then click Next. The Add New target page appears.
  - **b** Fill out the fields as described in Fill out the Target page, then click Next. The Script page appears.

c Fill out the Script page as described in Fill out the Scripts page, then click Next. The Modify Options page appears.

d Fill out the Modify Options page as described in Fill out the Options page, then click Next. You are returned to the Build Plan page.

When you are done adding source files or targets, click Finish. The new targets and source files appear in the Build Targets area of the Build Administration Console.

Modifying Build Targets

To modify an existing build target:

1 Make sure the build configuration containing the target is checked out.

2 Expand the build configuration until the target is visible in the project tree.

3 Select the name of the target. The target details appear in the content pane.

4 Edit the desired target details.

5 Check the build configuration back in.

Deleting Build Targets

To delete an existing build target or targets:

1 Locate the build configuration containing the target you want to delete.

2 Make sure the build configuration is checked out.

3 Expand the build configuration until the Build Targets entry is visible in the project tree.

4 Select the Build Targets heading or the name of the build configuration itself. Detail information for the build targets appears in the content pane.

5 Place a check mark next to the target or targets you want to delete.

6 Click the Delete icon (shaped like a red X). A confirmation dialog box asks you to confirm the deletion.

7 Click Yes to confirm the deletion; click Cancel to leave the targets as is.

Inputs

About Inputs

Inputs to a target can be sources, or other targets.

You can define new sources and targets as inputs, or select inputs from existing sources and targets.

Adding a New Target as an Input

To add a new target as an input:

1. In the Build Management tab, expand the navigation tree until you can see the target whose script you want to view.
2. Select the target. The Inputs section appears in the content pane.
3. Click the **Add new target as an input** icon.

   The Create New Target wizard appears.
4. Complete the Create New Target Wizard as described in Creating Build Targets on page 111.
5. When you are done with the wizard, check in the modified build configuration.

Adding a New Source as an Input

To add a new source as an input:

1. In the Build Management tab, expand the navigation tree until you can see the target whose script you want to view.
2. Select the target. The Inputs section appears in the content pane.
3. Click the **Add new source as an input** icon.

   The Add New Source wizard appears.
4. Enter a name for the new source.
5. Enter a path to the source that is relative to the project relative path. This restricts the path to the sub-directory that you specify. For example, if the project relative path is `C:\Projects\Qlarius\Java` and you specify a relative path of `Utilities`, the path to the source will be restricted to `C:\Projects\Qlarius\Java\Utilities`.

6 Click OK to accept the dialog box.

7 Check in the modified build configuration.

Adding an Existing Target as an Input

To add an existing target as an input:

1 In the Build Management tab, expand the navigation tree until you can see the target whose script you want to view.

2 Select the target. The Inputs section appears in the content pane.

3 Click the **Add existing target as an input** icon.

The Add Existing Target As Input dialog box appears.

4 Select from the list of existing targets.

5 Click OK to accept the dialog box.

6 Check in the modified build configuration.

Adding an Existing Source as an Input

To add an existing source as an input:

1 In the Build Management tab, expand the navigation tree until you can see the target whose script you want to view.

2 Select the target. The Inputs section appears in the content pane.

3 Click the **Add existing source as an input** icon.

The Add Existing Source As Input dialog box appears.

4 Select from the list of existing sources.

5 Click OK to accept the dialog box.

6 Check in the modified build configuration.

Modifying Inputs

To modify an input:

1 In the Build Management tab, expand the navigation tree until you can see the target whose input you want to modify.

2 Be sure that the build configuration is checked out.

3 Select the target. The Inputs section appears in the content pane.

4 Select the input you wish to modify.

5 Click the Edit selected objects icon (shaped like a pencil). If the input is a source, the Edit Source dialog box appears. If the input is a target, the Edit Build Target dialog box appears.

6 Edit the input:

- If the Edit Source dialog box appears, follow the instructions given in Modifying Build Sources on page 110.
- If the Edit Build Target dialog box appears, follow the instructions given in Modifying Build Targets on page 115.

7 When you have finished modifying the input, check in the build configuration.

Deleting Inputs

To delete an input:

1 In the Build Management tab, expand the navigation tree until you can see the target whose input you want to delete.

2 Be sure that the build configuration is checked out.

3 Select the target. The Inputs section appears in the content pane.

4 Select the input you wish to delete.

5 Click the Delete selected objects icon (shaped like a red X). A confirmation dialog box appears.

6 Click Yes to delete the input; click Cancel to leave the input as is.

7 When you have finished deleting the input, check in the build configuration.

Outputs

About Outputs

Build targets, both real and virtual, appear in the content pane as outputs.

When you add a new output, you are essentially adding a new target. Although there are two icons next to the Outputs title—one for adding a new output, the other for editing the selected output—the icons call the Create New Target Wizard and Edit Build Target dialog box respectively.

Use the procedures described in Creating Build Targets on page 111 and Modifying Build Targets on page 115 to create and modify outputs.

To view the outputs list, select a specific target in the navigation tree.

Scripts

About Scripts

Scripts determine the specific behavior of a build job. For example, you can define a script to do any of the following:

- compile and link
- clean up the build area before the build executes
- rename files that have been transferred into the build area, before the build executes
- delete temporary files after the build executes

Scripts can be defined at a build configuration level or a target level. (Target-level scripts are known as transition scripts.) There are important differences between the two.

Build Configuration Scripts

Scripts associated with build configurations generally apply to the build as a whole, and come in four different varieties.

- Pre-Script

  This type of script is executed before the main script, but after sources are transferred into the build area.

- Main Script

  This script is not required, but can be useful when you have many targets that can use the same script. In that event, you can specify that individual targets should use this script. If the scripts are different, the transition script for an individual target will take precedence.

- Post-Script

  This type of script is executed after the main script.

- Clean-Up Script

  This type of script is executed before the main script, and before sources are transferred into the build area.

When you create a new build configuration there is a tab for each of these scripts.

Transition Scripts

Transition scripts apply to one or more targets. Unlike build configuration scripts, transition scripts come in only one variety, and are defined at the time a target is created.

Reusing Scripts

Dimensions CM offers you two ways to reuse scripts.

- Once a transition script has been defined, any other target in the build configuration can use the script. The Target page of the Create New Target Wizard contains a field named **Reuse script from another target**:

  The pulldown list allows you to select a script that has already been written for another target in the build configuration.

- You can also re-use a build configuration script as a transition script. Select **Use the build configuration main script** when creating a transition script.

Adding a Script

Whether you are creating a build configuration script or a transition script, you can select any of the following three ways to add a script:

- Input script content manually
- Use a Dimensions-controlled file as a script
- Use a file in the build area as a script

You can use each of these methods whether you are creating a Pre-Script, Main Script, or Post-Script, or Clean-Up Script.

NOTE When you create a new target, you must fill out at least the name of the target before you can move to the Script page of the Create New Target Wizard.

If you are creating a transition script, you can also select **Use the build configuration main script**.

Entering Content Manually

You can enter the content of the script manually. This method is best when you are creating test configurations, or when the content of the build script is simple enough so that you can type it from memory.

To enter the content of the script manually:

1. From the Script page of the Create New Target wizard, select **Input script content manually** as a script from the drop-down list. The Script Content text field appears.
2. Enter the content of the script into the Script Content area.

   NOTE Dimensions CM will look for the build script and any other files in the build directory. If you want to refer to files that are elsewhere, you will have to specify the location of those files in the script.

3. Click Next to define build options, or click Finish to complete the target definition.

Using a Dimensions-Controlled File

NOTE If you select a Dimensions-controlled file as a script, be aware that the file will be pushed to the build area if you are using a work area. If you are using a deployment area, Dimensions Build assumes that the script has already been deployed to that area.

To use a Dimensions-controlled file as a script:

1. From the Script page of the Create New Target wizard, select **Use Dimensions-controlled file** as a script from the drop-down list. A text field appears, with a Browse button next to it.
2. Click the Browse button. The Select File from Dimensions dialog box appears.
3. Expand the Projects or Baselines tree to display the available files.
4. Select the Dimensions-controlled file. A check mark appears next to the file name.
5. Click OK. The specification for the file appears in the text field for the script name.
6. Click Next to define build options, or click Finish to complete the target definition.

Using a File in the Build Area

To use a file in the build area as a script:

1. From the Script page of the Create New Target wizard, select **Use a file in the build area as a script** from the drop-down list. A text field appears.
2. If you already know the name of the file containing the build script, enter it in the text field of the dialog box; otherwise, click Browse to display the Build Area Browser.
3. Select the file from the browser, then click Select. The file appears in the Edit Transition Script dialog box.
4. Click Next to define build options, or click Finish to complete the target definition.

Using the Build Configuration Global Script

To use the build configuration global script:

1. From the Script page of the Create New Target wizard, select **Same as build configuration global script** from the pulldown list.
2. Click Next to define build options, or click Finish to complete the target definition.

Viewing a Build Configuration Script

To view a build configuration script:

1. In the Build Management tab, expand the navigation tree until you can see the build configuration whose script you want to view.
2. Select the build configuration.
3. Click the View icon or the Edit/View icon:
   - If you have the build configuration checked out, the icon will be shaped like a pencil. Clicking the icon displays the Edit Build Configuration dialog box.
   - If the build configuration is not checked out, the icon will be shaped like a magnifying glass. Clicking the icon displays the View Build Configuration dialog box.

Viewing a Transition Script

To view a transition script:

1. In the Build Management tab, expand the navigation tree until you can see the target whose script you want to view.
2. Select the target. The Transition Details section appears in the content pane.
3. Click the View icon or the Edit/View icon:
   - If you have the build configuration checked out, the icon next to the Transition Details title will be shaped like a pencil. Clicking the icon displays the Edit Transition Script dialog box.
   - If the build configuration is not checked out, the icon next to the Transition Details title will be shaped like a magnifying glass. Clicking the icon displays the View Transition Script dialog box.

Modifying a Build Configuration Script

The following procedure applies to Pre-Scripts, Main Scripts, Post-Scripts, and Clean-Up Scripts.

To modify a build configuration script:

1. If you already have the build configuration checked out and the Pre/Main/Post/Clean-Up Script tabs are visible, skip to Step 9.
2. In the Build Administration Console, click the Build Management tab.
3. Expand the Dimensions Projects tree until you can see the build configuration whose script you want to modify.
4. Select the project containing the build configuration. The Build Configurations section appears in the content pane.
5. Place a check mark next to the build configuration that will receive the script.
6. Check out the build configuration by clicking the Check Out icon.

7 Click the pencil icon. The Edit Build Configuration dialog box appears.

8 Click the Pre/Main/Post/Clean-Up Script tab.

9 To modify the script, select one of the following three choices:

- input script content manually

  If you select this choice, enter or modify the script in the **Script Content** text box. Select the Word Wrap check box if you want to have the text wrap.

- use a Dimensions-controlled file as a script. If you select this choice, enter or modify the location of the file, or click **Browse** to browse for it.

  NOTE If you select a Dimensions-controlled file as a script, be aware that the file will be pushed to the build area if you are using a work area. If you are using a deployment area, Dimensions Build assumes that the script has already been deployed to that area.

- use a file in the build area as a script

  If you select this choice, enter or modify the location of the file, or click **Browse** to browse for it.

10 If you added or selected a new script, and need to delete an existing script, do so now.

11 Click OK to accept the dialog box.

12 Check in the modified build configuration by clicking the Check In icon.

Modifying a Transition Script

The following procedure applies to transition scripts.

To modify a transition script:

1 In the Build Administration Console, click the Build Management tab.

2 Check out the build configuration containing the target whose transition script you wish to modify.

3 In the navigation tree, Select the target. The Transition Details section appears in the content pane.

4 Click the pencil icon next to the Transition Details title. The Script page of the Edit Target Wizard appears.

5 For **Build Order** enter a numerical value that specifies the order in which the transition is built.

- A value of 0 indicates no order.
- Transitions are ordered in the BRD (Build Request Definition) file from 1 to n regardless of where they appear in the tree in Dimensions Build. For example, all transitions with a value of 1 will be built before those with a value of 2, and so on.

- This functionality allows you to manually set the order of build steps for unrelated items that Dimensions Build cannot order automatically.
- The build order of a nested transition cannot be higher than its parent. For example, assume that you have the transition COBOL(*) to OBJECT(*) to LOAD(*). OBJECT(*) can not have a build order of 2 if LOAD(*) has a build order of 1 as it does not make sense to build the LOAD before the OBJECT.

6 Select the **Expand Wildcards** check box to:

- Expand all the wildcard inputs and outputs for this transition at build time.
- Create an instance of the transition for each item that matches the wildcard.

For more details see Support for Wildcards in Build Configurations on page 32.

NOTE

- The default state of this check box is determined by the platform of the build configuration. For an MVS platform the default is 'expand' (checked). For all other platforms, the default is 'no expand' (not checked).
- If you have multiple targets in a build configuration you can use both settings (expand and not expand).

7 If you want to select a different build tool, use the pull down list to make a new selection for that field.

8 To modify the main part of the script, select one of the following four choices:

- input script content manually

  If you select this choice, enter or modify the script in the **Script Content** text box. Select the Word Wrap check box if you want to have the text wrap.

- use a Dimensions-controlled file as a script

  If you select this choice, enter or modify the location of the file, or click **Browse** to browse for it.

  NOTE If you select a Dimensions-controlled file as a script, be aware that the file will be pushed to the build area if you are using a work area. If you are using a deployment area, Dimensions Build assumes that the script has already been deployed to that area.

- use a file in the build area as a script

  If you select this choice, enter or modify the location of the file, or click **Browse** to browse for it.

- use the build configuration main script

  If you select this choice, you need not enter any other information, as Dimensions CM knows where to find the main script.

9 If you added or selected a new script, and need to delete an existing script, do so now.

10 Click Finish to accept the changes.

11 Check in the modified build configuration by clicking the Check In icon.

Deleting a Build Configuration Script

Use the following procedure to delete a build configuration script. Remember that at least the main script must be present for a build configuration to run successfully.

To delete a build configuration script:

1. If you already have the build configuration checked out and the Pre/Main/Post-Script tab is visible, skip to Step 9.
2. In the Build Administration Console, click the Build Management tab.
3. Expand the Dimensions Projects tree until you can see the build configuration whose script you want to delete.
4. Select the project containing the build configuration. The Build Configurations tab appears.
5. Place a check mark next to the build configuration to be modified.
6. Check out the build configuration by clicking the Check Out icon.
7. Click the pencil icon. The Edit Build Configuration dialog box appears.
8. Click the Pre/Main/Post-Script tab.
9. Select **input script content manually** if it is not selected already.
10. Delete any text that appears in the **Script Content** text box.
11. Click OK to accept the dialog box.
12. Check in the modified build configuration by clicking the Check In icon.

Deleting a Transition Script

Use the following procedure to delete a transition script.

To delete a transition script:

1. In the Build Administration Console, click the Build Management tab.
2. Check out the build configuration containing the target whose transition script you wish to delete.
3. In the navigation tree, select the target. The Transition Details section appears in the content pane.
4. Click the pencil icon next to the Transition Details title. The Edit Transition Script dialog box appears.
5. Delete any text that appears in the **Script Content** text box. (You should of course replace the deleted text using some other source, or the target may not build successfully.)
6. Click OK to accept the dialog box.
7. Check in the modified build configuration by clicking the Check In icon.

Build Options

About Build Options

Build options provide a way of passing configuration information to the build environment. You can define a build option to pass along compiler flags, or you can define a build option to specify the path to a specific compiler. For details of the build options see *Dimensions Build Predefined Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

Dimensions CM offers you several different ways to define build options.

- From the Build Configuration

  Define options at the build configuration level if you want the options to be available to all targets in that build, as well as to the Pre-Script, Post-Script, and Clean-Up Script. An example of this would be a particular search path.

- From the Build Area

  Define options at the build area level if you want the options to be available to all builds in that area. An example of this would be the name of the License Manager host.

- From the Target

  Define options at the target level when they are specific to a particular target. An example of this would be a debug or trace option.

  Options defined during the process of creating a new target, or editing an old target, are actually defined at the transition level. Therefore, the options are available to any other target in the build configuration. In fact, you can easily attach the options defined for one target to a second target by selecting **Reuse script from another target** when you create the second target.

- From the Run Build Wizard

  Define options in the Run Build wizard if you want the options to apply only to that one execution of the build. An example of this would be to add an option to print the time of execution.

Creating Build Options

To create a build option from a build configuration, build area, or target:

1. Check out the build configuration if you have not already done so.
2. Select the build configuration, build area, or target.
3. Click the New Object icon (shaped like a plus sign) next to the Build Options area. The Add New Build Option dialog box appears.
4. Enter a **Name** for the option. This is the name that will be used in build scripts later. The name can contain spaces.

   NOTE If you use the same name for build options in all areas, the following precedence table applies, in order of highest priority to lowest priority:

a Build options defined in the Run Build wizard

b Build options defined at the target level

c Build options defined at the build area level

d Build options defined at the build configuration level

5 Enter an optional **Description** for the option.

6 For the **Value Type**, select from Freeform text, Option Rule, and Option Link.

- Freeform text
- Option Rule
- Option Link

7 Enter the **Value** for the option.

8 Click **OK**. A confirmation dialog box appears briefly, and the new option appears in the Build Options area.

NOTE You can also specify build options in the Run Build wizard (see page 134). However, options that you specify are not saved.

Modifying Build Options

You cannot edit build options defined in the Run Build wizard, because they are not saved with the build job. You can edit build options defined in a build configuration, build area, or build target.

To edit a build option:

1 Locate the build option you want to modify.

2 Make sure the build configuration containing the build option is checked out.

3 Select the build area, build configuration, or target. The Build Options section appears on the right side of the screen.

4 Click the name of the build option. The Edit Build Option dialog box appears.

5 Make the desired edits to the build option, then click OK. A brief confirmation dialog appears, and the edited build option appears in the Build Options section.

Deleting Build Options

To delete a build option:

1. Locate the build configuration containing the build option you want to delete.
2. Make sure the build configuration is checked out.
3. Select the entry containing the build option:
   - If the build option is attached to the build configuration, select the build configuration name. The Build Options heading should be visible in the content pane.
   - If the build option is attached to a specific target, select the target name. The Build Options heading should be visible in the content pane.
4. Place a check mark next to the option or options you want to delete.
5. Click the Delete icon (shaped like a red X). A confirmation dialog box asks you to confirm the deletion.
6. Click Yes to confirm the deletion; click Cancel to leave the build option as is.

Creating Build Option Groups

See Build Option Groups on page 72 for instructions on creating build options groups.

Versions

About Versions

Versions represent previously-checked-in editions of a build configuration. You can do only certain things with previous versions:

- You can view previous versions of a build configuration, as described in Viewing Build Configurations on page 98.
- You can copy previous versions of a build configuration, as described in Copying Build Configurations on page 100.
- You can rollback to a previous version.
- You can rewrite the current build configuration so that it is identical to a previous version.
- You cannot delete a previous version.

For information on how to rollback or rewrite, see Versioning on page 143.

Exporting and Importing

About Exporting and Importing

You can export your Dimensions CM build configurations to an external XML file. These files can be imported at a later time, or to a different machine.

You can also import build configuration files from either Ant or the Openmake/Serena ChangeMan Builder products. The procedures for importing build configurations from either of those products differ from the procedures presented in Importing a Build Configuration on page 129. See Using Dimensions CM Build with Ant, on page 183, and Using Dimensions Build with Openmake, on page 289.

Exporting a Build Configuration

If you need to save your build configuration, you can export the build configuration to an XML file.

To export a build configuration to an XML file:

1. Select the name of the surrounding project.
2. Select the name of the build configurations you want to export.
3. Click the Export icon.
4. A browser window opens with the XML file.
5. Save the XML file.

   NOTE You can use the default filename of `HandleBuildConfig.do`, or rename the file as you see fit.

Importing a Build Configuration

Once you have saved a build configuration to an XML file, you can restore that build configuration by importing the XML file.

Before You Begin

- It's best if you have a project ready to receive the imported build configuration.
- Don't worry about the configuration having the same name as the one from which it was exported; you will be given an opportunity to rename it during the import.

To import a build configuration:

1. Select the name of the project that will receive the build configuration.
2. In the Build Configurations section, click the Import build configuration icon. The Import Build Configuration Wizard appears.
3. Enter or browse for the name of the XML file to be imported.
4. Select the name of the build configuration you want to import. (If there is only one build configuration in the file, the wizard skips this step.)

5 Enter a new name for the build configuration.

6 Click Finish to confirm the import operation.

7 A brief confirmation displays, and the new build configuration appears in the Build Configurations list.

Importing an Ant build.xml File

Users of Ant can import their `build.xml` files into build configurations in Dimensions CM.

NOTE Importing Ant and Openmake files requires the use the Import Wizard—not the Import icon that appears on the top-level row of icons when a project or the Dimensions Projects heading is selected.

To import an Ant `build.xml` file into Dimensions CM:

1 In Dimensions CM, select the Build Management tab.

2 In the navigation pane, click Build Projects, expand your build project, and click the build configuration where you want to import the `build.xml` file.

3 In the content pane, in the Build Targets section, click Launch Import Wizard.

The Import Wizard appears. Note that the on the left side of the wizard there are links to four pages: Select Import Type, Choose File, Parameters, and Confirm. By default, the Select Import Type page is displayed.

4 On the Select Import Type page, select **Ant build configuration (build.xml) file** and click **Next**.

5 On the Choose File page, in the **Select Ant build configuration file** field, do one of the following:

- Type the full path to the `build.xml` file that you want to import.
- Click **Browse**, navigate to the build.xml file that you want to import, select it, and click **Open**.

Click **Next**.

6 On the **Parameters** page, select the sources that you want to import from the `build.xml` file, then click **Next**.

7 On the **Confirm** page review the details of the target that you are going to create and click Finish. Dimensions CM briefly displays a confirmation dialog, and afterwards you can see the new targets in the Build Targets section.

8 Click the Sources tab. Verify that the new sources were added to your list of build sources.

9 To review the transition script that will be used to build the target do the following:

- In the navigation pane, select the build target that you just imported.
- In the content panes, in the Transition Details section, click Edit Build Script.

  The Script page of the Edit Target wizard appears. The Script Content field displays the following default script:

  ```
  ant -buildfile "build.xml" "<name of target>"
  ```

- Modify the script if required.
- Click Finish.

10 Optionally add build options and option groups to the build target.

Importing an Openmake Build Configuration

Users of the Openmake product from Catalyst Systems Corporation, or of the Serena ChangeMan Builder product, can import the TGT file that defines build targets under either of those two systems.

See Using Dimensions Build with Openmake on page 289.

Copying a Project with its Build Configurations

When you copy a project or stream you can also copy all its build configurations. Use one of the following methods:

- In the web and desktop clients open the New Project wizard and select the option Copy build configuration. For details see the *User's Guide*.
- In the DWS and CS commands specify the option /COPY_CONFIG. For details see the *Command-Line Reference*.

NOTE: The new project or stream will not have references to deployment/build areas. Therefore, you need to edit the build configuration and attach areas after setting up the project/area relationships.

Tips

Here are a few build configuration tips.

- When you create a new build configuration, it is automatically checked out. You must check it in before running it.
- If you Undo Checkout on a build configuration that has not been checked in, all details will be lost without warning.
- When the build configuration traffic light symbol shows a yellow light, it means that the build is in progress.

Chapter 6

Executing Builds in the Build Administration Console

About Executing Builds 134
Running a Build Configuration 134
Running a Scheduled Build 136
Running a Non-Scheduled Build from the Build Scheduling Tab 136

About Executing Builds

This section describes how to run builds in the Build Administration Console.

For information on running builds in the web and desktop clients, see the *User's Guide* and the help for those clients.

For information on running builds in the command-line client see the *Command Line Reference*.

For information on running builds in the ISPF client see the *Dimensions for z/OS User's Guide* and the ISPF panel help.

You can run builds in the following ways in the Build Administration Console:

- Run a build configuration.
- Run a build using a scheduled start.
- Run a non-scheduled build from the Build Scheduling tab.
- Use the Launch new build button from the Build execution statuses dialog box of an executed build.

Running a Build Configuration

Follow this procedure to run a build configuration from the Build Administration Console.

To run a build configuration:

1 In the navigation pane of the Build Management tab of the Build Administration Console, expand Dimensions Projects and select the project containing the build configuration that you want to build.

2 Select the build configuration and on the toolbar click Run.

The Run Build wizard appears.

3 From the **Dimensions Project** list accept the default project/stream or select the project containing the configuration that you want to build.

4 (Only displayed if one or more baselines have been created from the project or stream) To build a baseline, select one from the list. If you do not want to build a baseline select <none>.

5 From the **Build Configuration** list select a build configuration for the project or stream.

6 From the **Version** list accept the default version of the build configuration (Latest) or select an earlier version.

7 From the **Build Stage** list select the deployment stage at which the targets will be built. To build in a work area select <none>. You can only build a baseline in a work area.

8 From the **Build Area** list select the build or work area where the targets will be built.

9 If you are building in a work area do the following:

- From the **Start search path from stage** list select the stage where you want the search path to start.
- (Option not displayed for baseline builds) To download the files to the local work area before the build is executed select **Download files to work area before each build**.
- To apply the system date and time to files that are downloaded, select the option **Apply system date/time to downloaded files**.

10 Click Next.

11 To capture the build outputs and check them into Dimensions CM select the option **Capture build outputs**.

12 If you are checking in build outputs do the following:

- To check the build outputs into a different project or stream, from the **Select Dimensions Project for output preservation** list select a project or stream.
- To specify the requests that the build outputs will be related to when they are checked into Dimensions, in the **Provide Change Documents IDs** field enter the request IDs in the following format (separate request IDs with a semicolon):

  ```
  "QLARIUS_CR_44";"QLARIUS_CR_43"
  ```

13 Click Next.

14 On the Options page select build options:

- **Audit area(s) before build**: (not available for baseline builds) produces an audit report for the build areas associated with the Dimensions project/stream. The report is produced before the build.
- **Execute clean-up script before build**: runs the clean script (in the build configuration).
- **Lock area(s) in search path**: locks all deployment areas associated with the build and prevents other changes to these areas while the build is running. Only applies to work area builds.
- **Rebuild all targets**: rebuilds all targets. If you select this option the following option is also available:
  - **which are in the current stage**: only builds targets whose source is at the current stage.

15 Enter additional options in the **Other build options** field. For details about build options see *The Templating Language and Processor* chapter of the *Developer's Reference*.

16 Click Next.

17 On the Targets page select All targets or the specific targets that you want to build.

18 Click Next.

A summary of the build command that will be executed is displayed.

19 Click Run. If prompted enter the ID of the Dimensions user who owns the build area. You also need to enter and confirm their password.

The Build execution statuses dialog box appears. For details about evaluating the success or failure of a build, see Monitoring Builds in the Build Administration Console on page 157.

Running a Scheduled Build

You can schedule a build to run in the future, for details see Scheduling Dimensions Build Jobs on page 147.

Running a Non-Scheduled Build from the Build Scheduling Tab

You can run a build from the Build Scheduling tab without having to schedule it.

1. In the navigation pane select the build you want to run.
2. Click Run.

   The Run Build wizard appears.
3. Complete the Run Build wizard as described in "Running a Build Configuration" on page 134.

If you select a specific version of a build, the build must have been executed at least once or you may get an error referring to build areas.

Chapter 7

Using Filters

About Using Filters	138
What Can Be Filtered	138
Filtering Build Jobs by User Name	138
Filtering Build Jobs by Date and Time	139
Filtering Dimensions Projects	140
Editing Filter Conditions	141

About Using Filters

This section describes how to use filters in Dimensions CM.

Filters define criteria that the data must match in order to be displayed. A photographic filter can limit the wavelengths of light visible to a camera; a filter in Dimensions CM can limit the build projects or build jobs visible to the user.

You can also use filters to specify which build projects or build jobs should be hidden, instead of visible. For example, you can use a filter to limit the display of past builds, or to exclude all projects other than the one you are working on.

These tasks are discussed in this chapter.

What Can Be Filtered

Dimensions CM allows you to filter either build projects, or build jobs.

- On the Build Management tab: Dimensions projects
- On the Build Scheduling tab: Dimensions projects
- On the Build Monitoring tab: Build jobs (running jobs and History)
- On the Notifications tab: Dimensions projects

Filtering Build Jobs by User Name

You can limit the displayed build jobs to a specific user name by setting a user name filter.

To set a build job user name filter:

1 Select the Build Monitoring tab.

2 In the navigation pane select Running Jobs.

3 On the toolbar click Filter.

The Set Build Job Filter dialog box appears.

4 Click Add.

The Add Build Job Filter Condition dialog box appears.

5 Define the filter condition using the following guidelines:

- If you want to view your own build jobs, click the check box labeled Current User.
- If you want to view build jobs by another user, enter the name of that user.

NOTE The filter defines an exact match. Entering a string such as "Smith" will match "Smith" only—not "Smithson" or "Smithee". To match multiple strings, use the asterisk (*) as a wildcard—in this example, you would enter "Smith*".

- If you want to exclude build jobs by yourself or by another user, select Exclusive from the Inclusion Type field.
- To match (or exclude) multiple user names, enter multiple conditions. You can only specify one name per condition.

6 When you have finished defining the filter condition, click Apply Condition. The filter condition appears in the Filter Conditions list box.

7 Click OK to accept the Add Build Job Filter Condition dialog box. The Active Jobs heading changes to indicate that the view is filtered.

Filtering Build Jobs by Date and Time

You can limit the displayed build jobs to a specific date and time range by setting a range filter.

To set a build job date and time range filter:

1 Select the Build Monitoring tab.

2 In the navigation select the name of the project whose build jobs you want to filter.

3 On the toolbar click Filter.

The Set Build Job Filter dialog box appears.

4 Click Add.

The Add Build Job Filter Condition dialog box appears.

5 In the Property field, select **Range**. The date and time fields appear.

6 Define the filter condition using the following guidelines:

- To match (or exclude) build jobs from the last N days, where N is an integer, enter a value in the Last day(s) field.
- To match (or exclude) build jobs since a certain Start Time, click the down arrow and select the date and time.
- To match (or exclude) build jobs within a certain range, click the down arrow next to Start and select a starting date and time, then click the down arrow to the right of End by and select an ending date and time.

7 When you have finished defining the filter condition, click Apply Condition. The filter condition appears in the Filter Conditions list box.

8 Click OK to accept the Add Build Job Filter Condition dialog box. The Active Jobs heading changes to indicate that the view is filtered.

Filtering Dimensions Projects

You can limit the displayed Dimensions projects with a filter.

To set a Dimensions project filter:

1. In the navigation pane select the highest node that uses filters. For example, on the Build Management tab select Dimensions Projects.
2. On the toolbar click Filter.

   The Set Dimensions Project Filter dialog box appears.
3. Click Add.

   The Add Dimensions Project Filter Condition dialog box appears.
4. Enter the name of the project you wish to search for (or exclude).

   NOTE The filter defines an exact match. Entering a string such as "Project" will match "Project" only—not "Project1" or "New_Project". To match multiple strings, use the asterisk (*) as a wildcard—in this example, you would enter "*Project*".
5. If you wish to exclude the named project, click the Exclusive radio button.
6. When you have finished defining the filter condition, click Apply Condition. The filter condition appears in the Filter Conditions list box.
7. Click OK to accept the Add Dimensions Project Filter Condition dialog box. The Dimensions Projects heading changes to indicate that the view is filtered.

Editing Filter Conditions

Editing filter conditions includes any of the following:

- Altering the definition of the filter
- Removing specific filter conditions
- Removing all filter conditions

Altering the Definition of a Filter Condition

To alter the definition of a filter condition:

1 Click the Filter icon. The filter dialog box appears.

2 Select the specific filter condition you wish to alter. You can only select one condition at a time.

3 Make the desired changes to the filter condition, then click Apply Condition. The altered filter condition appears in the list of Filter Conditions.

4 Click OK to accept the dialog box.

Removing Specific Filter Conditions

To remove specific filter conditions:

1 Click the Filter icon. The filter dialog box appears.

2 Select the specific filter condition you wish to remove. You can only select one condition at a time.

3 Click Remove. Dimensions CM deletes the selected filter condition.

4 Repeat until all of the filter conditions you wish to delete are gone.

5 Click OK to accept the dialog box.

Removing All Filter Conditions

To remove all filter conditions:

1 Click the Filter icon. The filter dialog box appears.

2 Click Remove All. All of the filter conditions disappear.

3 Click OK to accept the dialog box.

Chapter 8

Versioning

Versioning	143
Example of Rollback	144
Example of Rewrite	146
Deleting Versions Not Allowed	146

About Versioning

This section describes how to work with different versions of your build configurations.

The three main things you can do are:

- Rollback

  If you select an existing version of a build configuration and then click the Rollback icon, Dimensions CM will create a new version identical to the selected version, and check it in with a new version number. Use this feature to return to a previous version.

- Rewrite

  If you select an existing version of a build project and then click the Rewrite icon, Dimensions CM will rewrite the current build configuration so that it is identical to the selected build configuration. The current build configuration is NOT checked in. Use this feature to return to an existing build configuration so that you can create a variation on it.

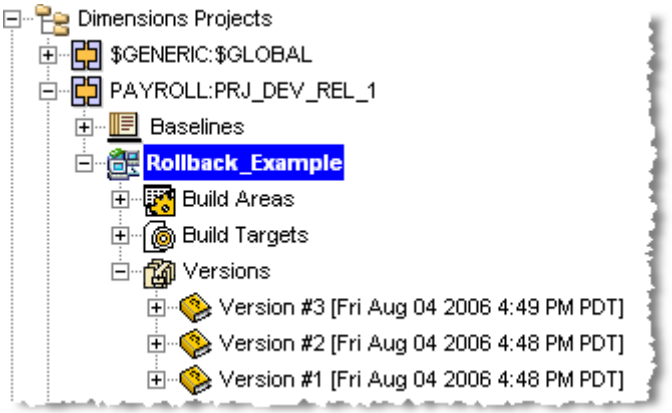
  NOTE You can only Rollback or Rewrite from the Build Management tab. You cannot perform these operations from the other tabs.

- Copy a version and save it as another build configuration

  If you select an existing version of a build configuration and then click the Copy icon, you can save the copied version under a new name. See Copying Build Configurations on page 100.

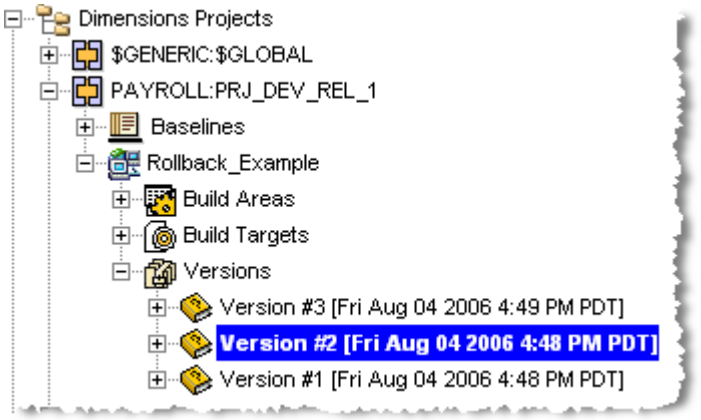
Example of Rollback

Consider a build configuration with several existing versions:



To roll back to a previous version:

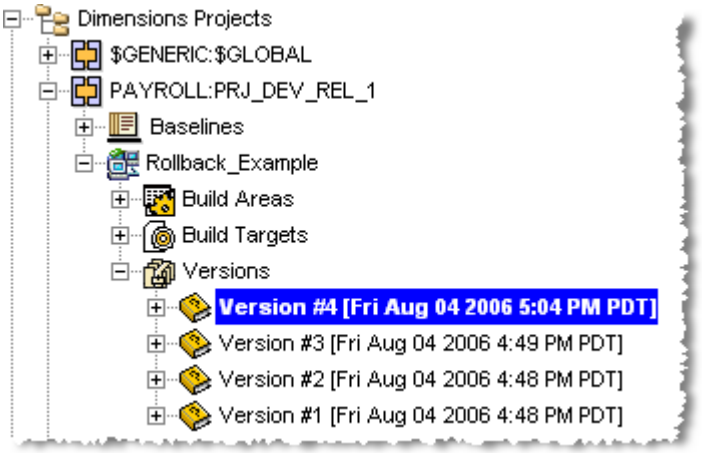
1 On the Build Management tab select the version of the build configuration that you want to rollback (in this example, Version #2).



2 On the toolbar click Rollback.

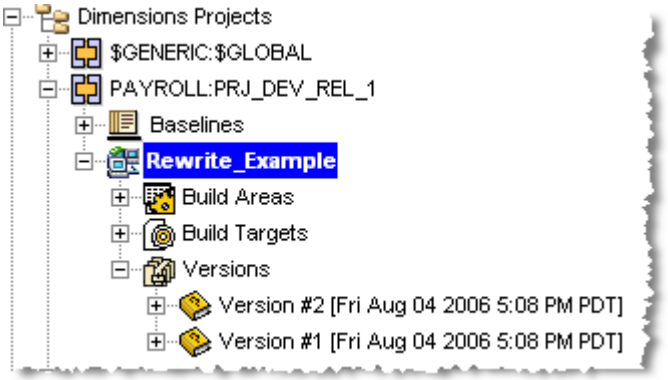
The Rollback dialog box appears.

3 Enter a comment (as shown above), then click OK. A dialog box briefly confirms the rollback operation, and a new version appears in Dimensions CM:



Example of Rewrite

Consider a build configuration with two existing versions:



To Rewrite to a previous version:

1 Select the version of the build configuration that you want to rewrite (in this example, Version #1).

2 On the toolbar click Rewrite.

A warning dialog box appears:

3 Click OK to confirm the Rewrite operation. Dimensions CM replaces the current build configuration with the selected build configuration.

Deleting Versions Not Allowed

You are not allowed to delete a version of a build configuration.

Chapter 9

Scheduling Dimensions Build Jobs

About Scheduling	148
Managing the Scheduler Service	148
Filtering Dimensions Projects	149
Managing Build Job Schedules	149
Checking the Results of Scheduled Build Jobs	156
Running Builds from the Build Scheduling Tab	156

About Scheduling

This chapter describes how to set up schedules that execute build jobs in the future. You can schedule a build to execute once or to repeat at specific intervals.

When you create a build job schedule you specify the following parameters:

- The build configuration to be executed.
- The build area to be populated.
- The targets to be built.
- When and how often the build job schedule will repeat.

Before you add a build job schedule you must first:

- Create a build configurations, for details see page 95.
- Configure the Dimensions Build Scheduler Service, for details see below.

Managing the Scheduler Service

This section describes how to configure and modify the Dimensions Build Scheduler Service and to verify that the service is running. You must configure the Scheduler Service to enable scheduled build jobs to run.

Configuring the Scheduler Service

Purpose Follow this procedure to configure the Scheduler Service. You can also use this procedure to modify the current settings of the service.

To configure the Scheduler Service:

1. In Dimensions Build click the Build Scheduling tab. In the navigation pane click Scheduling.
2. In the menu area click Scheduler.

   The Scheduler Service Configuration dialog box appears.
3. Type the required credentials. Use the same values as the one that you used to log in to the Administration Console. You can also specify the credentials of another user but you must verify that they have sufficient privileges to run a service.
4. To restart the Scheduler Service after applying the new or modified credentials, select the **Start after update** check box.
5. Click OK.

IMPORTANT!

The Scheduler Service does not restart if Tomcat is restarted

If you restart the Tomcat server the service that controls scheduling does not restart. To automatically restart the scheduling service every time that Tomcat restarts, edit the `scheduler.start` variable in the following file:

- Windows:
  `%DM_ROOT%\..\Common Tools\tomcat\8.0\webapps\bws\WEB-INF\web.xml`
- UNIX: `$DM_ROOT/../common/tomcat/8.0/webapps/bws/WEB-INF/web.xml`

Edit the `param-value` of the `scheduler.start` variable so that it is set to "1" or "true" and check that this init-param is uncommented.

If you are using a remote build server Dimensions Build may not be able to distinguish between multiple IP addresses. Specify the following variables in the file web.xml:

- The hostname in `build.server.host`
- The port number in `build.server.port`

Filtering Dimensions Projects

You can use a filter to limit the Dimensions projects that are displayed in the navigation pane. This feature is useful when you have many Dimensions projects and you only want to display a sub-set of them. For details about using filters see page 137.

Managing Build Job Schedules

This section describes how to manage build job schedules.

Viewing Scheduled Build Jobs

Purpose Follow this procedure to view the build jobs that are currently scheduled.

To view scheduled build jobs:

1 In Dimensions Build click the Build Scheduling tab.

2 To view the build jobs scheduled for a Dimensions project, in the navigation pane click Scheduling and click the Dimensions project. The content pane lists all the build jobs scheduled for the Dimensions project.

3 To view the build jobs scheduled for a build configuration, in the navigation pane click the build configuration. The content pane lists all the build jobs scheduled for the build configuration.

Adding Build Job Schedules

Purpose Follow this procedure to add a new build job schedule.

NOTE Before you add a build job schedule you must first create a build configuration and configure the Scheduler Service.

To add a build job schedule:

1 In Dimensions Build click the Build Scheduling tab.

2 In the navigation pane click Scheduling and select the Dimensions project containing the build configuration where you want to add a schedule.

3 In the menu area click **Schedule** or in the Scheduled Build Jobs section of the content pane click **New**.

The Add Scheduled Build Job dialog box appears.

4 From the **Dimensions Project** list select a Dimensions project.

5 From the **Build Configuration** list select a build configuration. To build all configurations select *All configurations*.

6 From the **Version** list select the version of the build configuration that you want to build. If you are building all configurations this option is set to *Latest* and you cannot change it.

7 From the **Build Area** list select the build area where you want the build configuration to be built. To build all build areas associated with the build configuration select *All areas*. If you are building all configurations this option is set to *All areas* and you cannot change it.

When you setup the build configuration and attached a build area, if you specified that a password is required at runtime, the **Provide passwords for selected areas** link appears to the right of the Build Area list.

To enable access by the users to the build areas when the scheduled job is run, you must provide a password for each user. Do the following:

a Click **Provide passwords for selected areas**. The Set Build Password(s) dialog box appears.

b Type and confirm a password for each user.

c Click Save.

8 From the **Build Target** list select the build target to be built. To build all build targets for the build configuration select *All targets*. If you are building all configurations this option is set to *All targets* and you cannot change it.

9 To specify a start time for the build job schedule do the following:

   a Click the down arrow to the right of the Start Time field.

   The calendar picker appears.

   b From the **Month** list select a month, or use the left and right arrows to scroll through the months.

   c From the **Year** list select a year.

   d Select a date. Dates with a light grey background occur on Saturdays and Sundays. Dates with a dark grey background occur in different months and you cannot select them (choose the next or previous month).

   e From the **Time** lists select an hour, minute, and PM or AM.

   f To use this start time click the tick button at the top right corner of the calendar picker.

   The start time appears in the Start Time field.

10 From the **Time Out** list optionally specify the minutes, hours, or days after which the build job is terminated if it has not finished.

11 Select the **Execute clean-up script before build** check box to have the clean-up script defined in the build configuration executed before the build begins (all files in the build area are removed before the build begins).

12 Select the **Do not transfer sources** check box if you do not want copies of the source files retrieved to the build area.

13 Select the **Do not preserve targets** check box if you do not want the target files to be preserved in Dimensions.

14 (If you are preserving targets) From the **Select Dimensions Project for targets preservation** list select the Dimensions project that will receive the target files produced by the build. This list is not displayed if you select the **Do not preserve targets** check box.

15 If this is a one-time schedule for a future build job that you do not want to repeat, click OK. The new build job schedule is added to the content pane.

   NOTE If passwords are required at runtime for the build areas but you have not specified them (see step 7), you will be prompted for the passwords.

   To set the schedule to recur see below.

Scheduling the Frequency and Range of Recurring Build Jobs

Purpose Follow the procedure below to set the frequency and range that a new build job is repeated.

Pre-requisites Setup a build job schedule, for details see page 149.

To schedule the frequency of a recurring build job:

1. If you are adding a recurrence to a new schedule in the Add Scheduled Build Job dialog box, go to step 2.

   If you are adding a recurrence to a new schedule but have closed the Add Scheduled Build Job dialog box, do the following:

   a. In the navigation pane click Scheduling and select the Dimensions project containing the build job schedule.

   b. In the Scheduled Build Jobs section of the content pane click the scheduled build job. The Edit Scheduled Build Job dialog box appears.

2. On the **Scheduled Build Job Details** tab select the **Recurring** check box and click the **Recurrence** tab.

3. To specify the frequency that the build job is repeated, select one of the following options:

   - **Hourly**: specify an interval in hours, for example, every 2 hours.
   - **Daily**: specify an interval in days, for example, every 3 days.
   - **Weekly**: specify an interval in weeks and select the day(s) that the build job will repeat. For example, you can specify that the build job will repeat every week on Monday and Friday.
   - **Monthly**: specify the day and the month that the build job will repeat, for example day 16 of month 12.

4. Click the **Range of Recurrence** tab.

5. To specify when the build job recurrence will start, click the down arrow to the right of the Start field. The calendar picker appears. To select a start time and date do the following:

   a. From the **Month** list select a month, or use the left and right arrows to scroll through the months.

   b. From the **Year** list select a year.

   c. Select a date. Dates with a light grey background occur on Saturdays and Sundays. Dates with a dark grey background occur in different months and you cannot select them (choose the next or previous month).

   d. From the **Time** lists select an hour, minute, and PM or AM.

   e. To use this start time click the tick button at the top right corner of the calendar picker. The start time appears in the Start Time field.

6 To specify when the build job recurrence will end, choose one of the following options:

- **No end date**: the build job recurs indefinitely until you modify or delete the build job schedule.
- **End after *n* occurrences**: specify the number of times the build job will be repeated.
- **End by**: select the date that the build job schedule will stop. Click the down arrow to the right of the field and use the calendar picker to select a date (see steps 5a to 5e).

7 Click OK.

Modifying Build Job Schedules

Purpose Follow this procedure to modify the parameters of an existing build job schedule.

To modify a build job schedule:

1 In Dimensions Build click the Build Scheduling tab.

2 In the navigation pane click Scheduling and select the Dimensions project containing the build job schedule that you want to modify. In the Scheduled Build Jobs section of the content pane, in the Scheduled Time column, click the schedule. The Edit Scheduled Build Job dialog box appears.

3 From the **Dimensions Project** list select a Dimensions project.

4 From the **Build Configuration** list select a build configuration. To build all configurations select *All configurations*.

5 From the **Version** list select the version of the build configuration that you want to build. If you are building all configurations this option is set to *Latest* and you cannot change it.

6 From the **Build Area** list select the build area where you want the build configuration to be built. To build all build areas associated with the build configuration select *All areas*. If you are building all configurations this option is set to *All areas* and you cannot change it.

When you attached a build area to the build configuration, if you specified that a password is required at runtime, the **Provide passwords for selected areas** link appears to the right of the Build Area list.

To enable access by the users to the build areas when the scheduled job is run in the future, you must provide a password for each user. Do the following:

a Click **Provide passwords for selected areas**. The Set Build Password(s) dialog box appears.

b Type and confirm a password for each user.

c Click Save.

7 From the **Build Target** list select the build target to be built. To build all build targets for the build configuration select *All targets*. If you are building all configurations this option is set to *All targets* and you cannot change it.

8 To modify the start time for the build job do the following:

a Click the down arrow to the right of the Start Time field.

The calendar picker appears.

b From the **Month** list select a month, or use the left and right arrows to scroll through the months.

c From the **Year** list select a year.

d Select a date. Dates with a light grey background occur on Saturdays and Sundays. Dates with a dark grey background occur in different months and you cannot select them (choose the next or previous month).

e From the **Time** lists select an hour, minute, and PM or AM.

f To use this start time click the tick button at the top right corner of the calendar picker.

The start time appears in the Start Time field.

9 From the **Time Out** list optionally specify the minutes, hours, or days after which the build job is terminated if it has not finished.

10 Select the **Execute clean-up script before build** check box to have the clean-up script defined in the build configuration executed before the build begins (all files in the build area are removed before the build begins).

11 Select the **Do not transfer sources** check box if you do not want copies of the source files retrieved to the build area.

12 Select the **Do not preserve targets** check box if you do not want the target files to be preserved in Dimensions.

13 (If you are preserving targets) From the **Select Dimensions Project for targets preservation** list select the Dimensions project that will receive the target files produced by the build. This list is not displayed if you select the **Do not preserve targets** check box.

14 Click OK.

NOTE If passwords are required at runtime for the build areas but you have not specified them (see step 6), you will be prompted for the passwords.

Modifying the Frequency and Range of Recurring Build Jobs

Purpose Follow the procedure below to modify the frequency and range that a build job is repeated.

To modify the frequency of a recurring build job:

1 In Dimensions Build click the Build Scheduling tab.

2 In the navigation pane click Scheduling and select the Dimensions project containing the build schedule recurrence that you want to modify. In the Scheduled Build Jobs section of the content pane click the schedule. The Edit Scheduled Build Job dialog box appears.

3 Click the **Recurrence** tab.

4 To modify the frequency that the build job is repeated select one of the following options:

- **Hourly**: specify an interval in hours, for example, every 2 hours.
- **Daily**: specify an interval in days, for example, every 3 days.
- **Weekly**: specify an interval in weeks and select the day(s) that the build job will repeat. For example, you can specify that the build job schedule will repeat every week on Monday and Friday.
- **Monthly**: specify the day and the month that the build job will repeat, for example day 16 of month 12.

5 Click the **Range of Recurrence** tab.

6 To modify the start time of the build job recurrence, click the down arrow to the right of the Start field. The calendar picker appears. To select a start time and date do the following:

a From the **Month** list select a month, or use the left and right arrows to scroll through the months.

b From the **Year** list select a year.

c Select a date. Dates with a light grey background occur on Saturdays and Sundays. Dates with a dark grey background occur in different months and you cannot select them (choose the next or previous month).

d From the **Time** lists select an hour, minute, and PM or AM.

e To use this start time click the tick button at the top right corner of the calendar picker. The start time appears in the Start Time field.

7 To modify when the build job recurrence will end, choose one of the following options:

- **No end date**: the build job recurs indefinitely until you modify or delete the build job schedule.
- **End after *n* occurrences**: specify the number of times the build job will be repeated.
- **End by**: select the date that the build job will stop. Click the down arrow to the right of the field and use the calendar picker to select a date (see steps 6a to 6e).

8 Click OK.

Deleting Build Job Schedules

Purpose Follow this procedure to delete build job schedules.

To modify build job schedules:

1. In Dimensions Build click the Build Scheduling tab.
2. In the navigation pane click Scheduling and click the Dimensions project containing the build job schedules that you want to delete.
3. In the content pane select the build job schedules and click Delete.
4. Click Yes to confirm that you want to delete the build job schedules.

Checking the Results of Scheduled Build Jobs

To check the progress and results of scheduled build jobs use the Build Monitoring tab. For details see Chapter 10, "Monitoring Builds in the Build Administration Console" on page 157.

Running Builds from the Build Scheduling Tab

You normally run builds from the Build Management tab. However you can also run builds from the Build Scheduling tab.

NOTE To run a build configuration you must first check it in.

To run a build from the Build Scheduling tab:

1. In the navigation pane of the Build Scheduling tab click Scheduling and select the Dimensions project containing the build configuration that you want to run.
2. On the menu area click Run. The Build wizard appears. For details about using the wizard see page 134.
3. Click Run. You can monitor the progress of the build in the Build Monitoring tab.

Chapter 10

Monitoring Builds in the Build Administration Console

About Monitoring Builds	158
Reviewing Build Status	158
Monitoring a Running Build	163
Canceling a Running Build Job	164
Viewing Build Execution History	164
Viewing Dependencies	165

About Monitoring Builds

This section describes how to use the Build Monitoring features of Dimensions CM. These features allow you to examine details such as:

- what time the build started
- what time the build completed
- whether or not there was an error
- what Dimensions project and build configuration were used
- the dependencies of the build job
- Bill of Materials information
- who initiated the build

In addition to these features, Dimensions CM also allows you to rebuild previous build jobs, and to cancel running build jobs. These features are discussed after the basic build monitoring features.

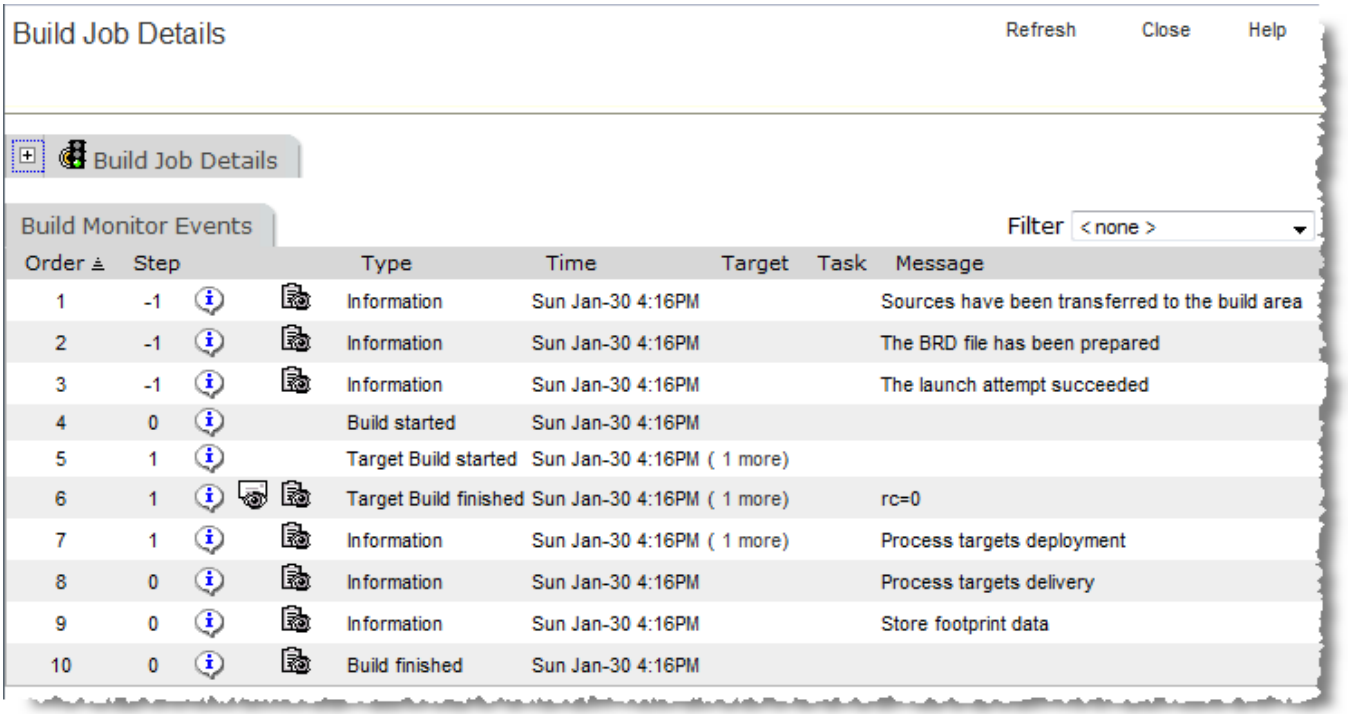
In general, build monitoring falls into three categories:

- Reviewing build status immediately after launch
- Monitoring the status of a running build
- Reviewing the history of previous builds

Reviewing Build Status

After you launch a build the **Build execution statuses** dialog box appears.

The **Build Job Details** dialog box appears shortly afterwards and displays information such as the start time, the build configuration that was used, and the build monitor events.



By default, the Build Job Details section is hidden and the Build Monitor Events section is displayed.

If the Build Job Details dialog box does not appear, or you close it, click the **See Build Event**s icon in the Monitor column in the Build Execution Statuses dialog box.

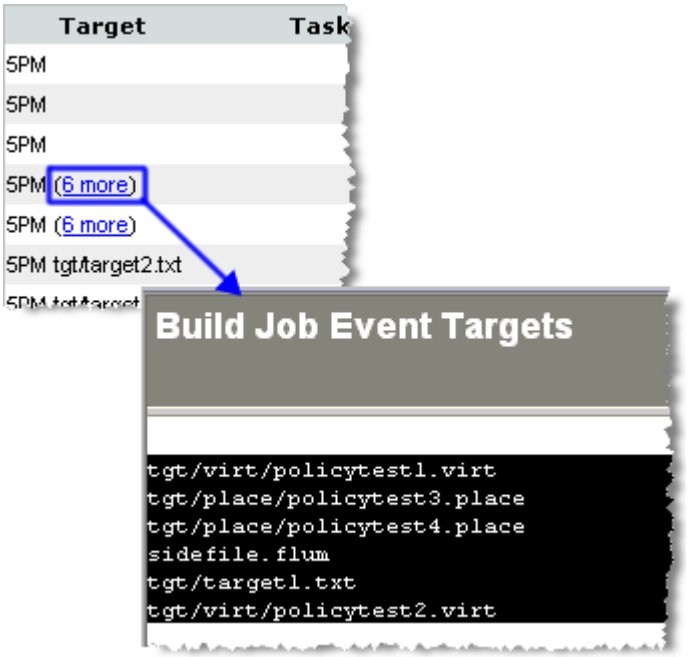
NOTE On mainframe machines, build jobs do not always execute immediately and may be queued. Dimensions CM will wait a certain period of time for a response from the remote node executing the job. If the job execution occurs after this amount of time, Dimensions CM is unable to display any information in the Build Execution Statuses dialog box.

Interpreting Build Monitor Events

The Build Monitor Events section displays a table of events that occurred during the build. Each row represents an event. Earlier events are displayed at the top of the table; later events at the bottom.

Build events typically show events such as when the build started, when the build of specific targets started and finished, and errors that occurred during specific steps.

If a build step produces multiple targets, and the length of their names is equal to or more than 25 characters, a link called '*n* more' is displayed in the Target column. Click this link to open the Build Job Event Targets dialog box and display the names of all the targets.

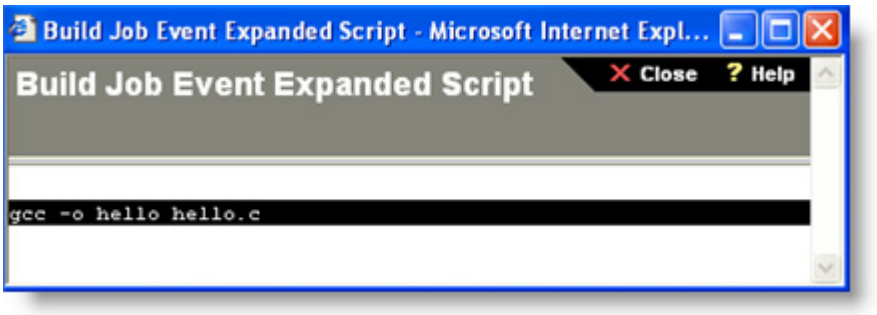


Specific events may display one of the following icons:

Icon	Description
	Display the Build Job Event Expanded Script dialog box. This displays a command-prompt-type display with the script for the build step in question.
	Display the Build Job Event Output Log dialog box. This log displays information such as file deployment statuses, the success of the step, and so on.
	Display the Build Job Event Error Log dialog box. This log displays error information for build steps that fail.

Expanded Script

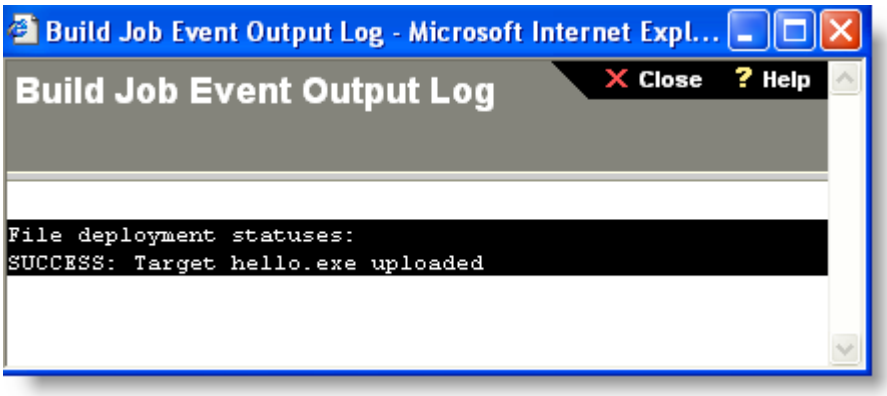
This is an example of the expanded script:



The expanded script displays the build execution command found in the build scripts.

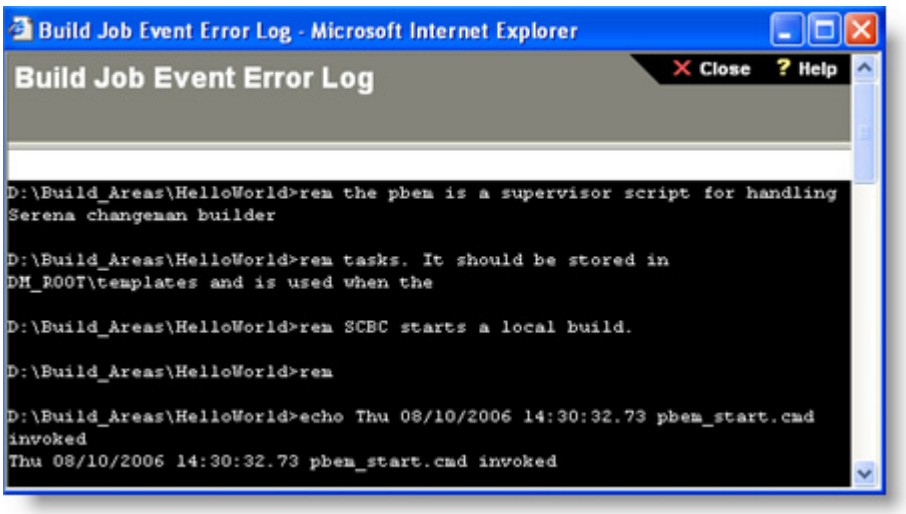
Build Job Event Output Log

This is an example of the build job event output log.



Error Log

This is an example of the error log:

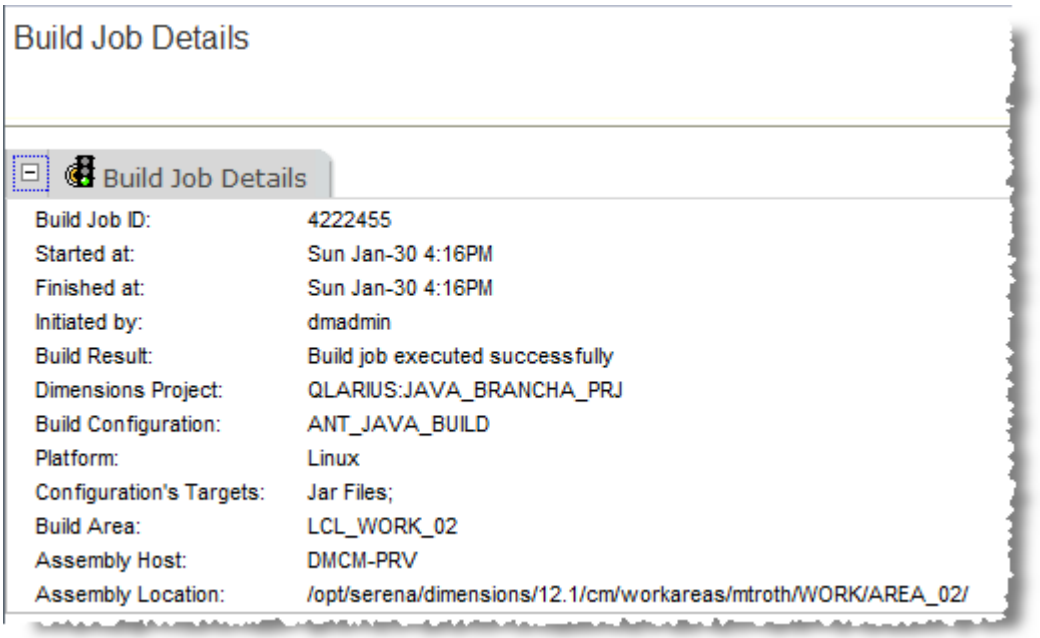


In the error log you can find information on which tasks failed, for example:

```
16PB0011I Task 1 executed with errc = 1 - failed
16PB0044I Processing rc: 1; Highest Significant Subtask Rc: 1
16PB0045I Of the 1 tasks requested; 0 succeeded, 1 failed, and 0 were
    skipped
```

Viewing Build Details

Expand the Build Job Details section to display all its information:



This section contains information such as who initiated the build, which Dimensions project and build configuration were used, and what platform the build was for.

Monitoring a Running Build

Other sections have described how to look at the details of a build you just launched. If the build is a long-running job, you can also examine its details using the Build Monitoring tab.

To monitor a running build:

1. Log on to the Dimensions CM Administration Console.
2. Click the Build Monitoring tab. In the navigation pane you can see the tree with Running Jobs and History entries.
3. Click Running Jobs. The content pane displays a list of the active (currently running) build jobs:

NOTE If instead of clicking Running Jobs you click the plus sign to the left of Running Jobs, the list of build jobs appears in the object tree only.

Notice also that the top of the display now shows three icons—Refresh, Cancel, and Filter:

- **Refresh** refreshes the display.
- **Cancel** cancels the selected build jobs. You can cancel more than one build job at a time.
- **Filter** applies criteria that you define to limit what is displayed. Defining a filter is covered in Using Filters on page 137.

4. Click the build job that you are interested in. The Build Job Details and Build Monitor Events appear on the right in the content pane.

Canceling a Running Build Job

You can cancel a running build job from the Build Monitoring tab of the Build Administration Console.

To cancel a running build job:

1. In the Build Administration Console, click the Build Monitoring tab.
2. Click Running Jobs to display the list of running jobs.
3. Select the check box to the left of the build job you want to delete. The Cancel icon turns red.
4. Click the Cancel icon. Dimensions CM displays a confirmation dialog box and then deletes the selected build job.

Monitoring Past Builds

When you want to view the history of build jobs that have been launched in the past, you can use the features available under the Build Monitoring tab.

Viewing Build Execution History

To view the execution history for a past build:

1. In the Build Administration Console click the Build Monitoring tab.
2. Click History to display a list of Dimensions projects.
3. Select the Dimensions project containing the build configuration and then select the configuration. The Execution History tab is displayed in the content pane.
   - The status column shows whether or not the build completed successfully.
   - To see more details of a the build execution click the link in ID column in the Execution History tab.
   - The Build Job Details and Build Monitor Events for that build are displayed in the content pane.

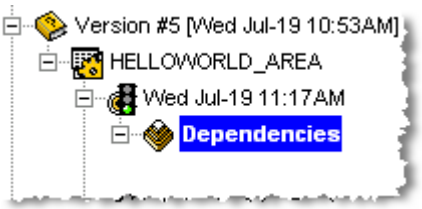
Viewing Dependencies

You can view build dependencies from the Build Monitoring tab. The dependencies show which modules or items were used to create the final targets.

NOTE To display dependencies, you must configure the build job to preserve targets.

To view the dependencies for a specific target:

1. In the Build Administration Console click the Build Monitoring tab.
2. Expand the History tree until you find the versions, and underneath them, the specific build execution (a date and time) you are interested in.
3. Expand the build execution record and the dependencies entries appear:



4. Select the Dependencies entry. The Job Targets and Job Sources listings appear in the content pane.

Job Targets

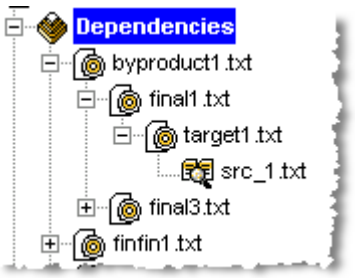
Name	Relative Path	Dimensions Location
byproduct1.txt		QLARIUS:BYPRODUCT1 TXT~01.A-SRC;3
final1.txt		QLARIUS:FINAL1 TXT~01.A-SRC;3
final3.txt		QLARIUS:FINAL3 TXT~01.A-SRC;3
finfin1.txt		QLARIUS:FINFIN1 TXT~01.A-SRC;3
finfin2.txt		QLARIUS:FINFIN2 TXT~01.A-SRC;3
policytest1.virt		QLARIUS:POLICYTEST1 VIRT~02.A-DAT;1.6
policytest3.place		QLARIUS:POLICYTEST3 PLACE~02.A-DAT;1.6
sidefile.flum		QLARIUS:SIDEFILE FLUM~02.A-DAT;1.6
target1.txt		QLARIUS:TARGET1 TXT~02.A-SRC;7
target2.txt		QLARIUS:TARGET2 TXT~02.A-SRC;3
target3.txt		QLARIUS:TARGET3 TXT~02.A-SRC;3

Job Sources

File	Relative Path	Dimensions Location
src_1.txt		QLARIUS:SRC 1 TXT.A-SRC;1
src_2.txt		QLARIUS:SRC 2 TXT.A-SRC;1
src_3.txt		QLARIUS:SRC 3 TXT.A-SRC;1

These listings indicate the targets and sources which the build job depends on.

5 To view the hierarchy of dependencies for a specific file, expand the entries in the Dependencies tree:



Selecting specific entries in hierarchies such as the one shown causes the content pane to display source/target details, and dependent sources and targets.

Automatic Detection of Dependencies

On the mainframe side, Dimensions Build natively supports the Serena mainframe build engine that is installed with Dimensions for z/OS. This utility monitors all I/O by each step to provide detailed dependency analysis.

Chapter 11

Managing Notifications

About Notifications	168
Managing the Notification Service	170
Managing Notification Templates	171
Managing Notification Subscriptions	174

About Notifications

Dimensions Build includes a notification facility that enables you to send email messages containing information about build events. For example, you can notify users when a build is completed or when a build area is updated.

Formatting Notification Templates and Events

The format of the email messages generated by notification events is in HTML. To customize the look of these messages you can insert any HTML formatting tag, such as <br> and <b>, into the message field of your templates and events. You do not have to specify the opening and closing <html> and <body> tags.

TIP To ensure that email message can be read easily, add the line break tag <br> at the end of each line in your message, for example:

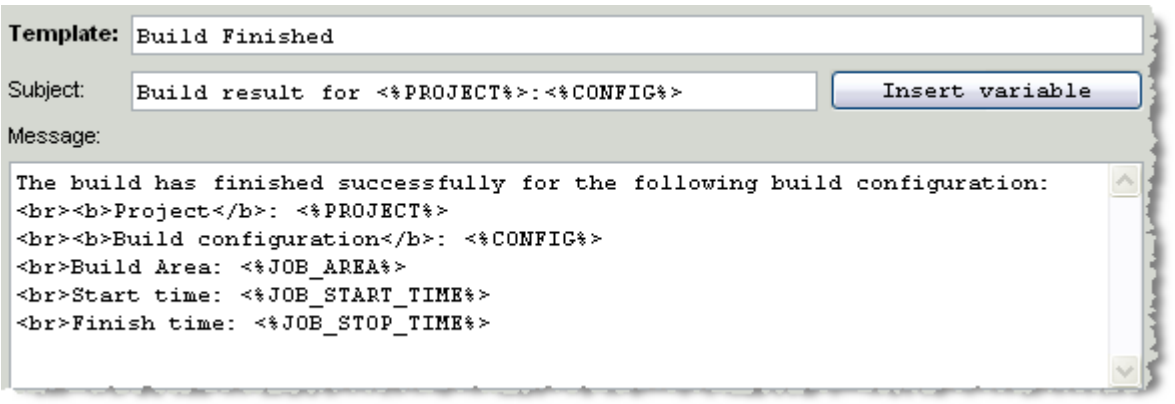
```
Build configuration name <%CONFIG%><br>
Build configuration prescript <%CONFIG_PRESCRIPT%><br>
Build configuration main script <%CONFIG_MAINSCRIPT%><br>
```

Notification Templates

You can create notification templates and use them when you add notification events. Notification templates can include freeform text and any of the pre-defined variables. Templates are useful when you want to re-use the same subject and message in multiple notification events.

In the example below the template includes the following variables:

- <%PROJECT%>: displays the name of the parent Dimensions project.
- <%CONFIG%>: displays the name of the build configuration where the notification event is setup.
- <%JOB_AREA%>: displays the name of the build area.
- <%JOB_START_TIME%>: displays the time that the build job started.
- <%JOB_STOP_TIME%>: displays the time that the build job finished.



Notification Events

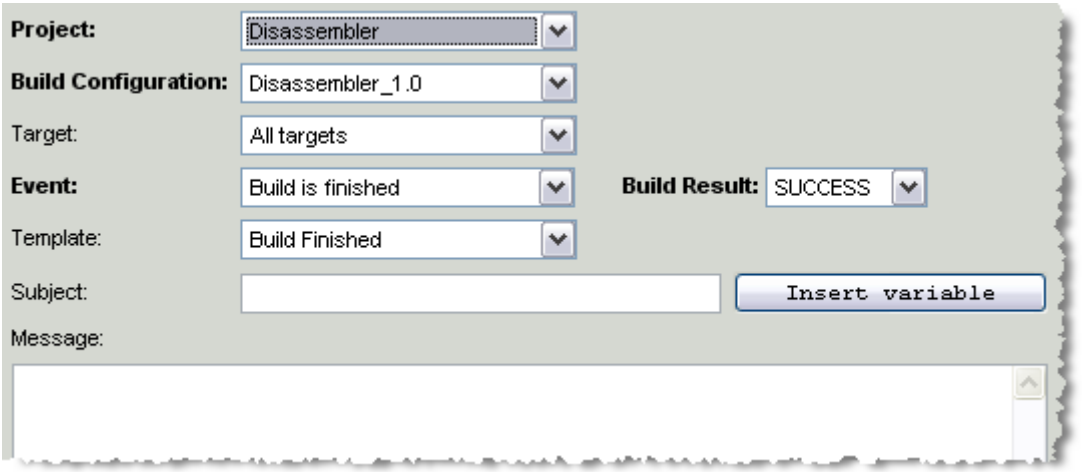
A notification event specifies the action that causes an email message to be sent to the users that are subscribed to it. You can optionally use notification templates to automate and simplify the process of setting up events.

For each notification event you specify the following:

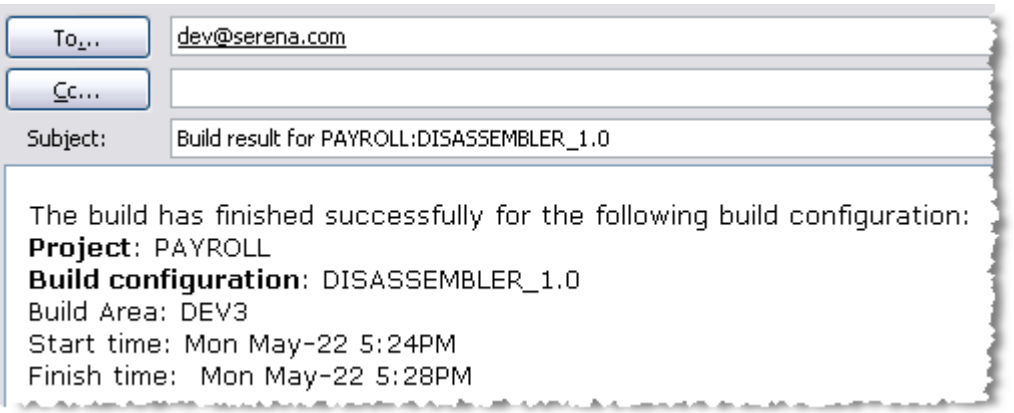
- The Dimensions project, build configuration, and targets to which the event applies. You can specify an individual target or all targets.
- The type of event that triggers the notification. You can choose any of the pre-defined event types, for example:
  - When a build is started.
  - When the targets are built.
  - When there are errors.
- (Optional) The notification template to be used in the email message.
- The subject of the email message. Not required if you are using a template that includes a subject.
- The body of the email message. Not required if you are using a template that includes the message.

Notification Example

In the example below, the notification is sent to the event subscribers after the build of all the targets in *Disassembler_1.0* finishes successfully. The template used is *Build Finished*. No subject and body are required as they are taken from the template:



The email message for this example, using the template *Build Finished* described on page 168, looks like this:



Notification Subscriptions

After you have created a notification event you add subscribers to it. Subscribers can be individual Dimensions users or groups of users. After you add the subscribers they will start to receive email messages based on the notification event criteria that you specified.

Managing the Notification Service

This section describes how to configure and modify the Dimensions Build Notification Service. You must configure the Notification Service to be able to send email notifications.

Configuring the Notification Service

Purpose Follow this procedure to create or modify the parameters required to use an SMTP server to send email notifications.

To configure the Notification Service:

1. In Dimensions Build click the Notifications tab.
2. In the navigation pane click Notifications. In the menu area click **Notification**. The Notification Service Configuration dialog box appears.
3. For **Host** type the name of the node hosting the email server that will send the notifications.
4. For **Port** type the port number that the email server is listening on.
5. For **Hello host** type the string required to establish a connection through the SMTP protocol.
6. For **User** type the name of a user that can access the email server.

7 For **Password** type the password for the user that you specified in the previous step. For **Confirm Password** retype the password.

8 From the **Authentication** list select one of the following SMTP server authentication methods:

- <none>
- LOGIN
- PLAIN
- CRAM-MD5

NOTE If you select <none> you do not need to specify a user and password.

9 For **Sender Mail** type the email address from who the notification emails will be sent.

10 For **Sender Name** type the name that will appear in the From field in the notification emails.

11 For **Available Event Types** ensure the event types are defined, for example: START_BUILD,END_BUILD,MSGE

12 Click OK.

Managing Notification Templates

Viewing Notification Templates

Purpose Follow this procedure to view a list of all the notification templates that are currently defined.

To view notification templates:

1 In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Templates. The navigation and content panes refresh and display all the notification templates that are currently defined.

2 To view the details of a template, select it in the navigation and content panes. The navigation and content panes refresh and display details of the notification template.

Adding Notification Templates

Purpose Follow this procedure to add a new notification template.

To add a notification template:

1 In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Templates. In the menu or content pane click New Object.

The Add New Notification Template dialog box appears.

2 For **Template** type a name for this notification template.

3 For **Subject** type the subject of this notification template. To add a pre-defined variable do the following:

a Place the cursor where you want to add the variable.

b Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.

c From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.

d Click OK.

TIP After you have inserted variables you can add freeform text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

4 For **Message** type the message for this notification template. Add a pre-defined variable if required. Add HTML formatting tags if required.

5 Click OK. The new notification template is added to the content pane.

Editing Notification Templates

Purpose Follow this procedure to edit an existing notification template.

To edit a notification template:

1 In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Templates. In the menu or content pane click the name of the notification template that you want to edit.

2 In the content pane click Edit Notification Template Information.

The Edit Notification Template dialog box appears.

3 For **Template** modify the name of this notification template.

4 For **Subject** modify the subject of this notification template. To add a pre-defined variable do the following:

a Place the cursor where you want to add the variable.

b Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.

c From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.

d Click OK.

TIP After you have inserted variables you can add freeform text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

5 For **Message** modify the message for this notification template. Add pre-defined variables and HTML formatting tags if required.

6 Click OK. The details of the notification template are updated in the content pane.

Copying Notification Templates

Purpose Follow this procedure to copy an existing notification template.

To copy a notification template:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Templates. In the content pane select the check box next to the notification template that you want to copy. Click Copy.

   The Create New Copy of the Notification Template dialog box appears.

2. For **Template** type a name for the new notification template.

3. For **Subject** modify the subject of this notification template. To add a pre-defined variable do the following:

   a Place the cursor where you want to add the variable.

   b Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.

   c From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.

   d Click OK.

   TIP After you have inserted variables you can add freeform text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

4. For **Message** modify the message for this notification template. Add pre-defined variables and HTML formatting tags if required.

5. Click OK. The new notification template is added to the content pane.

Deleting Notification Templates

Purpose Follow this procedure to delete notification templates.

To delete notification templates:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Templates.

2. In the content pane select the check box next to each notification template that you want to delete.

3. Click Delete. To confirm that you want to delete the notification templates click Yes.

Managing Notification Subscriptions

Filtering Dimensions Projects

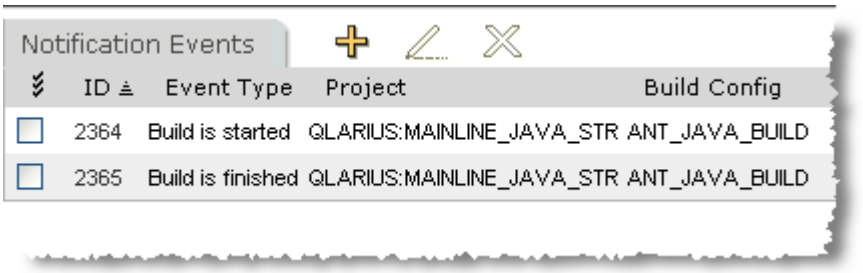
You can use a filter to limit the Dimensions projects that are displayed in the navigation pane. This feature is useful when you have many Dimensions projects and you only want to display a sub-set of them. For details about using filters see page 137.

Viewing Notification Events, Details, and Subscribers

Purpose Follow this procedure to view a list of all the notification events that are currently defined for a Dimensions project. You can also view details of individual notification events including lists of subscribers.

To view notification events, details, and subscribers:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Subscriptions.
2. In the navigation or content panes click the Dimensions project containing the notification events. The content pane refreshes and displays all the notification events that are currently defined for the build configurations in that Dimensions project. For example:



3. To view the notification events defined for a specific build configuration, in the navigation pane expand the Dimensions project and click the build configuration. The content pane refreshes and displays the notification events that are currently defined for that build configuration.
4. To view the details of an individual notification event, in the navigation pane expand the parent build configuration, expand Events, and click the notification event. The content pane refreshes and displays the following information:
   - The details of the notification event.
   - The users and groups that are subscribed to the notification event.

Adding Notification Events

Purpose Follow this procedure to add a new notification event.

NOTE If you do not specify a subject, message, or notification template the notification event is ignored.

To add a notification event:

1 In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Subscriptions. In the navigation or content panes click the Dimensions project containing the build configuration where you want to add a notification event. In the menu or content pane click Create New Notification Event.

The Add New Notification Event dialog box appears.

2 From the **Project** list select the Dimensions project where you want to add a notification event.

3 From the **Build Configuration** list select the build configuration where you want to add a notification event.

4 From the **Targets** list select the target to which the notification event will apply. To trigger a notification for all targets select *All Targets*.

5 From the **Event** list select the type of event that will trigger the notification event. Some events also require that you select an additional option. For example, if you select the *Build is finished* event type you must also select an option from the Build Result list.

6 From the **Template** list optionally select a notification template that will be used to populate the subject and message fields in the email message. If you do not want to use a template select <*none*>.

7 For **Subject** type the subject of this notification event. If you do not specify a subject the template subject is used. To add a pre-defined variable do the following:

a Place the cursor where you want to add the variable.

b Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.

c From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.

d Click OK.

TIP After you have inserted variables you can add freeform text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

8 For **Message** type the message for this notification event. If you do not specify a message the template message is used. Add pre-defined variables and HTML formatting tags if required.

9 Click OK. The notification event is added to the build configuration and displayed in the content pane.

Editing Notification Events

Purpose Follow this procedure to edit an existing notification event.

NOTE If you do not specify a subject, message, or notification template the notification event is ignored.

To edit a notification event:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Subscriptions. In the navigation or content panes click the Dimensions project containing the notification event that you want to edit.
2. Select the check box next to the notification event and click Edit.

   The Edit Notification Event dialog box appears.
3. From the **Project** list change the Dimensions project for the notification event.
4. From the **Build Configuration** list change the build configuration for the notification event.
5. From the **Targets** list change the target to which the notification event will apply. To trigger a notification for all targets select *All Targets*.
6. From the **Event** list change the type of event that triggers the notification event. Some events also require that you select an additional option. For example, if you select the *Build is finished* event type you must also select an option from the Build Result list.
7. From the **Template** list optionally select a notification template that will be used to populate the subject and message fields in the email message. If you do not want to use a template select <*none*>.
8. For **Subject** modify the subject of this notification event. If you do not specify a subject the template subject is used. To add a pre-defined variable do the following:
   - **a** Place the cursor where you want to add the variable.
   - **b** Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.
   - **c** From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.
   - **d** Click OK. The variable is added to the end of the field.

   TIP After you have inserted variables you can add free form text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.
9. For **Message** modify the message for this notification event. If you do not specify a message the template message is used. Add pre-defined variables and HTML formatting tags if required.
10. Click OK. The notification event details are updated in the Dimensions project content pane.

Deleting Notification Events

Purpose Follow this procedure to delete notification events.

To delete notification events:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications and click Subscriptions.
2. In the navigation or content panes click the Dimensions project containing the notification events that you want to delete.
3. In the content pane select the check box next to each notification event that you want to delete.
4. Click Delete. To confirm that you want to delete the notification events click Yes.

Adding Notification Subscriptions

Purpose Follow this procedure to add a new notification subscriber to an existing notification event. You can add Dimensions users or groups of users that have been defined in your process model. For information about adding users and groups see the *Process Configuration Guide* or the Administration Console help.

To add a notification subscription:

1. In Dimensions Build click the Notifications tab. In the navigation pane click Notifications, click Subscriptions, and click the Dimensions project containing the build configuration where you want to add a notification subscription.
2. In the navigation pane click the build configuration, click Events, and select the notification event where you want to add a notification subscription.
3. In the Notification Subscribers section of the content pane click New Object.

   The Add New Notification Subscriber dialog box appears.
4. For **Subscriber Type** select User or Group.
5. From the **Subscriber Name** list select a user or group.
6. For **Subject** type the subject of this notification subscription. If you do not specify a subject the notification event subject is used. To add a pre-defined variable do the following:
   - **a** Place the cursor where you want to add the variable.
   - **b** Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.
   - **c** From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.
   - **d** Click OK. The variable is added to the end of the field.

   TIP After you have inserted variables you can add free form text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

7 For **Message** type the message for this notification subscription. If you do not specify a message the notification event message is used. Add pre-defined variables and HTML formatting tags if required.

8 Click OK. The new notification subscription is added to the content pane.

Editing Notification Subscriptions

Purpose Follow this procedure to edit an existing notification subscriber for a notification event. You can add Dimensions users or groups that have been defined in your process model. For information about adding users and groups see the *Process Configuration Guide* or the Administration Console help.

To edit a notification subscription:

1 In Dimensions Build click the Notifications tab. In the navigation pane click Notifications, click Subscriptions, and click the Dimensions project containing the build configuration where you want to modify an existing notification subscription.

2 In the navigation pane click the build configuration, click Events, and select the notification event containing the notification subscription that you want to edit. In the Notification Subscribers section of the content pane select the check box next to the notification subscription and click Edit.

The Edit Notification Subscriber dialog box appears.

3 For **Subscriber Type** select User or Group.

4 From the **Subscriber Name** list select a user or group.

5 For **Subject** modify the subject of this notification subscription. If you do not specify a subject the notification event subject is used. To add a pre-defined variable do the following:

a Place the cursor where you want to add the variable.

b Click **Insert Variable**. The Insert Notification Variable to the Field dialog box appears.

c From the **Variable Name** field select a variable. The content of the variable is displayed in the Preview field. The **Description** field briefly describes the variable.

d Click OK. The variable is added to the end of the field.

TIP After you have inserted variables you can add free form text anywhere in the field. You can also cut and paste variables if you want to rearrange their order.

6 For **Message** modify the message for this notification subscription. If you do not specify a message the notification event message is used. Add pre-defined variables and HTML formatting tags if required.

7 Click OK. The details of the notification subscription are updated in the content pane.

Deleting Notification Subscriptions

Purpose Follow this procedure to delete notification subscriptions.

To delete notification subscriptions:

1 In Dimensions Build select the Notifications tab.

2 In the navigation pane select Notifications, select Subscriptions, and select the Dimensions project containing the build configuration where you want to delete notification subscriptions.

3 In the navigation pane select the build configuration and select Events.

4 In the content pane select the check box next to each notification event that you want to delete.

5 Click Delete. To confirm that you want to delete the notification events click Yes.

Part 3

Integrating Dimensions Build with Third Party Build Engines

Part 3: Integrating Dimensions Build with Third Party Build Engines contains the following chapters

Using Dimensions CM Build with Ant 183

Chapter 12

Using Dimensions CM Build with Ant

About Using Dimensions Build with Ant	184
Deciding Where to Store the Ant Buildfile	184
Selecting or Creating a Dimensions Project	185
Creating or Selecting Build Areas	185
Importing Ant Buildfiles into a Build Configuration	185
Editing Details of the Imported Targets	186
Editing the Transition Script of an Imported Ant Target	187
Running the Imported Build Job	188

About Using Dimensions Build with Ant

It is possible to integrate your Ant build jobs with Dimensions CM.

The integration between Dimensions CM and Ant enables you to:

- Import Ant buildfiles (`build.xml`) files into Dimensions CM
- Manage your build configurations in Dimensions CM
- Execute Ant tasks from Dimensions CM

Overview of the steps to import your Ant buildfiles to Dimensions CM.

1. Decide whether or not you want to store your Ant buildfile in Dimensions as a controlled file.
2. Decide which Dimensions project should contain your Ant build configuration, or create a new project.
3. In Dimensions, create any work areas or deployment areas needed to duplicate the directory structure used in your Ant build configuration.
4. In Dimensions CM, create a new build configuration to receive the imported Ant build targets.
5. Use the Import Wizard to import the Ant buildfile.
6. Edit the imported targets as necessary.
7. Check in the build configuration.
8. Check in the Ant buildfile.

Deciding Where to Store the Ant Buildfile

The Ant buildfile is required to run the Ant build jobs, so you should decide whether or not to store your Ant buildfile in Dimensions before beginning the import process.

For an informal set of build tasks, you have the option of keeping the Ant buildfile in the build area. For any work that requires version and access control, you should use Dimensions to make the buildfile a controlled file.

Selecting or Creating a Dimensions Project

It is necessary to have a Dimensions project as Dimensions CM will require this for the creation of a build configuration. You should select or create a Dimensions project to receive the targets defined in the Ant buildfile.

If you do not already have source files archived in Dimensions, use the structure defined in the Ant buildfile to set up your source directories under the project.

See the *Serena Dimensions CM User's Guide* for instructions on creating a Dimensions project.

Creating or Selecting Build Areas

You should have a build area ready before importing the Ant buildfile. The base directory of the Ant buildfile should correspond to a Dimensions work area.

Also consider using Dimensions deployment areas for some or all of the directories. For example, if the buildfile refers to a `\dist` directory, that would be a good candidate for a deployment area.

If it is necessary to create a new build area, remember to attach the work area. The procedure for doing this is described in Build Areas on page 102.

Importing Ant Buildfiles into a Build Configuration

Purpose Follow this procedure to import an Ant buildfile into Dimensions CM. After the buildfile is successfully imported, the Ant targets are added to the build configuration.

To import an Ant buildfile into a build configuration:

1. In Dimensions Build, select Build Management.
2. Create a new build configuration to receive the imported Ant buildfile.
3. In the content pane, in the Build Targets section, click Launch Import Wizard.

   The Import Wizard appears. On the left side of the wizard there are links to four pages:

   - Select Import Type
   - Choose File
   - Parameters
   - Confirm

   By default, the Select Import Type page is displayed.

4 On the Select Import Type page, select **Ant build configuration (build.xml) file**.

NOTE An Ant build configuration file need not be named `build.xml`, of course, but for the sake of convenience, the prompt uses the name `build.xml`.

Click Next. The Choose File wizard page appears.

5 On the Choose File page, in the **Select Ant build configuration file** field, do one of the following:

- Type the full path to the Ant buildfile that you want to import.
- Click Browse to navigate to the Ant buildfile that you want to import.

Click Next. The Parameters page appears.

6 The **Parameters** page displays the following information:

- The name of each target found in the Ant buildfile.
- The description given in the Ant buildfile. If no descriptions are entered in the buildfile, none will appear in the wizard page.

7 Select the targets that you want to import. Use the triple check mark to select all targets.

8 Click Next. The Confirm page appears.

9 On the Confirm page review the details of the targets that you are going to create.

NOTE If there are details you wish to edit, you will be able to do that from within Dimensions CM.

10 Click Finish. The dialog box displays a brief confirmation, and then the imported targets appear in the content pane.

11 Check in the build configuration to preserve the imported targets.

Editing Details of the Imported Targets

Purpose Follow this procedure to edit details of targets imported from an Ant buildfile. For example, you may wish to edit small details such as the description. You may also want to inspect the transition script, as each target will still depend on the Ant buildfile.

To edit details of targets imported from an Ant buildfile:

1 Select the Build Management tab if it is not selected already.

2 Expand the Dimensions Projects tree until the tree area shows the build configuration to which the Ant targets were imported.

3 Select the build configuration. You should be able to see the imported targets. Verify that the new targets were added to your list of build targets.

4 Check out the build configuration if it is not checked out already.

5 Click the name of the target you wish to edit. The Edit Build Target dialog box appears.

6 Edit the target details as desired. Click OK to accept the dialog box.

7 Check in the modified build configuration.

NOTE Any changes to the name or description of the target will display only in Dimensions CM and will not affect the Ant buildfile itself.

Editing the Transition Script of an Imported Ant Target

Purpose Follow this procedure to edit the transition script for a target imported from an Ant buildfile. Note that this is not the same as editing the Ant buildfile itself, and that the buildfile is still needed to build the targets that were imported from it.

To edit the transition script of a target imported from an Ant buildfile:

1 Select the Build Management tab if it is not selected already.

2 Expand the Dimensions Projects tree until the navigation pane shows the build configuration to which the Ant targets were imported.

3 Select the desired build configuration.

4 Check out the build configuration if it is not checked out already.

5 Expand the build configuration until the individual targets are visible in the navigation pane. You cannot edit the transition script unless you can see the individual targets in the Dimensions Projects tree.

6 Select the target whose transition script you wish to edit. The Transition Details section becomes visible in the content pane.

7 In the content pane, in the Transition Details section, click Edit Build Script.

The Edit Transition Script dialog box appears. The Script Content field displays the transition script. For example:

```
ant -buildfile "build.xml" "create-manifest"
```

8 Edit the transition script as desired.

NOTE Be careful editing the transition script. If you break the connection between the target and the Ant buildfile, Dimensions CM will lose the ability to build the target.

9 Check in the build configuration to save the edited target.

Running the Imported Build Job

To run the build configuration, you should follow the procedures described in Executing Builds in the Build Administration Console on page 133. You should also pay attention to the following:

- Make sure the Ant buildfile is in the build directory or is in a location that both Ant and Dimensions CM can find.

If you have targets that need to be preserved, create the Dimensions archives for those targets and then be sure that the **Do not preserve targets** check box is not selected in the Run Build wizard.

Part 4

Integrating Dimensions CM with Build Management Tools

Part 4: Integrating Dimensions with Build Management Tools contains the following chapters

Integrating Dimensions CM with Apache Ant	191
Integrating the Maven SCM Dimensions CM Provider	197
Integrating Dimensions CM with Jenkins	211

Chapter 13

Integrating Dimensions CM with Apache Ant

Introduction	192
Registering the Dimensions Ant Task with Ant	192
Using the Dimensions Ant Task	192
Syntax and Attributes	194

Introduction

Apache Ant is a Java-based build tool. Serena® Dimensions® CM provides an Ant task that you can use to access Dimensions CM functionality from within an Ant build script.

For more information about Apache Ant see the following web site:

http://ant.apache.org/

Registering the Dimensions Ant Task with Ant

You can register the Dimensions Ant task with a definition similar to the following:

```
<taskdef name="dimensions"
classname="com.serena.dmtpi.DimensionsTask">
    <classpath>
        <fileset dir="${DM_ROOT}/java_api/lib">
            <include name="*.jar"/>
        </fileset>
    </classpath>
</taskdef>
```

where you have set the Ant DM_ROOT property to an appropriate value elsewhere in the script.

For more information see the documentation for the Ant <taskdef> task on the Apache Ant web site.

Using the Dimensions Ant Task

To run Dimensions command-lines in a target, add an element similar to the following to your Ant build.xml script:

```
    <dimensions userID="builduser" password="oH.1x!@46nL"
                database="DEV@PROD" server="prod1">
        <run cmd='download /workset="DEV:CM_FUTURE"
                           /directory="java_build"
                           /user_dir="C:\Workspaces\CM_FUTURE"'/>
</dimensions>
```

You must be careful with the following characters in command-lines as they may need to be escaped:

Character	Description	Escaped as
"	Double-quotation. You only need to escape a double-quotation if the characters that enclose it are the same, for example: ■ Escape not required: '"' ■ Escape required: ''"''	&quot;
'	Single quotation mark. You only need to escape a single quotation mark if the characters that enclose it are the same, for example: ■ Escape not required: "'" ■ Escape required: '''	&apos;
&	Ampersand	&amp;
<	Less-than	&lt;
>	Greater-than	&gt;

If you use a single quotation mark to quote around the `cmd` attribute you normally only need to escape the following characters:

"

&

<

If you use a double quotation mark to quote around the `cmd` attribute you normally only need to escape the following characters:

'

&

<

You can run multiple commands in one login session by repeating the `<run>` nested element. For example:

```
<dimensions userID="dmsys" password="dmsys_test"
            database="DEV@PROD" server="prod1">
    <run cmd='SCWS DEV:CM_FUTURE'/>
    <run cmd='LWSD java_build'/>
</dimensions>
```

See the Dimensions CM *Command-Line Reference* for details of Dimensions commands.

Syntax and Attributes

Syntax

```
<dimensions>
   <run>
   <getCopy>
   <export>
```

where:

<dimensions>

Provides access to Dimensions CM command-line functionality.

Attribute	Description	Required?
userID	Dimensions user ID to connect as.	Yes
password	Password for the user ID.	Yes
database	Dimensions base database to connect to in the format NAME@CONNECTION.	Yes
server	Dimensions server hostname with optional port number, for example, ord-dm:672.	Yes

<run>

Runs a Dimensions command-line.

Attribute	Description	Required?
cmd	Dimensions command-line to run.	Yes

<getCopy>

Uses the FI command to fetch item revisions from a project to a local working location. This command is now deprecated and to improve performance you should use the <run> element with a DOWNLOAD command-line.

Attribute	Description	Required?
`srcProject`	The source project to fetch from.	Yes
`srcPath`	The source project folder.	Yes
`destPath`	The destination working location.	Yes
`expand`	Uses the /EXPAND qualifier on the FI command.	No (false)
`recursive`	Fetches items recursively from the child project folders.	No (false)

<export>

Uses the AIWS command to export revisions from one project to another. This command is now deprecated and you should use the <run> element with an AIWS command-line.

Attribute	Description	Required?
`srcProject`	The source project to export from.	Yes
`srcPath`	The source project folder.	Yes
`destProject`	The destination project to export into.	Yes
`allRevisions`	Includes all revisions (not just the tip revisions).	No (false)
`recursive`	Exports items recursively from child project folders.	No (false)

Chapter 14

Integrating the Maven SCM Dimensions CM Provider

What is Maven?	198
Terminology	198
Installing and Configuring the Maven SCM Dimensions CM Provider	200
Maven SCM Plug-in Commands and Options	202
Scenarios	205

What is Maven?

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information. For more information about Maven see the following web site:

http://maven.apache.org/

Maven SCM provides plug-ins with a common API for SCM operations. Dimensions CM has a plug-in that integrates with Maven. For more information about Maven SCM see the following web site:

http://maven.apache.org/scm/

Terminology

Archetype

An archetype is an original model or type after which other similar objects are patterned. Maven can use different archetypes when creating a project.

Artifact

An artifact is an object that is produced or used by a project. Examples of artifacts produced by Maven for a project include JARs, source and binary distributions, and WARs. Each artifact is uniquely identified by a group ID and an artifact ID that is unique within a group.

Dependency

A typical Java project relies on libraries to build and/or run. Those are called "dependencies" inside Maven. Those dependencies are usually other projects' JAR artifacts but are referenced by the POM that describes them.

Goal

Goals are executed to perform an action on a project. For example, the `jar:jar` goal compiles the current project and produces a JAR file. Each goal exists in a plug-in except for those that you define yourself. The goal name usually reflects the plug-in, for example, `java:compile` comes from the Java plug-in.

Group ID

A group ID is a universally unique identifier for a project. This is often just the project name, for example, `commons-collections`. It is helpful to use a fully-qualified package name to distinguish it from other projects with a similar name, such as `org.apache.maven`.

Project

Maven operates with projects and everything that you build are projects. Projects follow a well defined Project Object Model (POM). Projects can depend on other projects called "dependencies". A project may consistent of several subprojects, however these subprojects are still treated equally as projects.

Plug-in

Maven is organized in plug-ins and every piece of functionality in Maven is provided by a plug-in. Plug-ins provide goals and use the metadata found in the POM to perform their task. Examples of plug-ins are jar, eclipse, and war.

Project Object Model (POM)

The Project Object Model is the metadata that Maven needs to work with your projects. It is called `pom.xml` and it is located in the root directory of each project.

Repository

A repository is a structured storage of project artifacts organized under the following structure:

```
$MAVEN_REPO/groupId/artifact type/project-version.extension
```

For instance, a Maven JAR artifact will be stored in a repository under:

```
/repository/maven/jars/maven-1.0-beta-8.jar
```

Maven uses the following repositories:

- Remote repositories are a list of repositories that Maven downloads from. For example, an internet repository, its mirrors, and a private company repository.
- The "central repository" is the repository to which generated artifacts are uploaded, for example, for use by developers.
- The "local repository" is the repository that you have on your local machine. Artifacts are downloaded just once, unless they are a snapshot, from the remote repository to your local repository.

Snapshots

Projects can and should have a special version including a snapshot to indicate that they are a "work in progress" and are not yet released. When a snapshot dependency is encountered it is always searched for in all remote repositories, and downloaded again if it is newer than the local copy. The version can either be the string SNAPSHOT, indicating "the very latest" development version, or something similar to 1.1-SNAPSHOT, indicating development that will be released as 1.1 (newer than 1.0, but not yet 1.1).

Checkin and checkout

If you are using streams in Dimensions CM the following terminology is different: you deliver changes (check in) and update your work area (check out).

Installing and Configuring the Maven SCM Dimensions CM Provider

The following sections describe how to install and configure the Dimensions CM Maven plug-in.

Downloading and Installing

1. Download Maven from the following web site:

   http://maven.apache.org/download.html

2. Install Maven.

3. Download the Dimensions CM Maven SCM plug-in:

   a. Logon to the Serena Support web site:

      http://support.serena.com

   b. Select the Downloads tab.

   c. From the Product list select Dimensions CM.

   d. In the Available Operating Systems column select the appropriate version of Windows or Linux.

   e. In the CM Integrations section download the Dimensions CM Integration for Maven zip or tar file.

4. Specify one of the following environment variables:

   - MAVEN_HOME
   - M2_HOME

5. Extract the contents and run the Maven SCM plug-in installer.

Specifying User Credentials in pom.xml

The format for the user credentials in `pom.xml` is:

```
scm:dimensions://[<username>][:<password>]@<server>[:<port>]/
    <dbName>@<dbConnection>[/product][:project][/relativeLocation]
```

Examples:

```
scm:dimensions://fred:fred_test@dimserver:671/cm_typical@dim12/
    qlarius:mainline_vs_str/
```

```
scm:dimensions://fred:fred_test@dimserver:671/cm_typical@dim12/
    qlarius:mainline_vs_str/Qlarius_Underwriter/Qlarius_Underwriter/
    Dialogs/
```

Edit your `pom.xml` file accordingly.

Specifying User Credentials in an External File

The user credentials in `pom.xml` are optional and you can specify them in an external file, `settings.xml`. To specify user credentials do the following:

1 Remove any user credentials from `pom.xml`. For example:

```
scm:dimensions://@dimserver:671/cm_typical@dim12/
    qlarius:mainline_vs_str/
```

2 Open `settings.xml` in the Maven local repository, typically located in:

`${user.home}/.m2/settings.xml`

Note: If `settings.xml` does not exist in the above location copy it from:

`$M2_HOME/conf/settings.xml`.

3 Under the `<servers>` element add a new `<server>` element and specify:

- The server and port.
- The <username> and <password>.

For example:

```
<server><id>dimserver:671</id><username>fred</
    username><password>fred_test</password></server>
```

These steps will ensure that:

- There is no confidential information in `pom.xml` and it can be distributed for use by other members of your organization.
- The correct login credentials are used when Maven SCM commands are issued.

Goals

The high level goals currently supported by the Dimensions CM Maven plug-in are:

Goal	Description
`scm:bootstrap`	■ Executes `scm:checkout` to fetch the project or stream to a clean area. ■ Executes the default goal specified in the newly fetched pom.xml file located at the root of the project/stream.
`release`	Through a combination of `release:prepare`, `release:perform` and `release:rollback` supports the automatic revision of pom.xml versions, tagging and deployment. `scm:tag` creates a Dimensions baseline. **Note**: Dimensions CM deployment is currently not supported with the release goal.

Maven SCM Plug-in Commands and Options

Commands

The table below lists the Maven SCM plug-in commands that are completely or partially supported by the Dimensions CM Maven plug-in.

Command	Options	Extension or deviation to standard Maven command
add		■ Currently there is no way to 'move' the association of an added file if after the command is complete but before check-in, the file is subsequently moved or renamed. ■ Currently there is no way to explicitly schedule folders for addition. This only affects delivery of empty folders since new folders containing files scheduled for delivery will be created implicitly. ■ Any `message` or `binary` parameter specified by the command is ignored.
changelog	■ `dimensions.project` ■ `dimensions.relativeLocation`	■ The file set specified by the command is currently ignored and all changes in the project or stream scope and date range are reported. ■ Any branch parameter specified by the command is ignored. ■ dateFormat is ignored. ■ 'Excludes' and 'includes' parameters are ignored.
checkin	■ `dimensions.all` ■ `dimensions.add` ■ `dimensions.project` ■ `dimensions.relativeLocation` ■ `dimensions.requests`	■ Executes a Dimensions DELIVER command. ■ Any `scmVersion` parameter specified by the command is ignored.
checkout	■ `dimensions.baseline` ■ `dimensions.project` ■ `dimensions.relativeLocation`	■ Executes a Dimensions UPDATE command to a clean work area. ■ Any `scmVersion` parameter specified by the command is ignored.
status	■ `dimensions.all` ■ `dimensions.add` ■ `dimensions.project`	

Command	Options	Extension or deviation to standard Maven command
tag	■ dimensions.attributes ■ dimensions.project ■ dimensions.requests ■ dimensions.type	■ Server operation to create a baseline using the specified tag as the new tip baseline ID. ■ The file set specified by the command is currently ignored and all latest items within the project or stream scope are included in the baseline. ■ Any scmTagParameters values specified by the command are ignored.
update	■ dimensions.baseline ■ dimensions.project ■ dimensions.relativeLocation	■ Any scmVersion parameter specified by the command is ignored. ■ Parallel development is not supported.

Options

Some of the commands listed above support options that are not available in the Maven SCM plug-in. To pass an option through to the Dimensions plug-in, specify it on the command line as -Dkey=value. This passes the option through as a system property. The following table lists the currently supported options.

Key	Type	Description
dimensions.all	boolean	Specifies to scm:checkin and scm:status that all foreign content is to be considered, not just files known to have been fetched from the stream being delivered to. This is a stream specific option that has been implemented to support the case where scm:checkout is against a baseline, but scm:checkin is back to the stream from where the baseline originated.
	Example: mvn scm:checkin -Ddimensions.all=true	
dimensions.add	boolean	Specifies to scm:checkin and scm:status that all additions are to be considered, not just those scheduled for delivery by scm:add.
	Example: mvn scm:checkin -Ddimensions.add=true	
dimensions.attributes	user defined attributes	JSON formatted set of Dimensions CM user defined attributes.
	Example: When creating a baseline use the tag command for a type with a mandatory attribute called 'planet': mvn:tag -Dtag="MYTIPBL" -Ddimensions.type="MYBLTYPE" -Ddimensions.attributes={planet:Mars}	

Key	Type	Description
dimensions.baseline	baseline specification	Baseline specification that overrides any project or stream specified within the URL for scm:checkout or scm:update. You can specify just the baseline ID on the command line with the product obtained from the URL.
	Example: mvn:checkout -Ddimensions.baseline=mybaseline	
dimensions.project	project/stream specification	Project or stream specification that overrides any specified in the URL. You can specify just the product in the URL and the project/ stream ID on the command line.
	Example: mvn:checkout -Ddimensions.project=mystream	
dimensions.relativeLocation	repository relative path	Project or stream relative path (using forward slashes) that overrides any path specified in the URL.
	Example: mvn:checkout -Ddimensions.relativeLocation="Qlarius_Underwriter/Qlarius_Underwriter/Dialogs"	
dimensions.requests	request list	Request specification, or a comma separated list of requests. Primarily used with the scm:checkin command.
	Example: mvn:checkin -Dmessage="the comment" -Ddimensions.requests="QLARIUS_CR_99"	
dimensions.type	object type name	Dimensions object type to override the default or for specifying user defined attributes. See the scm:tag command specified in the example for dimensions.attributes above.

Scenarios

The scenarios below show you ways that you can use the Maven plug-in.

Scenario 1: Updating an Existing Stream

This simple scenario describes how to update an existing stream and assumes that:

- A stream exists and contains files and folders.
- Maven has been installed and the mvn executable is on the path.
- The Maven SCM Dimensions CM Provider is installed.

Steps:

1. Create an empty folder on disk.
2. Copy the following into a new file called `pom.xml` in that folder:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.acme</groupId>
   <artifactId>JavaStr01</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   <name>JAVA_STR_01</name>
   <url>http://maven.apache.org</url>
   <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>
   <scm>
      <connection>
         scm:dimensions://fred:fred_test@localhost:671/cm_typical@dim12/
qlarius:java_str_01
      </connection>
      <url>http://localhost:8080/dimensions</url>
   </scm>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-scm-plugin</artifactId>
            <version>1.5</version>
            <dependencies>
               <dependency>
                  <groupId>org.apache.maven.scm</groupId>
                  <artifactId>maven-scm-provider-dimensions</artifactId>
                  <version>1.5</version>
               </dependency>
            </dependencies>
         </plugin>
      </plugins>
   </build>
</project>
```

3 Edit the <connection> element in pom.xml to specify your user credentials and server connection details.

4 On the command line execute (from the same folder where pom.xml resides):

```
mvn scm:update
```

Result:

The stream is fetched to the current folder.

NOTE

- The top section of elements in pom.xml specify the properties of the stream being fetched. Their values are not important when performing simple SCM operations and are used when Maven goals are run to compile/build sources fetched from Dimensions.
- The element <build> in pom.xml is required to help Maven find and load the Dimensions CM provider.

Scenario 2: Checking Out Using a Copy of pom.xml

This scenario is similar to scenario 1 above. The difference is that the same pom.xml file has been checked into the stream at the root of the project and copied to an empty folder to simulate having been emailed to a user who wants to fetch the stream.

Steps:

From the folder where pom.xml is located execute the following command line:

```
mvn scm:checkout
```

Result:

The stream is fetched to the following subfolder:

```
target\checkout
```

NOTE

- After this initial fetch you can discard the original pom.xml file and perform all future SCM operations directly from the subfolder where the stream was fetched (the pom.xml from the stream was fetched there).
- Alternatively you can also use any Dimensions CM client to fetch files and then run pom.xml for any subsequent operations.

Scenario 3: Checking In a Modified File

This scenario is similar to scenario 2 above. The difference is that you have modified files locally and want to check them in (deliver to Dimensions CM).

Steps:

1. To check in files to a location in the repository that is different from where you checked them out, add the following to the `scm` element of your `pom.xml` file:

   ```
   <developerConnection>scm:dimensions://fred:fred_test@localhost:671/
   cm_typical@dim12/qlarius:java_str_02</developerConnection>
   ```

2. Locally modify files.

3. Execute the command line:

   ```
   mvn scm:checkin -Dmessage="test modification"
   ```

Result:

The modified files are delivered to the stream.

Scenario 4: Checking In an Added File

This scenario is similar to scenario 2 above. The difference is that you have added a file locally, want to schedule a delivery, and check it in (deliver to Dimensions CM).

Steps:

1. To check in files to location in the repository that is different from where you checked them out, see step 1 in scenario 3 above.

2. Add a new local file: `AddTest.txt`

3. Execute the following commands:

   ```
   mvn scm:add -Dincludes="AddTest.txt"
   ```

   ```
   mvn scm:checkin -Dmessage="test addition"
   ```

Results:

- The first Maven command ensures the new file is scheduled for delivery.
- The second Maven command delivers the new file back to the stream.

Scenario 5: Checking Out and Executing Default Goals

This scenario shows you how to mix Maven and Dimensions CM commands.

The `boostrap` command issues the `scm:checkout` command and if the project or stream that is fetched has a `pom.xml` file at its root, the default goals specified in the file are executed. The scenario assumes that:

- Maven has been installed and the mvn executable is on the path.
- The Maven SCM Dimensions CM Provider is installed.
- A stream exists that contains the source code for a simple java app that is compatible with being compiled, unit tested, documented, and packaged as a *.jar file by Maven.

Steps:

1 Check a new `pom.xml` file into the root of the stream with the following content (replace the user credentials and connection details):

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.acme</groupId>
   <artifactId>SolarSystem</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   <name>SolarSystem</name>
   <url>http://maven.apache.org</url>
   <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>
   <scm>
      <connection>scm:dimensions://fred:fred_test@localhost:671/cm_typical@dim12/
qlarius:solarsystem</connection>
      <developerConnection>scm:dimensions://fred:fred_test@localhost:671/
cm_typical@dim12/qlarius:java_str_02</developerConnection>
      <url>http://localhost:8080/dimensions</url>
   </scm>
   <build>
      <defaultGoal>package</defaultGoal>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-scm-plugin</artifactId>
            <version>1.5</version>
            <dependencies>
               <dependency>
                  <groupId>org.apache.maven.scm</groupId>
                  <artifactId>maven-scm-provider-dimensions</artifactId>
                  <version>1.5</version>
               </dependency>
            </dependencies>
         </plugin>
      </plugins>
   </build>
   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>3.8.1</version>
         <scope>test</scope>
      </dependency>
   </dependencies>
</project>
```

2 Manually place a copy of this `pom.xml` file in a new empty folder to simulate being emailed for initial bootstrap.

3 From that folder execute the command line:

```
mvn scm:bootstrap
```

Result:

The stream is fetched from Dimensions CM to the "target\checkout" subfolder and Maven then compiles, unit tests, and packages the Java application.

Chapter 15

Integrating Dimensions CM with Jenkins

Introduction	212
Example: Installing Jenkins	213
Configuring Jenkins System Properties	214

Introduction

This chapter includes an example of how to set up a Java Ant build on Jenkins for the Qlarius sample projects that ship with Dimensions CM. The steps may differ if you are using your own source code and build. After you have a working Jenkins build you need to configure CM Pulse to use it.

IMPORTANT! These instructions assume working knowledge of Jenkins build systems.

The following instructions describe an example flow for installing and configuring Jenkins that you can adapt for your environment. All the steps are optional unless stated otherwise.

NOTE For information about using Jenkins with Dimensions CM Pulse see the following topics in the Pulse online help:

- *Check the Health and Quality of Work* | *About Experts and Expert Chains*
- *Check the Health and Quality of Work* | *Create Expert Chains*
- *Configure Experts* | *Jenkins Expert*

Example: Installing Jenkins

Preliminary Steps

1. Install a Java Development Kit (JDK).
2. Install Apache Ant if your build will be using the Ant build engine.

Installing Jenkins Plugins

1. Stop the Serena Common Tomcat service.
2. Download the `jenkins.war` file from the Jenkins web site:

   https://jenkins-ci.org/
3. Copy the war file to your Serena Common Tomcat *webapps* folder, for example:

   ```
   C:\Program Files\Serena\common\tomcat\8.0\webapps
   ```
4. Copy the Dimensions SCM Plug-in to the Jenkins plugins folder:
   a. Copy the plug-in from your `webapps\pulse\WEB-INF\jenkins` folder, for example:

      ```
      C:\Program Files\Serena\common\tomcat\8.0\webapps\pulse\
          WEB-INF\jenkins
      ```
   b. Paste the plug-in into your `webapps\jenkins\WEB-INF\plugins` folder, for example:

      ```
      C:\Program Files\Serena\common\tomcat\8.0\webapps\jenkins\
          WEB-INF\plugins
      ```
5. Copy the following Dimensions CM Java API libraries from your Dimensions server installation folder to your Jenkins installation folder:
   - `darius.jar`
   - `dmclient.jar`
   - `dmfile.jar`
   - `dmnet.jar`

   For example, copy the libraries from:

   ```
   C:\Program Files\Serena\Dimensions 14.3\CM\AdminConsole\lib
   ```

   to:

   ```
   C:\Program Files\Serena\common\tomcat\8.0\webapps\jenkins\WEB-
       INF\lib
   ```
6. Restart the Serena Common Tomcat service.

Configuring Jenkins System Properties

Preliminary Steps

1 Open a web browser and go to the following URL:

http://<server>:8080/jenkins

NOTE Your server name and port number may be different.

2 Click **Manage Jenkins**.

3 Click **Configure System**.

Configuring a JDK

1 Click **Add JDK**.

2 Enter a name for the JDK, for example: `My JDK 1.7`

3 Unselect **Install automatically**.

4 Enter your JDK_HOME value, for example: `C:\Program Files\Java\jdk1.7.0_45`

Configuring Apache Ant

1 Click **Add Ant**.

2 Enter a name for the Ant installation, for example: `My Ant`

3 Unselect **Install automatically**.

4 Enter your ANT_HOME value, for example: `C:\apache-ant-1.9.2`

5 Click **Save**.

Configuring a New Build Job

1 Click **New Job**.

2 Enter a name for this project, for example: `Qlarius`

3 Select a project type, for example, *Build a free-style software project*.

4 Click **OK**.

Adding Parameters

1 Select the option **This build is parameterized**.

2 Optionally add the following parameters:

- String Parameter, Name: `cmkey`

  This parameter identifies a build run to a Jenkins Expert in Dimensions CM Pulse. You can configure Jenkins Expert to automatically add `cmkey` to your Jenkins jobs, if not you must add it manually. For details see the Pulse online help.

- String Parameter, Name: `repo`
- String Parameter, Name: `changeset`
- String Parameter, Name: `stream`
- String Parameter, Name: `version`

Selecting the JDK

If you have more than one installed JDK you may need to select the JDK you created earlier.

Adding the Dimensions CM Plugin as SCM

1 In **Source Code Management** select **Dimensions**.

2 In the **Project Name** box enter the stream name.

NOTE You can also specify "${stream}" if you are using a CM plugin that supports this variable. This makes copying jobs easier and allows a job to be triggered from more than one stream, but ties the job to CM Pulse.

3 In the **Folder** box enter: `/`

4 Select **Clear the contents of the workspace**.

5 To show all Dimensions fields click **Advanced**.

6 Enter your Dimensions CM connection details, for example:

- Login Name: `dmsys`
- Password: `dmsys_test`
- Server: `myserver`
- Database: `cm_typical@dim12`

7 Select **Use Update**.

Adding a Build Step

This example is for Ant builds. You may have different build steps that are unique to your build process.

1. Click **Add build step**.
2. Select **Invoke Ant**.
3. Select the Ant installation you created earlier.
4. Click **Advanced** and enter the location of your build.xml file in the **Build File** box, for example:

   ```
   Qlarius Underwriter/build.xml
   ```

Saving the Build Job

Click **Save**.

Testing the Build Job

1. To test the build job click **Build Now**.
2. Enter dummy values for:
   - `key`
   - `repo`
   - `changeset`
   - `stream`
3. Click **Build**.
4. Check that the build was successful.

Part 5
Appendices

Part 5: Appendices contains the following appendixes:

Troubleshooting Dimensions Build	219
Dimensions Build Configuration Symbols	225
Creating and Embedding Build Footprinting	231
Dimensions Build Utility Programs	239
Dimensions Build Security	259
The Build Configuration XML File Format	263
Customizing a Build Server	283
Load Balancing a Build Server	287
Advanced Build Settings	291

Appendix A

Troubleshooting Dimensions Build

About Troubleshooting	220
Viewing Errors Listed in Build Monitor Events	220
Specific Errors	221
Using Temporary Files to Debug Your Builds	223
Windows Server 2003	224

About Troubleshooting

This section describes various conditions and error messages you may see in the course of using Dimensions Build. Later, the section also covers debugging information you can find in temporary files.

Error messages may be written to any of several areas:

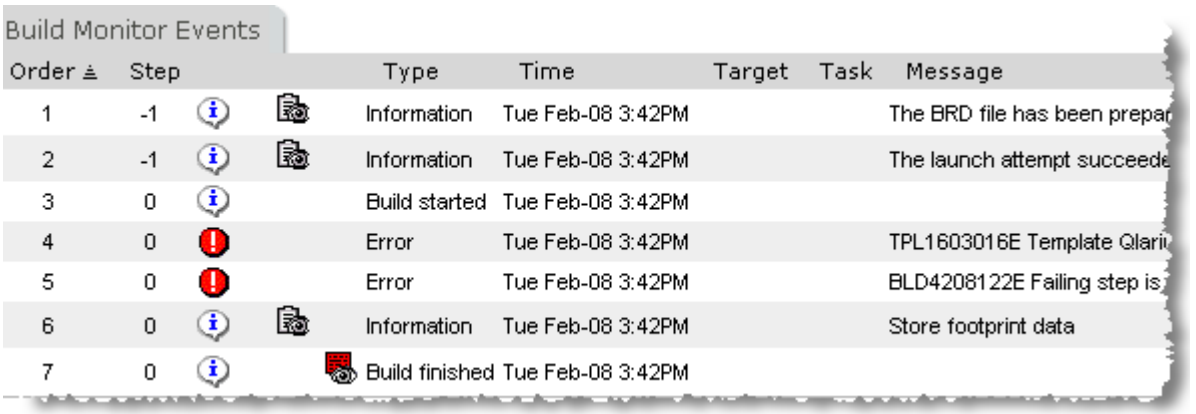
- in Build Execution Statuses
- in Build Monitor Events
- in the Tomcat window
- in wizard pages
- On MVS, in SDSF/JES spool
- On MVS, in the MVS System log

If you are having a problem, you should look for error messages, and then search within this chapter for any information relevant to the messages you see.

If you are unable to find helpful information in this chapter contact Serena customer support at http://www.serena.com/support.

Viewing Errors Listed in Build Monitor Events

If your build job encounters an error during execution, Dimensions CM displays an error listing in Build Monitor Events:



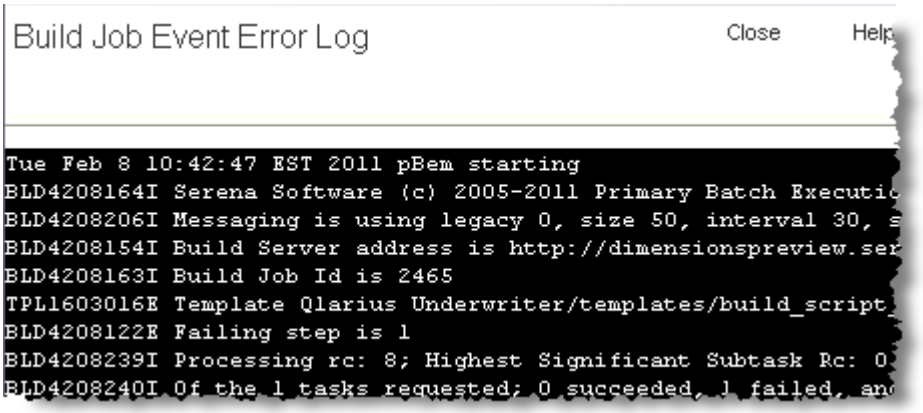
Recall that you view Build Monitor Events in one of two ways:

- After a build executes, the Build Execution Statuses dialog box appears; click the See Build Events icon on that dialog box to view Build Monitor Events.
- From the Build Job Monitoring tab, locate your build job under the History folder, then click the appropriate link under Execution History to view Build Monitor Events.

To view the error log:

1 Click the **View error log** icon:

The Build Job Event Error Log appears:



2 Review the information in the error log to determine what went wrong.

Specific Errors

This section presents a list of specific errors you may see, with possible causes.

Area <area name> is in use

The full error message is:

```
Area <area_name> is in use and you cannot update <property> at this time
```

If you see this error message, it means that you are trying to edit a build area that has already been associated with a checked in build configuration. Once a build configuration with an attached build area has been checked in, you can no longer modify the following build area properties:

- Area ID
- Stage ID
- Network node
- Directory

Attempt to close invalid connection

If your build does not execute, and if you see this message in the Tomcat window, it may mean that Tomcat has timed out your connection. Restart Tomcat using Start > Programs > Serena > Common Tools > Start Common Tomcat.

Failed to authenticate to the build agent

If your build displays this error message, it may mean that the you have entered an incorrect password for the user ID under which Dimensions CM logs in.

It could also mean that the build agent is not installed or not running on the build machine.

Failed to find product-specific upload rules

If your build executes but displays this error message, it may mean that your Dimensions administrator has not yet defined a default item library. Dimensions CM will not be able to upload the build target to Dimensions as a new item unless it can locate a default item library.

No build areas found for the selected configuration

If you see this error in the Run Build wizard, it could mean that you are trying to use the Build Scheduling tab to execute a never-before-executed version of a build configuration.

If a version of a build configuration has been checked in, but has never been run, it will not contain a build area association. A version of a build must be run at least once before you can execute it from the Build Scheduling tab.

If you see this error when trying to execute the build from the Build Management tab, it means that the build configuration has no build areas attached.

Template pbem_stop.cmd open failed

If your build job displays this error message, it may mean that the temp directory defined in the Dimensions configuration file contains a space.

Edit <DM_ROOT>/*dm.cfg* and change the value of DM_TMP to a path that does not contain spaces.

You do not have a role to extract item

If your build job displays this error message, it may mean that the Dimensions user you have logged in as does not have a role that allows updating the build target. More specifically, the Dimensions user may lack a role on the first transition of the item. For example, an .EXE file using the default LC_EXECUTABLE lifecycle will transition from BUILT to TESTED. If the Dimensions user does not have a role on that transition, the check-in of the build target may fail.

Use the Dimensions Administration Console to check the appropriate roles and transitions, or contact your Dimensions administrator.

Using Temporary Files to Debug Your Builds

During a build Dimensions CM produces temporary files containing information that can help you debug problems.

Preserving Temporary Files

Normally temporary files are deleted after each build but you can choose to preserve them:

1 Open the Dimensions configuration file, dm.cfg.

2 Enter this variable and value:

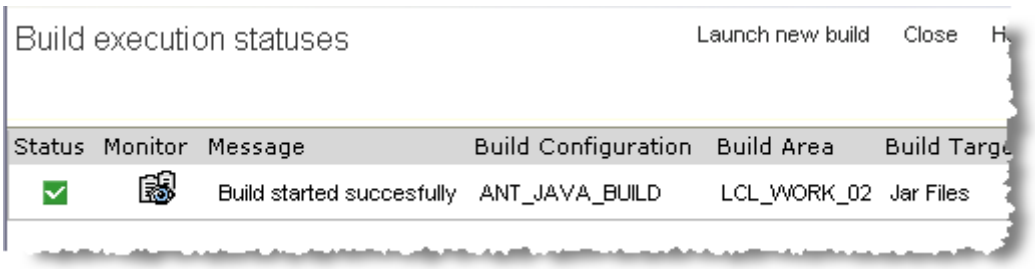
DM_BUILD_CLEAN_TEMP_FILES false

NOTE To make this variable take effect, restart the Dimensions listener.

Locating Temporary Files

To locate the temporary files do the following:

1 After running a build, the Build Execution Statuses dialog box should be visible. Click the See Launch Log icon:



The View Launch Attempt Log dialog box appears.

2 Scroll to the bottom of the Launch Attempt Log dialog box. You will see a message similar to this:

```
Passing the template for execution
cmd = cmd /c C:\DOCUME~1\dmsys\LOCALS~1\Temp\TPL1B6~3.BAT
    >"C:\DOCUME~1\dmsys\LOCALS~1\Temp\tpl2b64-2.tmp" 2>&1
Template processing and submission complete: no errors
```

3 Note the location of the .BAT and .TMP files and navigate to this directory. This location is controlled by the DM_TMP variable in dm.cfg.

NOTE On Windows, the directory `Documents and Settings\<username>\Local Settings` is a hidden directory. You will have to show hidden files and folders to see it.

4 Locate the files with the extension .BAT and .TMP. (The exact names of the files are different each time.) You may need to compare the files with similar files from an build that succeeded.

About the BRD File

Another file you will see listed in the Launch Attempt Log dialog box is the BRD file, for example:

```
BRD file name = C:\DOCUME~1\dmsys\LOCALS~1\Temp\ptb641.tmp
```

This file contains the following information:

- The values of build options.
- The steps in the build.
- The order in which the steps occurred.

In particular, users creating Openmake-type builds can read the values of the Openmake build variables from this file.

Windows Server 2003

If a remote node on which you are executing builds is running Windows Server 2003, the user ID executing the build script or job must be an administrator.

Appendix B

Dimensions Build Configuration Symbols

Dimensions Build Configuration Symbols 226

Dimensions Build Configuration Symbols

This appendix describes the symbols in the Dimensions configuration file that affect the Primary Build Execution Monitor (PBEM). The configuration file, `dm.cfg`, is located in:

- Windows: `%DM_ROOT%`
- UNIX: `$DM_ROOT`
- MVS: `MDH.V1031.MDHPARM(MDHTDCFG)`

DM_MAX_PBEM_RETRIES

Description	Specifies the maximum number of times the PBEM tries to send a message to the build server. This symbol is useful when a build server is heavily loaded.
Value Type	Integer
Default value	5
Example	`DM_MAX_PBEM_RETRIES 5`

DM_BUILD_OPTIMIZE

Description	Use this symbol to perform optimization. The PBEM will optimize the build by omitting steps that are not required. If you do not need optimization, comment out this symbol.
Possible Values	■ Y (Yes) ■ N (No)
Default value	Y (set during installation)
Example	DM_BUILD_OPTIMIZE Y

DM_BUILD_OPTIMIZE_EARLY_TEST

Description	Defining this symbol on MVS may improve performance. If you encounter templating problems try removing this symbol while you debug the template.
Possible Values	■ Y (Yes): enables optimization. ■ Undefined: disables optimization.
Default value	Undefined
Example	DM_BUILD_OPTIMIZE_EARLY_TEST Y

DM_BUILD_MSGQ_SIZE

Description	Sets the maximum queue size for messages. When more messages than the specified value are collected, the queue is flushed and sent to the build server.
Value Type	Positive whole number
Default value	200
Example	`DM_BUILD_MSGQ_INTERVAL 100`

DM_BUILD_MSGQ_INTERVAL

Description	Specifies the number of seconds before a flush is performed. If this time is reached before the queue fills up, it is flushed anyway. If you specify zero there is no interval checking and only the value of DM_BUILD_MSGQ_SIZE (see above) is important. If this symbol is not zero, the first message is always sent immediately.
Value Type	Positive whole number
Default value	0
Example	`DM_BUILD_MSGQ_SIZE 5`

DM_BUILD_MSGQ_STEP

Description	Causes the queue to be flushed on the last message for every step.
Possible values	■ 0: off ■ 1: on
Default value	0
Example	DM_BUILD_MSGQ_STEP 1

DM_BUILDERWS_URL

Description	Specifies the URL of a primary build server that is different from the default web server specified by the configuration symbol DM_WEB_URL. The build GUI is still invoked using DM_WEB_URL. This allows the build server to exist on a web server that is different from the regular server hosting the Dimensions clients and Administration Console.
Example	`DM_BUILDERWS_URL http://buildserver:8080/bws`

DM_MVS_REXEC_DELETE_JCL

Description	Deletes temporary files related to JCL generation and submission.
Possible Values	■ Y (Yes): delete. ■ Undefined: does not delete.
Default value	Undefined
Example	DM_MVS_REXEC_DELETE_JCL Y

DM_BUILD_CLEAN_TEMP_FILES

Description	Deletes listing files and the BOM after they are sent to the build server (user and system generated files). If you set this variable to No, temporary files used for a build are preserved and output files created by DMSAVESTDOUT and DMOSTDOUT are not affected.
Possible Values	■ Y (Yes) ■ N (No)
Default value	Y
Example	`DM_BUILD_CLEAN_TEMP_FILES Y`

DM_DELIVER_RETRY_TIMEOUT

Description	Specifies the timeout period (in minutes) for attempting to deliver files back to a stream.
Default value	0
Example	`DM_DELIVER_RETRY_TIMEOUT 5`

DM_BLD_ERROR_INVALID_REVISIONS

Description	When an item revision is not at the current stage that is being built, use this symbol to control whether the message that is issued is an error or a warning. If it is an error, the build is stopped before it is launched. For more details see page 43.
Possible Values	■ N (No): warnings are displayed and the build is launched. ■ Y (Yes): error messages are displayed and the build is stopped before it is launched.
Default Value	N
Example	`DM_BLD_ERROR_INVALID_REVISIONS Y`

DM_BUILD_ORDERING

Description	Globally enables or disables build ordering. For details see page 32.
Possible Values	■ Y (Yes) ■ N (No)
Default Value	Y (YES)
Example	`DM_BUILD_DMORDERING YES`

DM_MVS_TZ

To enable the build optimizer to work properly, set the symbol 'DM_MVS_TZ' to the time zone that the mainframe is in. For more details see the section *Customizing Variables in the Dimensions Configuration File* in the chapter *Installing Dimensions for z/OS* in the *Dimensions for z/OS User's and Administrator's Guide*.

DM_MVS_SBEM_TEMPNAME

Description	Use this build option to specify a pattern string for temporary file names. You can insert values into this string to create unique names. When you use this option in MDHTDCFG you must use the '%' symbol twice to defeat replacement processing. Note: The Secondary Build Execution Monitor (SBEM) `-g` option uses the pattern that you specify and overrides the value in MDHTDCFG. See page 245.
Possible Values	■ `%u`: MVS userid ■ `%z`: Unique symbolic value that is suitable as a data set qualifier. The maximum size is 7 bytes. Is unique only to the second within a century. See DMUNIQUE. ■ `%j`: JES job number ■ `%s`: Sequence number ■ `%t`: Name of temporary file from JCL.
Default Value	`%u.MDHS.%z.%j%s.%t.T`

Appendix C

Creating and Embedding Build Footprinting

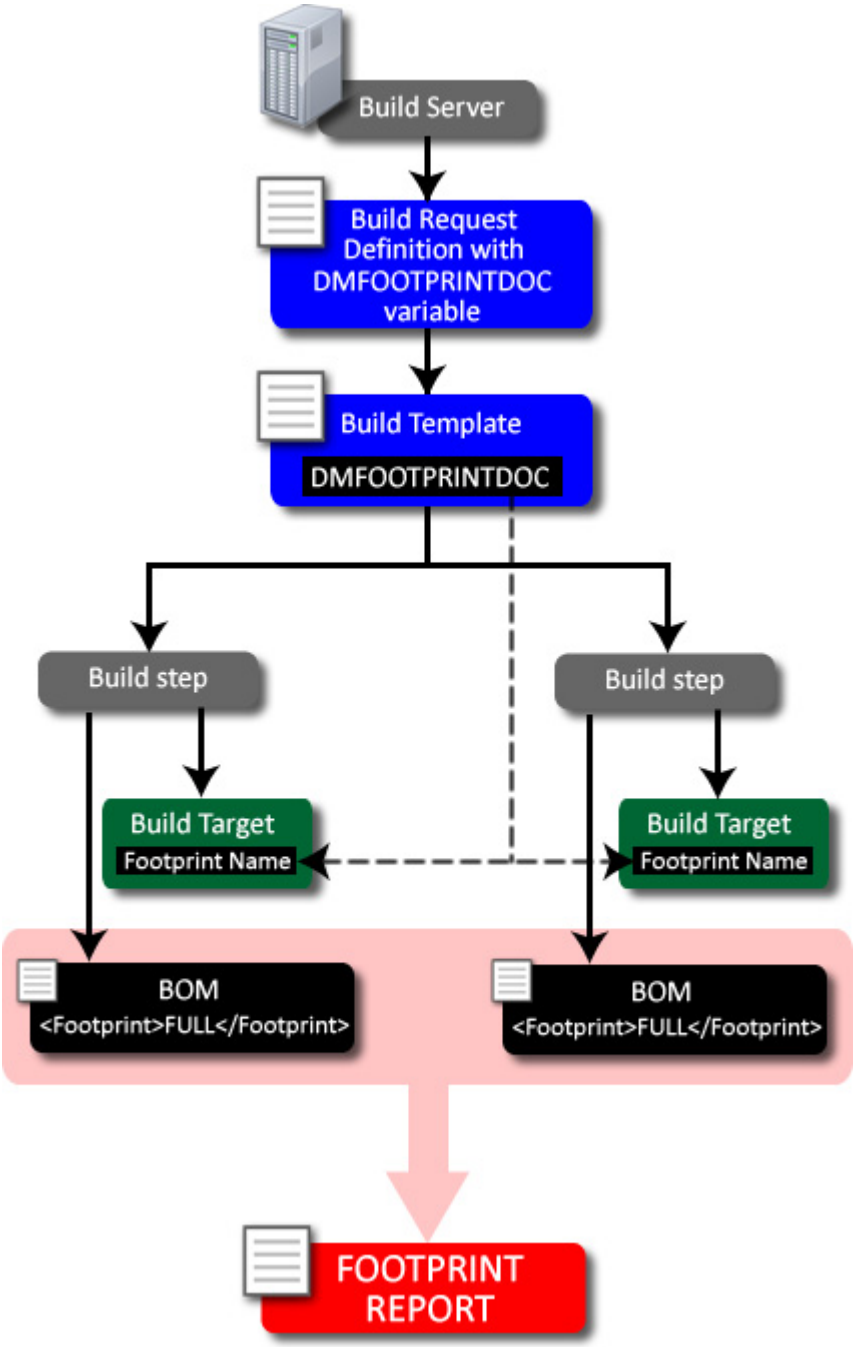
What is Build Footprinting?	232
How does Build Footprinting Work?	233
Setting Up Build Footprinting	234
The Footprint Report Format	236
Configuring Build Footprinting on MVS	237
Configuring Build Footprinting on MVS	237

What is Build Footprinting?

A build footprint is an XML document that describes all of the targets that were built for a specific build project and details the composition of these targets. The name of each footprint report is unique, and the footprint report is typically generated and checked into Dimensions when a build is completed. A footprint report is different from a BOM (Bill of Materials), which only provides detailed information about the makeup of each separate built target.

How does Build Footprinting Work?

Build footprinting works as follows:



1. When a build is launched the Build Management Server creates a BRD (Build Request Definition) file. By default the BRD includes the variable DMFOOTPRINTDOC.
2. If footprinting is required the build template driving the build uses the variable DMFOOTPRINTDOC when building a target. For details about writing build templates see the *Developer's Reference*.

NOTE You can also use a legacy variable called DMFOOTPRNT that is added by default to the BRD file. This variable contains a summary of a build that you can embed in a target. The MVS template MDHBLNKC that ships with Dimensions CM embeds this variable as well as the new footprint report variable, DMFOOTPRINTDOC.

3. In each build step:
   - A BOM file for the target is created by the build process that is driven by the build template. If footprinting is required the following optional tag is added to the BOM.

     `<Footprint>FULL</Footprint>`

     This line triggers a footprint report to be created for this unique build target.
   - The footprint report name is embedded by the build template into the build target. You can specify the name of the footprint report in the Dimensions configuration variable DM_FOOTPRINT_FOLDER, for details see "Specifying where the Footprint Report is Checked In" on page 235.
4. When the build is completed the footprint report is generated for the entire build using the specified name, and checked into Dimensions with the BOMs for each built target

Setting Up Build Footprinting

Enabling Build Footprinting

To enable build footprinting you must create a BOM. There are two ways that you can create a BOM:

- Use the BOM API from a customer written program called by the build template. This API can be used on all platforms. See the *Developer's Reference* for details.
- (MVS only) Use the SBEM (Secondary Build Execution Monitor) and in your build template. Add the following line:

  //*SBEM TFP DDNAME [(*)] [listing-type-name]

  Using TFP instead of the TGT causes the SBEM to generate the required entry in the BOM. For details about using the SBEM see page 244.

  There is an example of this technique in Dimensions in:

  `<Dimensions instance>.TEMPLATE(MDHBLNKC)`

Embedding the Footprint Report Name in the Target

Serena does not provide a solution for embedding footprint report names in targets. However, on MVS for standard LOAD modules Serena supplies a program called MDHLLNK0 that enables you to embed footprinting information. For more information see page 251.

Configuring the Footprint Report Name

Use the variable DM_FOOTPRINT_FOLDER in the Dimensions configuration file, dm.cfg, to configure the name of the footprint report and its associated path. You can customize the report name by including special formatting symbols prefixed by a plus '+' sign:

- +p: specifies the build node.
- +a: specifies the area name (spaces are replaced by an underscore '_').
- +j: specifies the job number.
- +z: a six digit character identifier (MVS only).

Default value: FOOTPRNT/J+z.XML

NOTE The +p,+a, and +j expansions are not suitable for use in MVS projects that use deployment areas as these may generate strings that are not compatible with MVS data set naming standards.

Specifying where the Footprint Report is Checked In

By default the footprint report is checked into Dimensions in the original project, which may cause the footprint to be deployed. If you do not want to deploy the footprint, use a different project to collect the footprint reports. The filename generated is globally unique so one project can collect footprint reports for all builds.

Use the variable DM_FOOTPRINT_CHECKIN_PROJECT in dm.cfg to specify where the footprint is checked in, where:

- *CURRENT: checks the footprint into the current project.
- *NONE: checks the footprint into $GENERIC.

Default value: the variable contains the name of the project in the format PRODUCT:PROJECT.

The Footprint Report Format

The footprint report is created at the end of each build. The report has environmental information for the whole build, followed by detailed explosion information about each fully footprinted target.

The format of the footprinting report is as follows:

```
<SerenaExecutableFootprint>
 <Workset>prod:proj</Workset>
  <BuildConfig>
   <ConfigName>ccccc</ConfigName>
   <ConfigDescription>ddddddd</ConfigDescription>
  </BuildConfig>
 <Environment>
  <Node>lllllll</Node>
  <Directory>stem</Directory>
  <Area>area name</Area>
  <DimensionsUser>dimuser</DimensionsUser>
  <RemoteUser>platformuser</RemoteUser>
  <SubmitData>job(Jnnnnnn)</SubmitData>
  <Created>dd-mmm-yy</Created>
 </Environment>
 <Module>
    :
    :
 </Module>
</SerenaExecutableFootprint>
```

Where:

- `prod:proj` is the name of the product and project from where the build was run.
- `ccccc` is the name of the build configuration that was used to build the target.
- `dddddd` is the description of the configuration.
- `lllllll` is the MOVS logical node name. For more information, look at the result of building code with the supplied templates.
- `stem` is a series of MVS qualifiers not including the final type (as used when specifying an area).
- `area name` is the name of the area where the build was performed.
- `dimuser` is the name of the Dimensions user who ran the build.
- `platformuser` is the user ID used for the build on the platform (often the area user ID).
- `job` is the number of the job that was submitted.
- `dd-mmm-yy` is the date the job was run.

The <Module> section is included once for each fully-footprinted program object and has the following format:

```
<Module>
 <Name>LOAD/module.LOAD;v</Name>
 <ItemSpec>product:Item spec.A-EXE;v</ItemSpec>
 <LibraryFileName>module name.load/v</LibraryFileName>
 <Dep>
  :
  :
 </Dep>
 :
 :
</Module>
```

Each <Dep> section has the following contents:

```
 <Dep>
  <Name>type/wsname.type;v/Name>
  <ItemSpec>prod:itemspec.A-SRC;v</ItemSpec>
  <LibraryFileName>name In ItemLibrary/v</LibraryFileName>
 </Dep>
```

Configuring Build Footprinting on MVS

MAINFRAME

The footprint CSECT has the following format:

```
CL24'SERENA FOOTPRINT'
AL4(starttableaddress,entrysize,addressoflastentry)
```

For more information contact Serena support.

Appendix D

Dimensions Build Utility Programs

The Primary Build Execution Monitor	240
The Secondary Build Execution Monitor	244
MDHLLNK0	251
bldcomms	256
Loading Multiple Members	257

The Primary Build Execution Monitor

NOTE For information about the templates and template symbols referred to below, see the *Developer's Reference*.

On mainframe platforms the Primary Batch Execution Monitor (PBEM) is a utility that controls, monitors, and reports results for builds that are initiated by Dimensions Build. On distributed platforms the PBEM run scripts for builds that are initiated by Dimensions Build. When running in a distributed environment the PBEM uses the templater to synchronously start sub tasks.

The PBEM receives input data from Dimensions Build in XML format in a Build Request Definition (BRD) file. The BRD file contains global information and a list of build steps to be executed. The BRD file is arranged as a symbol table, and a collection of sub-symbol tables for each build step. The BRD initiates, and communicates with, sub tasks to execute the build steps. Steps may have dependencies between them; the PBEM is responsible for managing the step dependency analysis and only starts steps when their dependent steps have completed.

The PBEM sends output data to the build server, via a SOAP utility, when:

- Processing starts.
- A step is cancelled.
- A generalized build step ends.
- A build completes.

The generalized PBEM flow is as follows:

- Reads the BRD file.
- Constructs scripts for executing each individual step.
- Examines targets to decide whether to skip steps based on source dependency analysis.
- Controls the execution of steps and decides whether to execute steps synchronously or asynchronously.
- Coordinates the return of results to the build server.

In normal operations the PBEM is initiated by REXEC processing in the Dimensions listener service, however you can also run it as a stand alone application.

Running the PBEM

To run the PBEM on distributed platforms type `pbem` at a command prompt. On MVS you can initiate the PBEM in a batch job using the template `MDHBPBM0`.

PBEM Parameters

Use the following parameters on distributed and mainframe platforms:

IMPORTANT! Apart from the `-m` and `-M` parameters, all other parameters are not case-sensitive.

Parameter	Description
`-a`	Specifies that the default allocator will be used.
`-b <brdname>`	Specifies the name of the BRD file to be processed.
`-c <number>`	Sets the maximum concurrency allowed for this execution where `<number>` is a decimal number. 0 or 1 force no concurrency and any template requests for concurrent execution are ignored. If you use a number that exceeds the maximum number of queued connection requests that TCP/IP can accommodate, processing errors will occur.
`-d <symbol> <value>`	Sets `<symbol>` to the value that you specify for `<value>`. Use a pair of double quotation marks ("") to surround values that include spaces. For example: `-d DMFOO "last used"`
`-i`	Dumps symbol tables at strategic points.
`-l`	Traces director processing.
`-M`	Traces storage allocations and frees.
`-m`	Traces message management logic.
`-n`	Ignores the symbol DMSERVER in a BRD file so that a BRD can be run standalone.
`-p`	Traces parse processing.
`-q`	Causes queues to be dumped at strategic points.
`-t <output spec>`	Specifies where to deliver the trace output. Use $ for stdout. On MVS you can use DD:DDNAME.
`-v`	Prints versions of components.

Dimensions Configuration Symbols

For information about the symbols in the Dimensions configuration file that affect the PBEM, see Appendix B, "Dimensions Build Configuration Symbols".

Example JCL for Running the PBEM

```
//STEP200 EXEC PGM=MDHLPBEM,
// PARM='POSIX(ON),ENVAR("_CEE_ENVFILE=DD:MDHPDIMV")/'
//STEPLIB  DD  DISP=SHR,DSN=MDH.V1010.MDHLLIB
//         DD  DISP=SHR,DSN=MDH.V1010.MDHLLPA
//MDHPTRCE DD  SYSOUT=*
//MDHPDIMV DD  DISP=SHR,DSN=MDH.DIM671.PARM(MDHTDIMV)
//CEEPRINT DD  SYSOUT=*
//MDHPPRNT DD  SYSOUT=*
//MDHPBRD  DD  PATHOPTS=(ORDONLY),
//             PATH='/tmp/my_brd.xml'
//MDHPPARM DD  *
-t DD:MDHPTRCE -q -l
-b DD:MDHPBRD
-c 2
/*
```

In the example above DD NAME is used as follows:

DD Name	Description
MDHPBRD	An optional name that you specify in the -b switch. Can be a USS file or a PDS member.
MDHPDIMV	The file that is used for the Dimensions environment variables. The name must match the name specified in the _CEE_ENVFILE control.
MDHPPARM	The input file used to extend the 'PARM=' statement. The portion after the forward slash '/' in the 'PARM=' statement is processed first, followed by the controls in columns 1-72 of the stream pointed to by MDHPPARM.
MDHPPRNT	The data set for standard output.
MDHPTRCE	The data set for trace output. You can change the DDNAME by altering the value for -t at the command line.

Overriding the PBEM Host Name

You can override the PBEM host name in environments where the main system is accessed via a Dynamic VIPA. This is a virtual host name that can resolve to any one of a collection of tightly coupled systems running on logical partitions (LPARs) of an OS/390 Central Complex (CEC). In this type of environment the host name returns the name of the Dynamic VIPA and not the name of the system (LPAR) on which the PBEM is running. However, some tasks, such as the Secondary Build Execution Monitor (SBEM), require access to a specific PBEM running on a specific system. Therefore, a method is required to directly address a particular LPAR. Use the following optional symbols in the mainframe Dimensions CM `dm.cfg` configuration file to translate the host name:

Symbol	Notes
DM_HOSTNAME_START <value>	Must be one of SMCASID, SYSNAME, LPAR, or HOSTNAME. All other values are used as the starting host name.
DM_HOSTNAME_KEYn	Checks the values specified for 'n', for example, n=1, n=2, and so on. Used as a match against the key retrieved in the previous symbol. Stops processing when there is no match or symbol.
DM_HOSTNAME_MAPn	Replaces the value KEYn if it is present.
DM_HOSTNAME_PATTERN	The pattern is scanned and %k is replaced by the current key value. At the end of this process the resulting value is used as the host name. If it is a name the DNS is searched. If it is an IP address it is used directly.

For SMCASID

Read the contents of location 16 (decimal) from the current address space. This points to a structure called the CVT. At offset 196 there is a four byte pointer to the SMCA. This structure has a four byte field at an offset of 16 decimal, which is the SMCASID or SMFID for the system. Use this four byte value. It is not null terminated but will need to be stripped of any trailing spaces.

For SYSNAME

Use the function `_uname` to obtain the field node name. This field is `&SYSNAME` in the IPL parameters. See the *IBM C/C++ Runtime Library Reference* for more information on this call.

For LPAR

Call the function `mvs_getlpar` in `libfilesys (mvs_lpar.c)` and use the value returned as the key.

Examples:

To replace the host name with an IP address:

```
DM_HOSTNAME_START 192.168.4.15
```

To start with the LPAR name and convert it to a suitable name:

```
DM_HOSTNAME_START LPAR
DM_HOSTNAME_KEY1 VM02
DM_HOSTNAME_VALUE1 D00A.SERENA.COM
DM_HOSTNALE_KEY2 VM01
DM_HOSTNAME_VALUE2 D00B.SERENA.COM
```

To add a domain onto the end of the SMFID:

```
DM_HOSTNAME_START SMCASID
DM_HOSTNAME_PATTERN %k.serena.com
```

Other configurations are possible depending on the system you are using.

The Secondary Build Execution Monitor

MAINFRAME

The Secondary Build Execution Monitor (SBEM) is an MVS only generalized batch execution utility. The SBEM reads and executes JCL-like syntax. It also processes comments of the form `//*SBEM` to control monitoring and error returns. You can run the SBEM in a JCL stream of its own using the MDHBSBM0 template, or from the USS command line. Running the SBEM from USS is useful when you want to execute JCL synchronously, for example, to populate a build area.

The current implementation of the SBEM is suitable for builds. The SBEM parses the whole JCL stream before initiating any work. It then executes the stream step by step, wrapping each executed program with calls to the SVC monitoring facility. At the end point(s) it replies to the PBEM with a task status.

There is no limit to the number of statements that you can code.

For details about using SBEM directives in build templates see the chapter *Build Templates for MVS Platforms* in the *Developer's Reference*.

SBEM Parameters

Parameter	Default value	Purpose
`-a <address>`		Specifies the address (TCP/IP or DNS name) of the PBEM.
`-b <project name>`		Specifies the name of the build project.
`-e`		ABENDs if allocation fails.
`-f`		ABENDs in result processing.
`-g "pattern string"`		Overrides the temporary file name pattern string, and the default, specified by the build option DM_MVS_SBEM_TEMPNAME. See page 230.
`-I <filespec>`	`DD:MDHSIN`	Specifies the location of the input file. Use $ for stdin.
`-j`		This run is to fail if not APF authorized.
`-k`		ABENDs at MDHLCOMP time
`-m`		Traces message processing.
`-n`		Returns information messages to the server.
`-o <filespec>`	`DD:MDHSPRNT`	Specifies where to deliver standard output. Use $ for stdout.
`-p <port number>`		Specifies the port for the PBEM connection.
`-r`		If processing fails, returns a non-zero return code.
`-s`		Traces storage.
`-t <filespec>`	`DD:MDHSTRCE`	Specifies where to direct trace outputs.
`-v <SVC number>`		Specifies the number to use for the Watcher SVC.
`-w <instream file mask>`	`DD:MDHSWORK`	Specifies a library to be used for instream data streams. This specification can include '%d' to identify the actual member to use. The SBEM generates a different name for each instream member. **Note**: If you are using the SBEM from USS, the work data set must be of the form: `-w "'//<data set name>(S%05d)'"` where <data set name> is either a pre-existent data set or is allocated by the SBEM prior to its first use.

Parameter	Default value	Purpose
-y <n>		Specifies the allocation number after which the SBEM fails (the allocator returns NULL).
-z n		Specifies the deallocation number after which the SBEM fails.

Example JCL for Starting the SBEM

```
//STEP100 EXEC PGM=MDHLSBEM,DYNAMNBR=1500,
// PARM='POSIX(ON),TRAP(OFF),HEAPCHK(OFF),ENVAR("_CEE_ENVFILE=DD:DV")/'
//STEPLIB  DD  DISP=SHR,DSN=MDH.V1010.MDHLLIB
//         DD  DISP=SHR,DSN=MDH.V1010.MDHLLPA
//MDHSPARM DD  *
-a MVSnode
-p 4059
-b "BUILD"
-t DD:MDHSTRCE
/*
//*        main execution summary goes in here
//MDHSPRNT DD  SYSOUT=*
//*        only used if exit tracing is turned on
//MDHOUT   DD  SYSOUT=*
//*        trace from sBem
//MDHSTRCE DD  SYSOUT=*
//*        Environment variables for Dimensions listener
//DV       DD  DISP=SHR,DSN=MDH.DIM671.PARM(MDHTDIMV)
//*        Common BOM library for this run
//MDHBOM   DD  SYSOUT=*
//*        SYSIN data gets copied here
//MDHWORK  DD  DISP=(NEW,CATLG,CATLG),
//         DSN=&&TEMP01,
//         UNIT=SYSDA,DSNTYPE=LIBRARY,
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PO),
//         SPACE=(TRK,(20,20,0))
//*        BOM template comes from here
//MDHTMPLT DD  DISP=SHR,DSN=MDHDEV.ISPF.TEMPLATE
//*        input JCL for execution - expanded script
//MDHSIN   DD  DATA,DLM='++'
<JCL stream>
++
```

The DD NAMEs are typically used as follows:

DD NAME	Purpose
MDHBOM	The data set where the bill of materials is produced. This bill of materials can be passed back to the build server to update relationships in the SCM configuration.
MDHOUT	The data set for SVC debugging output. Only used if you specify the -t parameter.

DD NAME	Purpose
MDHSIN	The data set for the SBEM JCL primary input. Can be one of the following: ■ instream ■ DATA with DLEM ■ from a member
MDHSPRNT	The data set for the SBEM primary output; it lists the JCL being executed and identifies any statements that have an error. Also contains brief messages about the progress of the individual steps being executed.
MDHSTRCE	The data set for trace output. Use the -t parameter to specify another name.
MDHTMPLT	Points at the template library that is used to format the bill of materials. The bill of materials is generated based on the MDHBBOM0 template.
MDHWORK	The container used to hold instream data until it is needed. Instream data must be in 80 byte records.
MSHSPARM	The extension of the 'PARM=' statement. Only columns 1-72 are used.

NOTE

- The SBEM implements a subset of JCL constructs that cover most of the basic syntax. If a feature that you require is missing, submit an enhancement request via the Serena support web site.
- If you use the SBEM to run a program that uses OMVS services (for example, FTP) you must specify POSIX(OFF) instead of POSIX(ON) in the PARM to MDHLSBEM.

Starting the SBEM from USS

You can also start the SBEM from a USS prompt, which can be useful when you use scripts. For example:

```
export  DMWORKOPT=-w "//'MY.DATASET(S%%05d)'"
#
# Use the common dimensions profile
#
. /serena/instance/.dmprofile
#
# - -r returns the rc so that the script has the execution rc returned
# - -o $ puts the sBem print to stdout
# - -w identifies a dataset pattern (e.g. 'FOO.A.CNTL(S%%05d)') for data
    storage
#
sbem -i $ -o $ -r $DMWORKOPT <<EOF
jcl goes here
EOF
export rc = $?
exit $rc
```

There is a similar template, `sbem_start.sh`, in the instance templates library that allows JCL to be sent as part of an REXEC and executed synchronously.

SBEM JCL Syntax

The SBEM reads 80 byte records off the input stream and interprets them as JCL.

Guidelines and Restrictions

The following guidelines and restrictions apply:

- Continuation rules are the same.
- JOB cards are skipped.
- There is no provision for references, PROC/PEND, symbolic replacements, or IF/ELSE/ENDIF.
- Instream data is supported provided that you have specified a work file and the instream data is blocked 80.
- Temporary files are supported.
- If the SBEM is running a program that dynamically allocates data sets, and ABENDS or does not close the DDNAME, and the SBEM then runs another program that also uses this DDNAME, the DDNAME is used again by the target program as it does not know to deallocate or close the data set at the ABEND, or if the program fails to do so.

Supported JCL Syntaxes

The following JCL language syntax is supported:

Handles

Parameter	Description
DD	Handles this
EXEC	Handles this
JOB	Skips over this

No other operations are supported.

EXEC

Parameter	Description
TIME=	Not implemented.
PARM=	Implemented.
REGION=	Not implemented.

DD

Parameter	Values/Notes
BLKSIZE=number	
DATA	
DATACLAS=id	

Parameter	Values/Notes
DCB=	Not implemented. The keywords that can be in () after DCB are supported as ordinary DD keywords, for example, LRECL, BLKSIZE, etc.
DDNAME=	Not implemented
DISP=(initial[,success[,fail]])	■ Initial: SHR/NEW/OLD/MOD ■ Success: PASS/KEEP/CATLG/ UNCATLG/DELETE ■ Fail: KEEP/CATLG/UNCATLG/DELETE
DISP=initial	
DLM='cc', DLM=qq	
DSN/DATASET='datasetname'/ datasetname/ 'datasetname(member)'/ datasetname(member)	
DSNAME=	Not implemented.
DSNTYPE= LIBRARY/PDS/FIFO/HFS	
DSORG= PS/PSU/DA/DAU/IS/ISU/PO/POU/CX/GS/TR/TX/MQ/CQ/VSAM	
DUMMY	Implemented
FREE= CLOSE/END	
LRECL=number	
MGMTCLAS=id	
PATH=	Not implemented.
PATHOPTS=	Not implemented.
RECFM=	String of letters that includes M, R, A, S, B, D, T, V, F, U
SPACE=(aunit,(prim,sec[,dir])) aunit - CYL/TRK/BLOCKS	
STORCLAS=id	
SYSOUT=	* letter
UNIT=devtype	No provision for ((ename,count)) **Note**: For Dimensions CM 2009 R1 and later, this clause can now refer to a unit number as well as an esoteric name.
VOL=SER=volume	Only this format is supported to enable use of non-SMS data sets.

Changes to the SBEM in Dimensions CM 10.1.2

The following changes were introduced to the SBEM in Dimensions CM 10.1.2 (and later) to make it more flexible:

- Temporary files are now possible. These are created under the userid that is running the step and are named `userid.MDHSBEM.unique.Tnnnnnn.tempname.T`. The files are cleaned up at the end of the job if the JCL does not do so.
- The PARM can now be split, so `PARM=('Foo','Bar')`, `PARM=(A,B,C)`, and similar constructs will work.
- The SBEM is now linked AC(1). If the program library is also authorized (either by adding the library to the linklist or by issuing the appropriate commands via the console to authorize the program) the SBEM will start authorized. However, if you do not have access to the resource `SERENA.SBEM.RUNAUTHORIZED` in class FACILITY at the level READ or better, the SBEM will turn off authorization early in its execution. If you do have access, the SBEM will start authorized steps. Therefore, the JCL stream can include invocations of IKJEFT01 or similar.
- DATA,DLM=QQ now works.

MDHLLNK0

MAINFRAME

Use the MDHLLNK0 utility before the link step to:

- Remove duplicate include statements from a link stream.
- Insert the footprint CSECT into a link deck.
- Use OPEN/CLOSE on members; you can use it to determine source dependencies that the link editor would see. The link editor uses DESERV, which the WATCH facility does not support.

Use MDHLLNK0 in the templates MDHBLNK0 and MDHBLNK1.

General Parameters

Parameter	Description
`-f`	Includes the footprint CSECT.
`-t <filename>`	Enables tracing to the file that you specify. You can use DD:FOO.
`-z`	Returns the code RC=0 even if some of the included files were not found.

Output Control Parameters

Parameter	Description
`-u<n>`	Controls what action is taken when a linkage editor "NAME" control card is found in the input: ■ `-u0` No action ■ `-u1` The output is opened only for reading. ■ `-u2` (Default) The output is opened for reading, and if that fails, is opened for writing. This action allows the SBEM to detect the targets in all cases. However, you may not require this behavior in some builds.

Passthrough Control Parameters

Control cards can be read with one of the following options:

Option	Description
C	Control statements.
D	Dependency INCLUDE statements.
I	INCLUDE statements.
X	Executable OBJ code (or LOAD/NCAL) data.

These options control what is written to the output file MDHFLAT. The use of this output file is optional and you can pass the original input into the linker.

Option I- INCLUDES

INCLUDES for non-dependency files are always copied to MDHFLAT as they are essential for link processing. For example:

- `INCLUDE OBJ(MODULE)`
- `INCLUDE NCAL(MODULE)`
- `INCLUDE LOAD(MODULE)`

Option D- DEPENDENCY INCLUDES

INCLUDES for dependency files are not generally needed by the linker once all the real INCLUDES that they contain have been flattened out. However, you can request them as follows:

- `-pd0`: does not include dependency includes.
- `-pd1`: (default) includes dependency includes.

For example:

- `INCLUDE DEP(MYDEPS)`
- `INCLUDE OBJ(MYDEPS)`

The nature of the include is decided by the data that the member contains. A member that has no executable code, but does have other INCLUDE statements, is considered to be a dependency file.

Option C - Control Cards

Other control cards may be encountered and you can optionally include them in the output:

- `-pc0`: does not output control cards.
- `-pc1`: (default) outputs control cards

Example:

```
NAME FOOBAR(R)
```

Option X - Executable Code

There are three methods that you can use to handle modules that mix executable (OBJ) records with control statements:

- `-px0`: does not output the OBJ code, but continues scanning for embedded control cards (if the PDS is FB-80).
- `-px1`: outputs the OBJ data directly into MDHFLAT. This produces 'instream' OBJ records for the linker.
- `-px2`: (default) when an x record is encountered, stops scanning for includes, and marks this as a non-dependency file.

Order of Output Lines

You can optionally output the INCLUDE line before or after the data that the included file contains. If the control statements refer to particular modules, ordering is important, and -po1 (the default) is recommended. For example, some control statements refer to the CSECT that comes next in the data stream.

- `-po0`: uses the recursive order (depth first). The body will be expanded first.
- `-po1`: (default) uses the normal order. The INCLUDE line comes before the body.

Examples:

Content of the file MODA:

```
A-START
 INCLUDE OBJ(MODB)
A-END
```

Content of the file MODB:

```
B-START
 INCLUDE OBJ(MODC)
B-END
```

Content of the file MODC:

```
C-START
C-END
```

Content of MDHFLAT using -po0:

```
A-START
B-START
C-START
C-END
   INCLUDE OBJ(MODC)
B-END
   INCLUDE OBJ(MODB)
A-END
   INCLUDE OBJ(MODA)
```

Content of MDHFLAT using -po1:

```
      INCLUDE OBJ(MODA)
   A-START
      INCLUDE OBJ(MODB)
   B-START
      INCLUDE OBJ(MODC)
   C-START
   C-END
   B-END
   A-END
```

Example JCL

The following example JCL is similar to that generated by the templates MDHBLNK0 and MDHBLNK1.

```
//PRELINK  EXEC PGM=MDHLLNK0,
//  PARM='POSIX(OFF),TRAP(OFF)/-t DD:MDHTTRC %PRELKOPT. -u0'
//STEPLIB   DD  DISP=SHR,DSN=MDH.V1010.MDHLLIB
//          DD  DISP=SHR,DSN=MDH.V1010.MDHLLPA
//OBJECT   DD  DISP=SHR,DSN=LIB1.DEV.OBJ
//         DD  DISP=SHR,DSN=LIB1.UT.OBJ
//         DD  DISP=SHR,DSN=LIB1.ST.OBJ
//         DD  DISP=SHR,DSN=LIB1.REL.OBJ
//LNKLIB   DD  DISP=SHR,DSN=LIB1.DEV.LNKLIB
//         DD  DISP=SHR,DSN=LIB1.UT.LNKLIB
//         DD  DISP=SHR,DSN=LIB1.ST.LNKLIB
//         DD  DISP=SHR,DSN=LIB1.REL.LNKLIB
//SYSLIN   DD  DISP=SHR,DSN=LIB1.DEV.SYSLIN(LMODULE)
//DEPENDCY DD  DISP=SHR,DSN=LIB1.DEV.DEPENDCY
//         DD  DISP=SHR,DSN=LIB1.UT.DEPENDCY
//         DD  DISP=SHR,DSN=LIB1.ST.DEPENDCY
//         DD  DISP=SHR,DSN=LIB1.REL.DEPENDCY
//IMPORT   DD  DISP=SHR,DSN=LIB1.DEV.IMPORT
//         DD  DISP=SHR,DSN=LIB1.UT.IMPORT
//         DD  DISP=SHR,DSN=LIB1.ST.IMPORT
//         DD  DISP=SHR,DSN=LIB1.REL.IMPORT
//         DD  DISP=SHR,DSN=CBC.SCLBSID
//         DD  DISP=SHR,DSN=CEE.SCEELIB
//SYSLMOD  DD  DISP=SHR,DSN=LIB1.DEV.LOAD(LMODULE)
//MDHFLAT  DD  DISP=SHR,DSN=LIB1.DEV.MDHFLAT(SMODULE)
//MDHLPRNT DD  SYSOUT=*
//MDHTTRC  DD  SYSOUT=*
```

The example JCL above uses the following DDNAMEs:

DD Name	Description
DDNAME	Other DDNAMEs are required if there are INCLUDE statements that reference them.
MDHFLAT	The data set for the output stream for the link editor.
MDHLPRNT	The data set for messages.
MDHTTRC	The data set for trace. Use the -t parameter to change this name.
SYSLIN	The data set for the input stream.

bldcomms

The bldcomms utility communicates from a build step back to the PBEM or a build server. You can run bldcomms from the Windows, UNIX, or USS command-line prompts, or from MVS batch.

To run bldcomms issue this command:

```
bldcomms monitormessage <node> <port> "<content of message>"
```

where:

Parameter	Description
"<content of message>"	Messages have two possible formats: ■ I=<step num> T=<timestamp> M=<qprefix messageq> ■ R=<num> I=<step num> T=<timestamp> where: ■ <step num> is the step number that identifies the BRD step that this message relates to (in templates use the symbol %DMSTEP.). ■ <timestamp> is the timestamp, in the format yyyyttddhhmmssuuuuuu, that is constructed by the PBEM as part of template processing and is checked to ensure a match. You can also use '**********' which always matches. ■ <qprefix messageq> is as follows: • 'q' is a quotation character. • 'prefix'is the prefix of a message. The last letter of the prefix message describes the severity of the message. E and W are always delivered to the build server, other levels (T, I, and D) are not. • 'message' is the content of the message. For example: "M='JMH001W This is a test message'" ■ <num> is the return code (an integer).
<node>	Specifies the node number of the application expecting the message, typically the PBEM. In a template use the symbol %DMPBEMNODE.
<port>	Specifies a randomly generated port number used by the PBEM to communicate with its started tasks. In a template use the symbol %DMPBEMPORT.

Using bldcomms on MVS

MAINFRAME

You can also use `bldcomms` on MVS, for example, with an Openmake build. The MVS version of bldcomms is `MDHBLCM`.

The following JCL example is for the first `<content of message>` format:

```
//STEP100  EXEC    PGM=MDHLBLCM,
// PARM=('POSIX(ON)',
//       '/monitormessage node port "R=&RC',
//       ' I=%DMSTEP. T=%TIMESTAMP."')
//STEPLIB  DD DISP=SHR,DSN=MDH.V1010.MDHLLIB
//         DD DISP=SHR,DSN=MDH.V1010.MDHLLPA
//SYSPRINT DD      SYSOUT=*
```

The following JCL example is for the second `<content of message>` format:

```
//STEP100  EXEC    PGM=MDHLBLCM,
// PARM=('POSIX(ON)',
//       '/monitormessage %DMPBEMNODE. %DMPBEMPORT. "I=%DMSTEP.',
//       ' T=%TIMESTAMP. M=''JMH001W This is a test message'''")
//STEPLIB  DD DISP=SHR,DSN=MDH.V1010.MDHLLIB
//         DD DISP=SHR,DSN=MDH.V1010.MDHLLPA
```

In the example above the `M=` clause has a single quotation mark (') but as it is inside a PARM statement it must have a double quotation mark (`''`). Note also that the comma (,) after DMSTEP appears in the message to the PBEM but is ignored because the parameter is converted to an integer. So you must ensure that the comma does not occur somewhere where it is not handled.

There is no provision for very long messages.

Loading Multiple Members

MAINFRAME

To load multiple members into a Dimensions database use the JCL stream in the MDHJLOAD data set. MDHJLOAD is located in the following data set:

`MDH.V1031.SUPP.EXAMPLES.CNTL`

Instructions for customizing and using MDHJLOAD are embedded in comments in the JCL, and the core is written in REXX.

Appendix E

Dimensions Build Security

Phase 1: REXEC	260
Phase 2: dmlibsrv	260
Phase 3: The PBEM	260
Phase 4: The SBEM	261
Phase 5: Output Collection	261
Dimensions and Build Area Accounts	261
The Default Userid	262
Dimensions Build Area Security	262
Using Build Areas Outside Dimensions Build	262

Phase 1: REXEC

A build starts when the Dimensions CM server 'dmappsrv' process sends an REXEC to a build node. You can also manually start a build using the REXEC command.

- If you execute the REXEC command manually, first use the AUTH command to authorize access to the build node, or use the /USER=<build area user> and /PASS=<build area password> options on the REXEC command.
- If Dimensions CM starts a build, the security credentials come from the AREA definition of the deployment area. These credentials are used in the same way as the REXEC options /USER and /PASS. If the area does not have credentials (which is an invalid setup for a deployment area), you may be prompted to enter the userid and password at build time. You can also issue an AUTH command prior to the BLD command.

Phase 2: dmlibsrv

The first visible result of the REXEC and AUTH commands is a process called 'dmlibsrv' on the remote machine. 'dmlibsrv' runs with the operating system security credentials of the build area. If you are a UNIX superuser, you can see this process and check which user it is using, by using the 'ps -elf' command at the UNIX prompt. From SDSF on MVS you can see this process from the 'PS' display.

The process 'dmlibsrv' now runs the build. All actions are performed from this user.

Phase 3: The PBEM

On UNIX, the PBEM (Primary Batch Execution Monitor) runs as a normal program (pbem.exe) using the current security credentials.

On MVS a batch job is submitted. The job runs under the security of the 'dmlibsrv' process.

NOTE You can change the job card. For example, you can set up RACF surrogate authority, but this happens outside of Dimensions Build, and also applies to non-build jobs. The process is similar to the build user logging in to TSO/ISPF, using the editor on some JCL, and then typing SUBMIT. The job would probably run as the logged in user. However, it may run as someone else. For example you may have put USER=XXX in the job card. You may also have a JES exit, or a RACF rule that says jobs matching a certain standard (for example JOBNAME), should run as a particular user.

Jobcards

You can customize the jobcards that build steps and the controlling PBEM use by changing their templates. For example, for a normal build step, the jobcard is found at the top of the SBEM template, TEMPLATE(MDHBSBEM0). The jobcard for the PBEM is in TEMPLATE(MDHBPBM0). There are many template expansion variables available to allow a flexible scheme for naming the jobs. For details of the template variables see *The Templating Language and Processor* chapter of the *Developer's Reference*

NOTE

- Changing these jobcards does not in itself change security in any way, unless the site has some additional RACF rules relating to jobnames and users.
- Care is needed to ensure that whatever jobname scheme is chosen, the PBEM and build steps can run together. For example, a build step cannot be given the same jobname as a PBEM controller, or a dealock situation will result.

Phase 4: The SBEM

The SBEM (Secondary Build Execution Monitor) runs under the same authority as the PBEM batch job. The SBEM, as part of its build execution activities, attaches some programs, such as the COBOL compiler. The attached programs run under the same user as the SBEM.

Phase 5: Output Collection

When a build is complete, the related objects can be added to Dimensions CM. This process uses the build area credentials on the remote node, and the credentials of the user logged in to Dimensions CM (either the user logged into Dimensions Build, or the user logged in to a Dimensions client to issue a BLD command).

Scheduled Builds

When you set up a scheduled build you may need to specify the password for each user that requires access to the build area. This applies if you specified that a password is required at runtime when you attached a build area to a build configuration.

Dimensions and Build Area Accounts

For a remote node the Dimensions userid does not matter as the REXEC always supplies the build area credentials.

NOTE There are variables of the form DMXXXXX that have Dimensions information but these do not affect security. These variables exist to enable a template to create correctly named files and logs. The template writer can decide whether to use these variables, or use some other naming standard. For example, a mainframe template could create a data set called "DMSYS.TEMP.XXX", where DMSYS is an uppercase version of the Dimensions user. However, this is just a name. The process will not be run as DMSYS, unless that is also the credential on the build area. For this to work, the area credentials will require access to create these data sets.

TIP The Dimensions userid and password are typically case sensitive. The RACF user and password are typically case insensitive. JCL dataset names have to be uppercase. To avoid problems, enter the MVS USER and PASSWORD in uppercase. However, if you are referring to the UNIX home directory, use lower case. (there is a template function to do this conversion)

The Default Userid

The Dimensions server initially tries to log in to a remote node using the same credentials as the server account. If you have a MVS RACF account called DMSYS with the same password as the Dimensions account DMSYS, the server will log in as this user. You can switch this behavior off. For information, see the description of the following variables in the *Dimensions for z/OS User's and Administrator's Guide*:

- DM_MVS_REJECT_USERS (in the MVS configuration file).
- DM_SKIP_SERVER_CRED_CHECK (in the server configuration file).

Dimensions Build Area Security

Because a build area definition includes security credentials, you can use Dimensions roles to restrict the use of areas to specific Dimensions users.

Using Build Areas Outside Dimensions Build

You can use build areas on their own outside of Dimensions Build.

When you use an area name directly, the directory that the area specifies is used automatically. For example, assume that MYAREA specifies the location MYNODE::USER.WORKAREA. The following commands achieve the same aim:

- `FI <item> MYNODE::USER.WORKAREA.COBOL(PROG1)`
- `FI <item> MYAREA::COBOL(PROG1)`

The method that uses MYAREA has two advantages:

- It is shorter.
- It logs on automatically using the credentials in the area.

Appendix F

The Build Configuration XML File Format

Introduction	264
Qlarius Sample Application	264
XML Example #1	265
XML Example #2	268
Build Configuration XML File Structure	271

Introduction

This appendix describes the format and usage of the build configuration XML file used by Dimensions Build. After reading this appendix should be able to generate a properly formatted XML file, which you can import into Dimensions Build, using one of the following methods:

- An XML parser with a programming language of your choice.
- A flat file I/O in any programming language.
- By hand using an XML file editor.
- The Dimensions EXPORT command.

Qlarius Sample Application

To help explain the XML format, we will examine a build configuration and export an XML file from the Qlarius Java sample application that currently ships with Dimensions. This will enable you to compare the data displayed in the Build user interface with the same information in the XML file. Since the source files for this sample are already installed with the qlarius_cm database, setting up a build configuration and build area are the only steps that you need to do to start using it. This appendix contains two sample build configurations that you can place into stand alone files and import into Build.

NOTE Please contact your Dimensions administrator if your installation does not contain the Qlarius sample database.

Log in to Dimensions Build and look at the entire Qlarius configuration. Notice that there are two main directories with source code in this sample:

- qlarius\interfaces
- qlarius\utilities

Each of these directories contains various Java sources that can be built into class files. Some of the sources in 'interfaces' depends on sources in 'utilities' so you need to be aware of this when setting up your build.

NOTE

The Id attribute on most of the tags in the configuration file is assigned by Build and is unique to each item on the database and in the file. It is mainly used in a configuration file to maintain the relationships between source, target, and transition items. For example:

- Source A has an ID of 1
- Source B has an ID of 2
- Target AA has an ID of 3

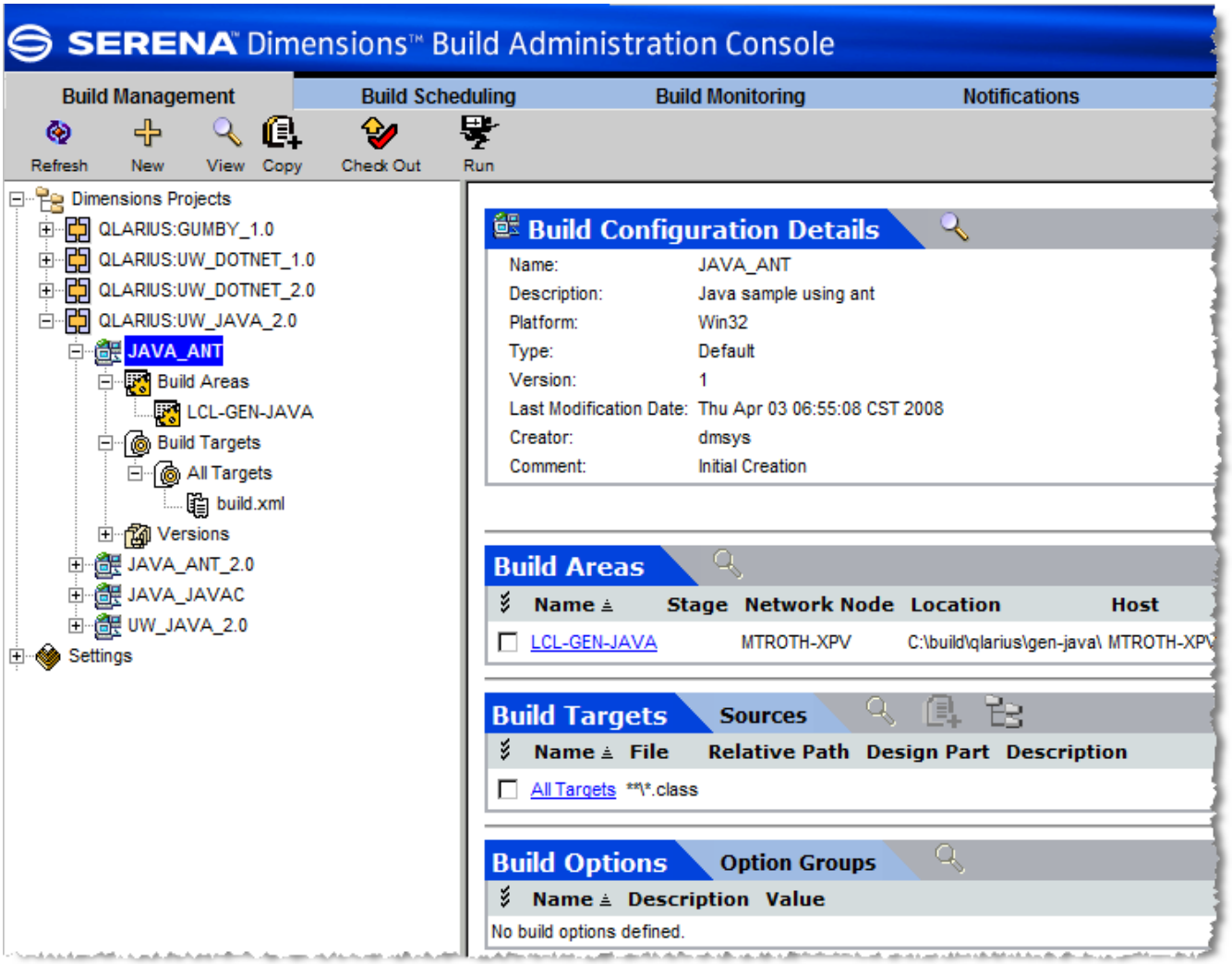
A transition to create Target AA would have an output ID of 3 with two source files of ID 1 and ID 2. The ID numbers are used to connect the source and target files together.

The Id number can be any numeric value as long as each Id is unique to the item it is identifying and all the pieces can be put together. Once the items are imported into the database, they are all assigned new Ids that are unique to the new database. The Id fields

on other items (build areas, etc.) are ignored as they do not provide any relationship information that is required for the import process.

XML Example #1

The screen shot and XML example below are for a Qlarius build configuration called JAVA_ANT.



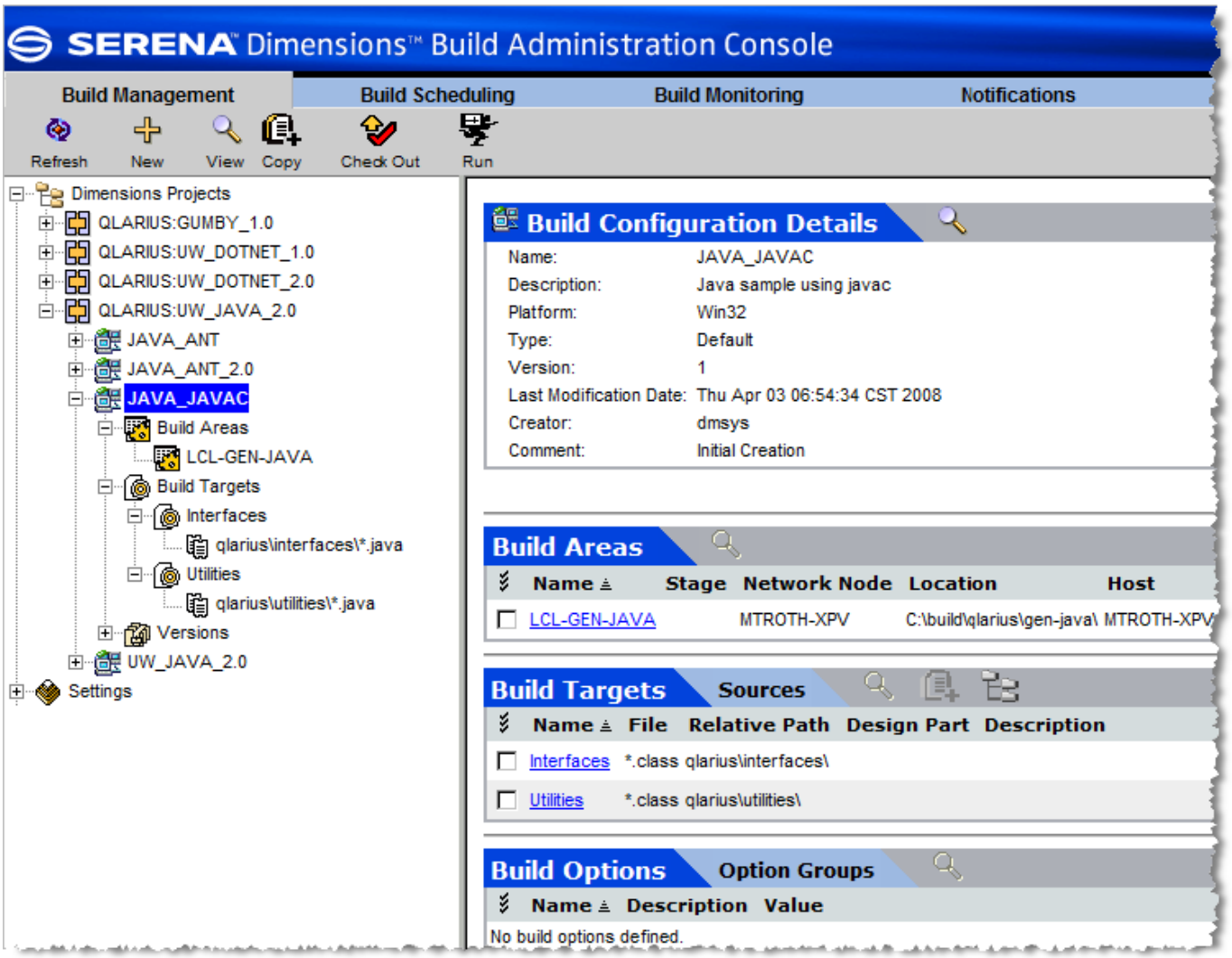
NOTE Each section of the XML example is highlighted in a different color.

```
<?xml version="1.0" encoding="UTF-8" ?>
<BuildConfigurations>
<BuildConfiguration Id="19">
<BuildProject Id="4203131">
<project><![CDATA[QLARIUS:UW_JAVA_2.0]]></project>
<description><![CDATA[Underwriter Desktop Java development project]]></
    description>
</BuildProject>
<name><![CDATA[JAVA_ANT]]></name>
<description><![CDATA[Java sample using ant]]></description>
<Platform Id="1">
<name><![CDATA[Win32]]></name>
</Platform>
<LaunchTimeout>0</LaunchTimeout>
<ExecutionTimeout>0</ExecutionTimeout>
<ConfigurationType Id="1">
<name><![CDATA[Default]]></name>
<code><![CDATA[DEFAULT]]></code>
</ConfigurationType>
<Targets>
<Target Id="128">
<name><![CDATA[All Targets]]></name>
<file><![CDATA[**\*.class]]></file>
<isfinal>true</isfinal>
<isvirtual>false</isvirtual>
</Target>
</Targets>
<Sources>
<Source Id="105">
<fileMask><![CDATA[build-user.xml]]></fileMask>
</Source>
<Source Id="106">
<fileMask><![CDATA[build.xml]]></fileMask></Source>
<Source Id="1339">
</Source>
<Source Id="103">
<fileMask><![CDATA[.classpath]]></fileMask>
</Source>
<Source Id="102">
<fileMask><![CDATA[**\*.java]]></fileMask>
</Source>
<Source Id="104">
<fileMask><![CDATA[.project]]></fileMask>
</Source>
</Sources>
<BuildAreaConfigs>
<BuildAreaConfig Id="28">
<BuildArea Id="4206024">
<name><![CDATA[LCL-GEN-JAVA]]></name>
<host><![CDATA[MTROTH-XPV]]></host>
<location><![CDATA[C:\build\qlarius\gen-java\]]></location>
<user><![CDATA[dmsys]]></user>
<password><![CDATA[EE961583AEC7144D]]></password>
</BuildArea>
<askPasswordAtRuntime>false</askPasswordAtRuntime>
<Options>
```

```
<Option Id="2">
<name><![CDATA[JAVA-HOME]]></name>
<OptionValues>
<OptionValue Id="4">
<value><![CDATA[c:\j2sdk1.4.2_12]]></value>
</OptionValue>
</OptionValues>
</Option>
<Option Id="1">
<name><![CDATA[ANT-HOME]]></name>
<OptionValues>
<OptionValue Id="3">
<value><![CDATA[c:\apache-ant-1.6.5\bin]]></value>
</OptionValue>
</OptionValues>
</Option>
</Options>
</BuildAreaConfig>
</BuildAreaConfigs>
<TransitionScripts>
<TransitionScript Id="143">
<Script Id="59">
<type>1</type>
<scriptName><![CDATA[build_script_ant.bat]]></scriptName>
</Script>
<outputIds>128</outputIds>
<inputSourceIds>106</inputSourceIds>
<transitionType>0</transitionType>
<transitionOrder>0</transitionOrder>
</TransitionScript>
</TransitionScripts>
</BuildConfiguration>
</BuildConfigurations>
```

XML Example #2

The screen shot and XML example below are for a Qlarius build configuration called JAVA_JAVAC.



NOTE Each section of the XML example is highlighted in a different color.

```
<?xml version="1.0" encoding="UTF-8" ?>
<BuildConfigurations>
<BuildConfiguration Id="18">
<BuildProject Id="4203131">
<project><![CDATA[QLARIUS:UW_JAVA_2.0]]></project>
<description><![CDATA[Underwriter Desktop Java development project]]></
    description>
</BuildProject>
<name><![CDATA[JAVA_JAVAC]]></name>
<description><![CDATA[Java sample using javac]]></description>
<Platform Id="1">
<name><![CDATA[Win32]]></name>
</Platform>
<LaunchTimeout>0</LaunchTimeout>
<ExecutionTimeout>0</ExecutionTimeout>
<ConfigurationType Id="1">
<name><![CDATA[Default]]></name>
<code><![CDATA[DEFAULT]]></code>
</ConfigurationType>
<Targets>
<Target Id="126">
<name><![CDATA[Interfaces]]></name>
<file><![CDATA[*.class]]></file>
<relpath><![CDATA[qlarius\interfaces\]]></relpath>
<isfinal>true</isfinal>
<isvirtual>false</isvirtual>
</Target>
<Target Id="127">
<name><![CDATA[Utilities]]></name>
<file><![CDATA[*.class]]></file>
<relpath><![CDATA[qlarius\utilities\]]></relpath>
<isfinal>true</isfinal>
<isvirtual>false</isvirtual>
</Targets>
</Target>
</Targets>
<Sources>
<Source Id="101">
<fileMask><![CDATA[*.java]]></fileMask>
<relPath><![CDATA[qlarius\interfaces\]]></relPath>
</Source>
<Source Id="100">
<fileMask><![CDATA[*.java]]></fileMask>
<relPath><![CDATA[qlarius\utilities\]]></relPath>
</Source>
</Sources>
<BuildAreaConfigs>
<BuildAreaConfig Id="27">
<BuildArea Id="4206024">
<name><![CDATA[LCL-GEN-JAVA]]></name>
<host><![CDATA[MTROTH-XPV]]></host>
<location><![CDATA[C:\build\qlarius\gen-java\]]></location>
<user><![CDATA[dmsys]]></user>
<password><![CDATA[EE961583AEC7144D]]></password>
</BuildArea>
```

```
<askPasswordAtRuntime>false</askPasswordAtRuntime>
<Options>
<Option Id="2">
<name><![CDATA[JAVA-HOME]]></name>
<OptionValues>
<OptionValue Id="2">
<value><![CDATA[c:\j2sdk1.4.2_12]]></value>
</OptionValue>
</OptionValues>
</Option>
<Option Id="1">
<name><![CDATA[ANT-HOME]]></name>
<OptionValues>
<OptionValue Id="1">
<value><![CDATA[c:\apache-ant-1.6.5\bin]]></value>
</OptionValue>
</OptionValues>
</Option>
</Options>
</BuildAreaConfig>
</BuildAreaConfigs>
<TransitionScripts>
<TransitionScript Id="141">
<Script Id="57">
<type>1</type>
<scriptName><![CDATA[build_script_javac.bat]]></scriptName>
</Script>
<outputIds>126</outputIds>
<inputSourceIds>101</inputSourceIds>
<transitionType>0</transitionType>
<transitionOrder>2</transitionOrder>
</TransitionScript>
<TransitionScript Id="142">
<Script Id="58">
<type>1</type>
<scriptName><![CDATA[build_script_javac.bat]]></scriptName>
<scriptContent><![CDATA[set java_home=%java-home. set path=%java-
    home.\bin;%ant-home.;%%path%% javac qlarius\utilities\*.java]]></
    scriptContent>
</Script>
<outputIds>127</outputIds>
<inputSourceIds>100</inputSourceIds>
<transitionType>0</transitionType>
<transitionOrder>1</transitionOrder>
</TransitionScript>
</TransitionScripts>
</BuildConfiguration>
</BuildConfigurations>
```

Build Configuration XML File Structure

The XML file normally has the following sections:

- General Configuration Items
- Targets
- Sources
- Build Areas
- Transitions

NOTE Some of the XML tags described in this section are not in the examples above. However, they are included as they can exist in an XML file depending on the build configuration.

General Configuration Items

The following section at the top of the file describes general information about the build configuration.

XML Example 1	XML Example 2	Description
<BuildConfigurations> ... </BuildConfigurations>	<BuildConfigurations> ... </BuildConfigurations>	Opening and closing tags for entire XML document. All of the other tags in the entire document are contained between these two tags.
<BuildConfiguration Id="19"> ... </BuildConfiguration>	<BuildConfiguration Id="18"> ... </BuildConfiguration>	Opening and closing tags for entire build configuration. A single XML document may contain more than one individual build configuration with each one being contained between these two tags.
<BuildProjectId="4203131"> <project> ![CDATA[QLARIUS:UW_JAVA_2.0]] </project> <description> ![CDATA[Underwriter Desktop Java development project]] </description> </BuildProject>	<BuildProject Id="4203131"> <project> ![CDATA[QLARIUS:UW_JAVA_2.0]] </project> <description> ![CDATA[Underwriter Desktop Java development project]] </description> </BuildProject>	Parent project information. This section consists of: ■ project ID ■ project name ■ description.

(Sheet 1 of 4)

XML Example 1	XML Example 2	Description
<name> ![CDATA[JAVA_ANT]] </name>	<name> ![CDATA[JAVA_JAVAC]] </name>	Build configuration name.
<description> ![CDATA[Java sample using ant]] </description>	<description> ![CDATA[Java sample using javac]] </description>	Build configuration description.
<ProjectRelPath> ABC </ProjectRelPath>		Build configuration path relative to the Dimensions project it is assigned to. This field allows the user to set a build configuration to a particular directory in the overall Dimensions project instead of being anchored on the root of the project.
<PlatformId="1"> <name> ![CDATA[Win32]] </name> </Platform>	<PlatformId="1"> <name> ![CDATA[Win32]] </name> </Platform>	Platform id and name. Valid values for this field are: ID: 1 Name: Win32 ID: 2 Name: AIX ID: 3 Name: HPUX ID: 4 Name: Linux ID: 5 Name: Solaris ID: 6 Name: USS ID: 7 Name: SCO ID: 8 Name: MVS.
<LaunchTimeout> 0 </LaunchTimeout>	<LaunchTimeout> 0 </LaunchTimeout>	Launch Timeout. When launching a build, this tag defines how many seconds the Build server waits before deciding the launch has failed if no communication is received from the remote build node. This value overrides the default value defined in web.xml.
<ExecutionTimeout> 0 </ExecutionTimeout>	<ExecutionTimeout> 0 </ExecutionTimeout>	Execution Timeout. When executing a build, this tag defines how many seconds the Build sever waits before canceling a build job that has not communicated any status. This value overrides the default value defined in web.xml.

(Sheet 2 of 4)

XML Example 1	XML Example 2	Description
<ConfigurationTypeId="1"> <name> ![CDATA[Default]] </name> <code> ![CDATA[DEFAULT]] </code> </ConfigurationType>	<ConfigurationType Id="1"> <name> ![CDATA[Default]] </name> <code> ![CDATA[DEFAULT]] </code> </ConfigurationType>	Configuration type. Is defined in Settings: Build Configuration Types and can be optionally assigned to a configuration.
<Options> ... </Options>		Option to be used for this transition. This is defined by id (unique to database), option name, and option values. If the option values are related to a build tool, the relationship is also defined here.
<OptionGroupIds> # </OptionGroupIds>		Option Group id(s) (unique to database) tied to this transition.
<preScript Id=”285”> <type>1</type> <scriptName> pre-script.bat </scriptName> </preScript>		Configuration pre-script: a script to be executed before all build processing has been started.
<mainScript Id=”286”> <type>3</type> <scriptContent> ant –f build.xml </scriptContent> </mainScript>		Configuration main-script: a script to be executed on a target if no other script has been defined.

(Sheet 3 of 4)

XML Example 1	XML Example 2	Description
<postScript Id="287"> <type>2</type> <ControlledAsset> <binding> I{ QLARIUS:UW_JAVA_2.0}templates /post-script.bat </binding> </ControlledAsset> </postScript>		Configuration post-script: a script to be executed after all build processing is complete.
<cleanScript Id="288"> <type>1</type> <scriptName> clean-script.bat </scriptName> </cleanScript>		Configuration clean-script: a script to be optionally executed before the pre-script.

(Sheet 4 of 4)

Targets

The targets section contains an entry for each target defined in the configuration.

XML Example 1	XML Example 2	Description
<Targets>	<Targets>	Opening tag for the targets section. All targets for this configuration are defined within the open and closing tags.
<Target Id="128">	<Target Id="126">	Opening tag for this particular target that contains the unique id for it.
<name> ![CDATA[All Targets]] </name>	<name> ![CDATA[Interfaces]] </name>	Target name.
<file> ![CDATA[***.class]] </file>	<file> ![CDATA[*.class]] </file>	Target filename.
	<relpath> ![CDATA[qlarius\interfaces\]] </relpath>	Relative path for the target. This is combined with the build area location, the path set on the build configuration (if any), and the directory qualifier set on the target filename (if any).
<isfinal> True </isfinal>	<isfinal> True </isfinal>	Flag to indicate whether the target is a final target.
<isvirtual> False </isvirtual>	<isvirtual> False </isvirtual>	Flag to indicate whether the target is a virtual target.
</Target>	</Target>	Closing tag for a single target.
</Targets>	</Targets>	Closing tag for the targets section.

Sources

The sources section contains an entry for each source defined in the configuration.

XML Example 1	XML Example 2	Description
<Sources> ... </Sources>	<Sources> ... </Sources>	Opening and closing tags for the sources section. All sources for this configuration are defined within these tags.
<Source Id="105">	<Source Id="101">	Opening tag for this particular source that contains the unique id for it.
<fileMask> ![CDATA[***.java]] </fileMask>	<fileMask> ![CDATA[*.java]] </fileMask>	Filename or mask for the source.
	<relPath> ![CDATA[qlarius\utilities\]] </relPath>	Relative path for the source. This is combined with the build area location, the path set on the build configuration (if any), and the directory qualifier set on the source filename (if any).
<buildType> ABC </buildType>		Build type (item format) for the source item. The item format/build type is listed between the opening and closing tag and is utilized when using source items with wildcards in their names.
</Source>	</Source>	Closing tag for a single source.
</Sources>	</Sources>	Closing tag for the sources section.

Build Areas

The Build Areas section contains an entry for each build area defined in the build configuration. Build areas are set up in the Dimensions Administration module and not in Dimensions Build.

When a build area is used in Build, two types of information are used:

- The area itself as defined in Dimensions.
- The specific attributes defined when the build area is added to a build configuration.

NOTE When importing a build area from a file, Build attempts to perform a name match between the build area name in the import file and the build areas in the receiving database. When a match is found, a relationship is made in the configuration to the existing build area. If no match is found, the build area in the import file is ignored and a build area is not created automatically. Therefore, the information related to the build area (node, location, userid, password) is not imported.

XML Example 1	XML Example 2	Description
<BuildAreaConfigs>	<BuildAreaConfigs>	Opening tag for build areas section. All build areas for this configuration are defined within the open and closing tags.
<BuildAreaConfig Id="28">	<BuildAreaConfig Id="27">	Opening tag for this particular configuration build area that contains the unique id for it.
<BuildArea Id="4206024">	<BuildArea Id="4206024">	Opening tag for the Dimensions defined build area that contains the unique id for it.
<name> ![CDATA[LCL-GEN-JAVA]] </name>	<name> ![CDATA[LCL-GEN-JAVA]] </name>	Build area name.
<host> ![CDATA[MTROTH-XPV]] </host>	<host> ![CDATA[MTROTH-XPV]] </host>	Dimensions node containing the build area. Note: information only, not used during the import process.
<location> ![CDATA[C:\build\qlarius\gen-java\]] </location>	<location> ![CDATA[C:\build\qlarius\gen-java\]] </location>	Build area location/path/ relative to the defined node or <host> tag. Note: information only, not used during the import process.
<user> ![CDATA[dmsys]] </user>	<user> ![CDATA[dmsys]] </user>	Security user id relative to the node. Note: information only, not used during the import process.
<password> ![CDATA[EE961583AEC7144D]] </password>	<password> ![CDATA[EE961583AEC7144D]] </password>	Security user password (encrypted) relative to user id. Note: information only, not used during the import process.

XML Example 1	XML Example 2	Description
<askPasswordAtRuntime> False </askPasswordAtRuntime>	<askPasswordAtRuntime> False </askPasswordAtRuntime>	Flag to indicate whether to ask for a password at runtime. This is a configuration specific attribute.
<Options> <OptionId="s"> <name> ![CDATA[JAVA-HOME]] </name> <OptionValues> <OptionValueId="4"> <value> ![CDATA[c:\j2sdk1.4.2_12]] </value> <buildToolId>0</ buildToolId> <ruleId>0</ruleId> </OptionValue> </OptionValues> </Option> <OptionId="1"> <name> ![CDATA[ANT-HOME]] </name> <OptionValues> <OptionValueId="3"> <value> ![CDATA[c:\apache-ant- 1.6.5\bin]] </value> <buildToolId>0</ buildToolId> <ruleId>0</ruleId> </OptionValue> </OptionValues> </Option> </Options>	<Options> <OptionId="2"> <name> ![CDATA[JAVA-HOME]] </name> <OptionValues> <OptionValueId="2"> <value> ![CDATA[c:\j2sdk1.4.2_12]] </value> <buildToolId>0</buildToolId> <ruleId>0</ruleId> </OptionValue> </OptionValues> </Option> <OptionId="1"> <name> ![CDATA[ANT-HOME]] </name> <OptionValues> <OptionValueId="1"> <value> ![CDATA[c:\apache-ant- 1.6.5\bin]] </value> <buildToolId>0</buildToolId> <ruleId>0</ruleId> </OptionValue> </OptionValues> </Option> </Options>	Build options defined on the area, see page 281 for details.
</BuildArea>	</BuildArea>	Closing tag for a single build area.
</BuildAreaConfig>	</BuildAreaConfig>	Closing tag for a single build area configuration.
</BuildAreaConfigs>	</BuildAreaConfigs>	Closing tag for the build configurations section.

Transitions

The Transitions section contains an entry for each transition defined in the build configuration. This is a critical section of the XML document as it ties together how items get built by bringing together the items defined in the sources and targets sections with the transitions that use or generate them.

XML Example 1	XML Example 2	Description
<TransitionScripts>	<TransitionScripts>	Opening tag for the transitions section. All transitions for this configuration are defined within the opening tag and the closing tag.
<TransitionScript Id="143">	<TransitionScript Id="142">	Opening tag for this particular transition that contains the unique id for it.
<Script Id="59">	<Script Id="58">	Opening tag for the script to be used for this transition that contains the unique id for it.
<type> 1 </type>	<type> 1 </type>	Build Script to be used for this transition. This is a defined numeric type value that indicates how the build script was entered or is being used See *page 280* for more information.
<scriptName> ![CDATA[build_script_ant.bat]] </scriptName>	<scriptName> ![CDATA[build_script_javac.bat]] </scriptName>	Build script name.
<outputIds> 128 </outputIds>	<outputIds> 126 </outputIds>	File output(s) of this transition as identified by the id. The definition of items listed here must appear in the targets section of the file. This is very important for piecing together the transitions and targets sections.
<inputSourceIds> 106 </inputSourceIds>	<inputSourceIds> 101 </inputSourceIds>	File input(s) of this transition that are true source files as identified by id. The definition of items listed here must appear in the sources section of the file. This is very important for piecing together the transitions and sources sections.
<inputTargetIds> # </inputTargetIds>		File input(s) of this transition that are generated from other transitions as identified by the id. The definition of items listed here must appear in the targets section of the file. This is very important for piecing together the transitions and targets sections.

XML Example 1	XML Example 2	Description
<Options> ... </Options>		Option to be used for this transition. This is defined by id (unique to the database), option name, and option values. If the option values are related to a build tool, the relationship is also defined here.
<OptionGroupIds> # </OptionGroupIds>		Option Group id(s) (unique to the database) tied to this transition.
<templateId> 0 </templateId>	<templateId> 0 </templateId>	Transition Rule Template id used to create this transition.
<transitionType> 0 </transitionType>	<transitionType> 0 </transitionType>	Expand (value 1) or no-expand (value 0) the wildcard inputs for this transition.
<transitionOrder> 0 </transitionOrder>	<transitionOrder> 2 </transitionOrder>	Build order assigned to this transition.
</Script>	</Script>	Closing tag for a single script.
</TransitionScripts>	</TransitionScripts>	Closing tag for a single transition.
</TransitionScript>	</TransitionScript>	Closing tag for the transitions section.

Scripts

There are three different ways a script can be specified. Each script type is identified by a unique type number and set of tags that contain the details of the type:

- Type 1: script exists in build area

  <scriptName></scriptName>)

- Type 2: script selected from source control

  <ControlledAsset></ControlledAsset>)

- Type 3: script entered in Build UI

  <scriptContent></scriptContent>)

Any script defined in a build configuration will utilize these sets of tags.

Build Options and Option Groups

Build options and option groups can appear in the following sections of the configuration XML file:

- General Configuration Items
- Build Areas
- Transitions

XML Example 1	Description
<Options>	Opening tag for the options defined within a section. All options for the option are defined within the opening tag and the closing tag.
<Option Id="2">	Opening tag for this particular option that contains the unique id for it.
<name> ![CDATA[JAVA-HOME]] </name>	Option name. This is the name used by the build script writer to reference this option.
<OptionValues>	Opening tag for the values for this option. Since an option may contain 1 to many values, this tag encompasses all the possible values for the option.
<OptionValue Id="4">	Opening tag for a single option value that contains the unique id for it.
<value> ![CDATA[c:\j2sdk1.4.2_12]] </value>	Option value.
<buildToolId>0</ buildToolId>	Build Tool id from which the option value was selected.
<ruleId>0</ruleId>	Build Tool rule id from which the option value was selected.
</OptionValue>	Closing tag for a single option value.
</OptionValues>	Closing tag for option values.
</Option>	Closing tag for a single option.
</Options>	Closing tag for entire options section.
<OptionGroupIds> # </OptionGroupIds>	Opening and closing tags for an option group. The numeric values of the option group are listed between the opening and closing tags.

Appendix G

Customizing a Build Server

Customizing a Build Server 284

Customizing a Build Server

This appendix describes parameters that control the function of the build server applet running in the web server. You can customize these parameters to change the behavior of the build server to meet your specific needs. The parameters are located in the following file on the primary build server.

```
<dm_root>/Common Tools/tomcat/8.0/webapps/bws/WEB-INF/web.xml
```

NOTE This appendix does not discuss all the parameters in web.xml, only the most commonly used ones.

```
<init-param>
   <param-name>view.brd.enabled</param-name>
   <param-value>false</param-value>
</init-param>
```

Adds a link in the build launch wizard in the Build user interface that enables you to view the Build Request Definition (BRD) file before submitting the build. This is helpful when you want to check what data is available to your build script and what order the build steps will execute in.

Default: false

```
<init-param>
   <param-name>brd.force.dependencies</param-name>
   <param-value>0</param-value>
</init-param>
```

Forces dependency information from previous builds to be added to the BRD file regardless of platform. By default, only z/OS build requests contain dependency information in the BRD as it is used by the Serena mainframe build tool. This information is omitted from BRD files for other platforms to save space. If you enable this option you could write a custom build script to use this information.

```
<init-param>
   <param-name>session.timeout</param-name>
   <param-value>60</param-value>
</init-param>
```

Specifies the number of minutes after which your build session times out.

Default: 60 minutes

```
<init-param>
   <param-name>build.missing.target.status</param-name>
   <param-value>4</param-value>
</init-param>
```

Specifies the action to be taken if a BRD step contains a target that is not specified in the build configuration. For example, specifying 2 (WARNING) allows the build to continue.

Default: 4 (issue an ERROR and stop the build).

```
<init-param>
   <param-name>build.job.execution.timeout</param-name>
   <param-value>60</param-value>
</init-param>
```

Specifies the number of minutes after which a response of any kind is expected from the PBEM. This information can help you to analyze the abnormal termination of build processes (crashes, hangs, etc). After this period the build job is cancelled. You can change this timeout for each build configuration.

Default: 60 minutes.

```
<init-param>
   <param-name>build.job.execution.start.timeout</param-name>
   <param-value>60</param-value>
</init-param>
```

Specifies the number of seconds after which a response is expected from the PBEM with the 'build start ' event. After this period the build job is cancelled. You can change this timeout for each build configuration.

Default: 60 seconds.

Appendix H

Load Balancing a Build Server

Setting Up Load Balancing 288

Restrictions and Limitations 289

Setting Up Load Balancing

Dimensions Build enables you to balance the loading on your servers by specifying multiple build servers that are different to the default web server.

The primary build server manages the pool of build servers and tells the Dimensions server which build server to use for each build request (BLD or BLDB command). Each build server is utilized in turn in a round-robin as required.

To set up load balancing:

1. Specify the primary build server by adding the variable `DM_BUILDERWS_URL` or DM_WEB_URL to the Dimensions configuration file (dm.cfg), for example:

   ```
   DM_BUILDERWS_URL http://buildserver:8080
   ```

2. Specify a list of secondary build servers in the parameter `build.server.list` in following configuration file:

   ```
   <dm_root>/Common Tools/tomcat/8.0/webapps/bws/WEB-INF/web.xml

   <init-param>
      <param-name>build.server.list</param-name>
      <param-value>http://server1:8080;http://server2:8080</paramvalue>
   </init-param>
   ```

   This parameter specifies the URL paths to secondary build servers that will be used for submitting build jobs. Use a comma or semicolon to separate different URL paths. By default this variable is disabled (commented out).

3. Uncomment the following tag when using load balancing to ensure proper keys on the database:

   ```
   <init-param>
      <param-name>build.use.scbc.sequence</param-name>
      <param-value>true</param-value>
   </init-param>
   ```

   This parameter specifies if a build web service uses a database sequence to generate primary keys (required for environments with multiple builder servers). By default this variable is disabled (commented out).

   Default: false

Restrictions and Limitations

The following restrictions and limitations currently apply:

- There is no user interface to manage the list of build servers. Use the parameters in the web configuration file as described in the section above.
- It is assumed that all the specified build servers are online. A build request will fail if the build server assigned to the current build task is offline.
- Builds that are invoked from the Dimensions Build user interface are always coordinated by the builder server that you are logged into, and do not use load balancing. Only the primary Dimensions clients (desktop, web, ISPF, and command line) use load balancing.

Appendix I

Advanced Build Settings

Using Wait Processing	292
Using Build Roles to Control User Builds	293
Disabling the Automatic Selection of Targets	294

Using Wait Processing

The BLD, BLDB, and REXEC commands support the ability to wait and not return control to Dimensions until the remote task being executed has completed.

Using the BLD, BLDB, and REXEC Commands

Use the following parameters in the BLD, BLDB, and REXEC commands to control wait processing:

`/BATCH and /NOBATCH`

Default: /BATCH

If you specify /NOBATCH, after the BLD or BLDB command has run it waits in turn for each of the started jobs. You can also set the variable DM_MAX_REXEC_WAIT in dm.cfg to specify the longest wait time (in seconds) for the server. If you do not set this variable, the server waits indefinitely until the database is updated with a job completion record. Each job may succeed or fail. A job may also timeout if you have enabled the timeout option.

Alternatively, use the RLIST <job-id> /WAIT command to achieve the same outcome. If there are different jobs being issued, you can achieve a higher degree of parallelism using the RLIST facility. You can also use the dm.cfg variable DM_MAX_REXEC_WAIT with RLIST.

If a BLD or BLDB command starts more than one job, use the above mechanism to wait for all of them.

Using Structured Information Return (SIR)

When a BLD, BLDB or REXEC command is started, the variable DM_JOB_ID is returned containing the job ID in the form R-uid. If a BLD or BLDB command starts more than one job, DM_JOB_ID is the first job started only. In addition, two other variables are also returned for the BLD and BLDB commands, DM_JOB_NAME and DM_JOB_STATII. Both these variables are arrays.

- DM_JOB_NAME has elements with names of the same form as DM_JOB_ID.
- DM_JOB_STATII has elements containing SUCCEEDED, FAILED, BACKGROUND, TIMEOUT, or ERROR.
  - BACKGROUND is only seen when /batch is used.
  - TIMEOUT appears if DM_MAX_REXEC_WAIT has been exceeded.
  - ERROR only occurs if there are database errors encountered trying to obtain the job status record.

If a build job starts a known number of builds, for example three, the syntax looks like this:

```
   BLD ... /BATCH
   SAVE BIG_NAMES DM_JOB_NAME
a:
   RLIST &(BIG_NAMES(0)). /WAIT
   RLIST &(BIG_NAMES(1)). /WAIT
   RLIST &(BIG_NAMES(2)). /WAIT
```

Build waits for all three jobs to complete, and additional processing is done at a: to force a higher degree of parallelism. If you use the Dimensions DTK to initiate the processing, the PcmsApi functions *PcmsGetSymbolInfo* and *PcmsGetStringSymbolValue* can obtain and work with these data items.

Using REXEC

Use the following additional parameters with /WAIT on the REXEC command:

- /CAPTURE

  Generates a one time certificate.

- /BATCH

  - Distributed platforms: choose between synchronous and asynchronous execution of the REXEC job.
  - MVS: the started job is always asynchronous and /NOBATCH does not cause a process to wait.

    For all platforms, if you use RLIST <job-id> /WAIT you have to connect back to Dimensions and issue an RSTAT command to update the final status of the job. Failure to do so will cause RLIST /WAIT processing for this job to wait indefinitely or time out.

NOTE Check that the client session timeout is less than the maximum time you expect to wait for a job or for jobs to complete.

Using Build Roles to Control User Builds

Optionally, Dimensions Build checks which build roles are required at each build stage, and verifies that the user running the build has the appropriate role. If user does not, the build request is rejected. This enables you to control which users can build at which stages.

The naming standard for build roles is as follows:

- Role names begin with BLD_ and end with the stage name.
- Any blanks or spaces in the stage name are converted to an underscore (_).

For example, the stage name SYSTEM TEST requires the role BLD_SYSTEM_TEST.

NOTE The build roles are not mandatory, but if they exist they are required. Therefore, if the role for a stage is not defined, the build is allowed to progress.

Disabling the Automatic Selection of Targets

By default, targets are automatically selected by Dimensions Build. To disable this feature, specify the following variable in the Dimensions configuration file, dm.cfg:

```
NO_BUILD_TARGET_PRESELECT PRODUCT1:PROJECT1, PRODUCT2:PROJECT2 …,
```

Projects that you specify must be separated by a comma or a space. If a project specification contains commas or spaces, it has to be specified in double quotes.

For example:

```
DM_NO_BUILD_TARGET_PRESELECT QLARIUS:PROJ1,QLARIUS:PROJ2 QLARIUS:PROJ3
    "QLARIUS:MY NEW PROJECT", "QLARIUS:PROJ A, WITH CHANGES"
```

Index

A

active item revisions 28
Ant and Dimensions Build
  archiving buildfile 184
  build areas 185
  Dimensions project required 185
  editing imported targets 186
  editing transition scripts 187
  importing buildfiles 185
  running imported build jobs 188
Apache Ant, integrating with Dimensions 191
application rule templates
  about 91
  adding 92
  deleting 93
  examples 91
  modifying 92
  transition rule templates, adding 93
  transition rule templates, deleting 94
  viewing 92
auditing, build areas 31
automatic selection of targets, disabling 294

B

Bill of Materials (BOM) *see* build footprinting
bldcomms utility
  about 256
  parameters 256
  using, on MVS 257
BRD file 22, 224, 240
build
  configuring 17, 19
  error messages, configuring 44
  fatal errors, controlling 44
  setting up 17, 18
build administration main window 48
build areas
  about 26
  active item revisions 28
  auditing 31
  cannot be deleted if used in build configuration 107
  creating a deployment area 104
  creating a library cache area 105
  creating a work area 102
  deleting 107
  locations 26
  managed development areas 27
  modifying 106
  overview 102
  security and ownership 26
build configuration type options
  about 63
  adding 64
  deleting 66
  modifying 65
build configuration types
  about 62
  adding 64
  deleting 66
  modifying 65
  Openmake 63
  viewing 64
build configurations
  about 96
  basic operations 98
  copying 100
  creating 98
  deleting 100
  modifying 99
  properties 101
  tips 132
  viewing 98
build configurations, copying 131
build environment for MVS, setting up 60
build errors
  controlling, fatal errors 44
  messages, configuring 44
build flow
  configuring 19
  on distributed platforms 20
  on MVS platforms 21
build footprinting
  introduction 232
  configuring on MVS 237
  report format 236
  setting up 234
build item types, using 41
build job schedules
  about 148
  adding 149
  deleting 156
  frequency 152
  modifying 153
  range of recurrence 152
  scheduler service, configuring 148
  start time, specifying 151
  viewing 149
build jobs

cancelling 164
execution on mainframe 159
Build Management Server 22
build management tab, build administration 49
build monitor event errors, viewing 220
build monitoring tab, build administration 53
build notification events
about 169
adding 175
deleting 177
editing 176
viewing 174
build notification subscriptions
about 170
adding 177
deleting 179
editing 178
viewing 174
build notification templates
about 168
adding 171
copying 173
deleting 173
editing 172
viewing 171
build notifications
about 168
emails, format of 168
example 169
notification service, configuring 170
variables, in templates 168
build option groups
about 72
adding 74
build options, adding 75
build options, deleting 77
build options, modifying 76
deleting 77
examples 73
modifying 76
viewing 74
build options
about 126
creating 126
deleting 128
modifying 127
overview
build order
in Dimensions CM 12.2.1 (or later) 34
pre-Dimensions CM 12.2.1 32
specifying in build targets 113
build outputs, capturing 29
build parameters, in web.xml 284
build privileges 29
build projects, *see* Dimensions projects
Build Request Definition (BRD) file 22
Build Request Definition file 240
build roles 29
build roles, using to control builds 293
build rules, controlling with item formats 41
build scheduling tab, build administration 52
build security 259, 263
build server
load balancing 287
multiple servers, defining 287
build server applet, modifying 284
build settings 61
build targets
Build Plan page of Create New Target Wizard 114
creating 111
deleting 115
disabling automatic selection 294
modifying 115
Options page of Create New Target Wizard 114
overview
Scripts page of Create New Target Wizard 113
Target page of Create New Target Wizard 112
build templates
using, with build configurations 88
build terminology 25
build tool options
adding 70
deleting 72
modifying 71
build tools
about 66
adding 69
deleting 72
example 68
external XML file format 66
importing, from XML file 70
modifying 71
viewing 69
builds, using roles to control 293

C

capturing, build outputs 29
closed-loop builds 29
configuration symbols, in the Dimensions configuration file 226, 242
configuring, Dimensions Build 17
configuring, the build flow 19
controlling builds 293
controlling, fatal build errors 44
copying, build configurations to a project or stream 131

D

debugging
  BRD file 224
  locating temporary files 223
  preserving temporary files 223
  temporary directory hidden 223
  temporary files 223
dependencies 165
  automatic detection of, on mainframes 166
  requires preserving targets 165
deployment area 104
deployment server 22
design parts
  relating to a target 112
  support for 32
Dimensions projects
Dimensions, integrating with build and project management tools 189
DM_BLD_ERROR_INVALID_REVISIONS 44, 229
DM_BLD_GETSRC_FUNCTION_OR 44
DM_BLD_MERGE_ALL_ITEMS 43
DM_BUILD_CLEAN_TEMP_FILES 229
DM_BUILD_MSGQ_INTERVAL 228
DM_BUILD_MSGQ_SIZ 227
DM_BUILD_MSGQ_STEP 228
DM_BUILD_OPTIMIZE 227
DM_BUILD_OPTIMIZE_EARLY_TEST 227
DM_BUILDERWS_URL 228
DM_DELIVER_RETRY_TIMEOUT 229
DM_MAX_PBEM_RETRIES 227
DM_MVS_REXEC_DELETE_JCL 228
DM_MVS_SBEM_TEMPNAME 230
DM_MVS_TZ 230
dm.cfg configuration symbols, in the PBEM 226, 242
DMORDERING 229

E

error messages
  Area is in use 221
  Attempt to close invalid connection 221
  Failed to authenticate to the build agent 222
  Failed to find product-specific upload rules 222
  No build areas found for the selected configuration 222
  Template pbem_stop.cmd open failed 222
  You do not have a role to extract item 222
errors, build monitor events 220
exporting
  overview 129
  to XML file 129

F

fatal build errors, controlling 44
filter conditions
  editing 141
  removing all conditions 141
  removing specific conditions 141
filtering
  introduction 138
  build jobs, by date and time 139
  build jobs, by user name 138
  Dimensions projects 140
  filter conditions, editing 141
footprinting *see* build footprinting

G

global script 121

I

importing
  from Ant build.xml 130
  from XML file 129
  overview 129
inputs
  adding new source as input 116
  adding new target as input 116
  overview 116
introduction, to Dimensions Build 16
item formats, using to control build rules 41
item revisions, at requested deployment level 43

L

library cache area 105
load balancing, a build server 287

M

managed development areas 27
Maven SCM Dimensions CM Provider, integrating with 197
MDHJLOAD, loading multiple members 257
MDHLLNK0 utility
  about 251
  example JCL 254
modifying, build parameters 284
monitoring builds
  immediately after launch 158
  past builds 164
  running builds 163

MVS build environment, setting up 60

N

NOFAIL_MISSING 44
NOFAIL_NEWER 44
NOFAIL_NOITEM 44
NOFAIL_OLDER 44
NOFAIL_SPEC 44
notification tab, build administration 55
notifications, *see* build notifications

O

options, *see* build options

P

PBEM, *see* Primary Batch Execution Monitor
placeholder item revisions 30
preservation policies
    about 30
    placeholder item revisions 30
preserving targets 30
Primary Batch Execution Monitor
    about 240
    configuration symbols 226, 242
    dm.cfg configuration symbols 226, 242
    example JCL 242
    host name override 242
    parameters 241
    running 240

R

recursive search 110
remote job execution 31
returning control 292
rewrite
    compared to rollback 144
    example 146
rollback
    compared to rewrite 144
    example 144
running builds
    error related to build areas 136
    from Build Scheduling tab 136
    single builds 134

S

SBEM, *see* Secondary Batch Execution Monitor
scheduler service, build administration 148
scheduling builds 148
Scripts
    overview 119
scripts 31
    deleting build configuration script 125
    deleting transition script 125
    modifying build configuration script 122
    modifying transition script 123
    using a Dimensions-controlled file 121
    using a file in the build area 121
    using global script 121
    viewing build configuration script 122
    viewing transition script 122
search paths, setting up 28
Secondary Batch Execution Monitor
    about 244
    changes in Dimensions 10.1.2 250
    example JCL 246
    JCL syntax 248
    parameters 245
    starting, from USS 247
security 259, 263
security and ownership, of build areas 26
setting up, a build 18
setting up, Dimensions Build 17
sources
    adding from file 107
    adding individual files 109
    and wildcard patterns 108
    deleting 111
    modifying 110
    overview 107
    recursive search 110
    wildcards not available on MVS 108
streams, support for 42
supported platforms 16

T

targets, *see* build targets
templates 31
temporary files 223
temporary files, preserving 223
terminology, Dimensions Build 25
transition rule template build option groups
    adding 86
    deleting 87
transition rule template build options
    adding 84
    deleting 86
    modifying 85
transition rule template input masks
    adding 81
    deleting 82

modifying 82
transition rule template output masks
adding 83
deleting 83
modifying 83
transition rule templates
about 78
adding 79
deleting 81
examples 78
modifying 80
script types, selecting 79
viewing 79
troubleshooting 219
build monitor event errors, viewing 220
error messages 221

V

versions
allowed operations 144
delete not allowed 146
possible operations 128

W

wait processing 292
web.xml, modifying build variables 284, 288
wildcards, expanding 113
wildcards, in build configurations
support for 32
wildcards, in Mainframe build configurations
use cases 37
Windows Server 2003, running builds on 224
work area 102

Z

z/OS mainframe nodes, creating definitions for 60