SERENA®

DIMENSIONS® CM 14.2.0.2

Developer's Reference

Serena Proprietary and Confidential Information

Copyright © 1988–2016 Serena Software, Inc. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by such license, no part of this publication may be reproduced, photocopied, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Serena. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

This document contains proprietary and confidential information, and no reproduction or dissemination of any information contained herein is allowed without the express permission of Serena Software.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Serena. Serena assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Third party programs included with the Dimensions product are subject to a restricted use license and can only be used in conjunction with Dimensions.

Trademarks

Serena, TeamTrack, StarTool, PVCS, Comparex, Dimensions, Prototype Composer, Mariner, and ChangeMan are registered trademarks of Serena Software, Inc. The Serena logo and Version Manager are trademarks of Serena Software, Inc. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

U.S. Government Rights

Any Software product acquired by Licensee under this Agreement for or on behalf of the U.S. Government, its agencies and instrumentalities is "commercial software" as defined by the FAR. Use, duplication, and disclosure by the U.S. Government is subject to the restrictions set forth in the license under which the Software was acquired. The manufacturer is Serena Software, Inc., 1850 Gateway Drive, 4th Floor, San Mateo, California 94404.

Publication date: March 2016

Table of Contents

Welcome to Serena Dimensions CM 29
Before you Begin 29
Contacting Serena Technical Support 29
Videos 29
License and Copyright Information for Third-Party Software 30

Part 1 C/C++ API DTK 31

Chapter 1 **What is the Dimensions CM Toolkit Interface? 33**
Overview 34
Positioning Your Solutions within the DTK 34
Client Architecture 34
Event Architecture 35
Interactions between the Two Architectures 35
Scope of the DTK Architecture 35

Chapter 2 **Writing Dimensions CM DTK Applications 37**
Introduction 38
DTK Return Codes 39
DTK Data Structures 40
PcmsCallbackStruct 40
PcmsChdAttachmentStruct 40
PcmsEventStruct 41
PcmsLcStruct 41
PcmsObjAttrDefStruct 42
PcmsObjAttrStruct 43
PcmsObjStruct 44
PcmsPendingUserStruct 44
PcmsPendStruct 45
PcmsRelStruct 45
PcmsRelTypeStruct 46
PcmsRoleStruct 46
PcmsTypeStruct 46
PcmsUserRoleStruct 47
DTK System Attribute Definitions 47
DTK Constant Definitions 52
Memory Allocation within the DTK 53
Usage of the Functions 54

Chapter 3 **DTK API Functions for C/C++ 57**
Introduction 59
Memory Allocation by DTK Functions 59
Access Security in DTK Functions 59
PcmsAttrDefInit - Get Attribute Definition 61
Purpose 61
Prototype 61
Parameters 61
Return Codes 61
Related Functions 61
PcmsAttrGetLov - Get Attribute's List of Values 62
Purpose 62
Prototype 62
Parameters 62
Return Codes 63
Sample 63
Related Functions 64
PcmsAttrValidate - Validate an Attribute Value 65
Purpose 65
Prototype 65
Parameters 65
Return Codes 65
Related Functions 65
PcmsCheckMessages - Check Results of Dimensions CM Command 66
Purpose 66
Prototype 66
Parameters 66
Return Codes 66
Comments 66
Related Functions 67
PcmsCntrlPlanGet - Get Dimensions CM Process Model Information 68
Purpose 68
Prototype 68
Parameters 68
Return Codes 70
PcmsConnect - Connect to Dimensions CM Database 71
Purpose 71
Prototype 71
Parameters 71
Return Codes 71
Comments 71
Sample 72
Related Function 72
PcmsDisconnect - Disconnect from a Dimensions CM Database 73
Purpose 73
Prototype 73
Parameters 73
Return Codes 73

Sample . . . 73
Comments . . . 73
Related Function . . . 73
PcmsEvntCalloc – Allocate Zero Initialized Memory . . . 74
Purpose . . . 74
Prototype . . . 74
Parameters . . . 74
PcmsEvntFree – Free Memory . . . 75
Purpose . . . 75
Prototype . . . 75
Parameters . . . 75
PcmsEvntMalloc – Allocate Memory . . . 76
Purpose . . . 76
Prototype . . . 76
Parameters . . . 76
PcmsEvntRealloc – Re-Allocate Memory . . . 77
Purpose . . . 77
Prototype . . . 77
Parameters . . . 77
PcmsExecCommand - Execute Dimensions CM Command Synchronously . . 78
Purpose . . . 78
Prototype . . . 78
Parameters . . . 78
Return Codes . . . 78
Sample . . . 78
Comments . . . 79
Related Function . . . 79
PcmsFullQuery - Find Dimensions CM Objects, Returning Complete Objects 80
Purpose . . . 80
Prototype . . . 80
Parameters . . . 80
Return Codes . . . 81
Sample . . . 81
Comments . . . 82
Related Functions . . . 83
PcmsGetAttachments - Obtain Request Attachment Structures . . . 84
Purpose . . . 84
Prototype . . . 84
Parameters . . . 84
Return Codes . . . 84
PcmsGetAttrDefNum - Get Attribute Definition Number . . . 85
Purpose . . . 85
Prototype . . . 85
Parameters . . . 85
Return Codes . . . 85
PcmsGetAttrFile - Get Request Descriptions . . . 86
Purpose . . . 86
Prototype . . . 86

Parameters . . . 86
Return Codes . . . 87
PcmsGetAttrs - Get Dimensions CM Object Attributes . . . 88
Purpose . . . 88
Prototype . . . 88
Parameters . . . 88
Return Codes . . . 88
Comments . . . 88
Related Functions . . . 88
PcmsGetBaseDbOptions - Retrieve Base Database Options . . . 89
Purpose . . . 89
Prototype . . . 89
Parameters . . . 89
Return Codes . . . 89
Comments . . . 89
Related Functions . . . 89
PcmsGetCandidates - Retrieve Candidates for Delegation . . . 90
Purpose . . . 90
Prototype . . . 90
Parameters . . . 90
Return Codes . . . 90
PcmsGetCommandLine – Get the Dimensions CM Command . . . 91
Purpose . . . 91
Prototype . . . 91
Parameters . . . 91
PcmsGetConnectDesc - Get Input File Descriptor . . . 92
Purpose . . . 92
Prototype . . . 92
Parameters . . . 92
Return Codes . . . 92
Comments . . . 92
Related Functions . . . 92
PcmsGetDimensionsVersions - Obtain List of Loaded DLLs and Dates and Times . . . 93
Purpose . . . 93
Prototype . . . 93
Parameters . . . 93
Return Codes . . . 93
Comments . . . 93
Related Functions . . . 93
PcmsGetPendingUsers - Obtain Inbox User Structures . . . 94
Purpose . . . 94
Prototype . . . 94
Parameters . . . 94
Return Codes . . . 95
Related Functions . . . 95
PcmsGetRSAttrs - Retrieve Attribute Numbers in a Role Section . . . 96
Purpose . . . 96

Prototype 96
Parameters. 96
Return Codes 97
Comments 97
Related Functions 97
PcmsGetRSNames - Obtain Role Section Names for a Product 98
Purpose 98
Prototype 98
Parameters. 98
Return Codes 98
Related Functions 99
PcmsGetStringSymbolValue - Obtain Values Associated with a Symbol. . . . 100
Purpose 100
Prototype 100
Parameters. 100
Return Codes 100
Comments 100
Related Functions 100
PcmsGetSymbolInfo - Look Up Symbol in Structured Information Return Table 101
Purpose 101
Prototype 101
Parameters. 101
Return Codes 101
Comments 102
Related Functions 102
PcmsGetUserRelTypes - Obtain User Relationship Subtypes 103
Purpose 103
Prototype 103
Parameters. 103
Return Codes 103
Comments 104
PcmsGetUserRoles - Obtain User Role Structures. 105
Purpose 105
Prototype 105
Parameters. 105
Return Codes 106
Comments 106
Related Functions 106
PcmsGetWsetObj - Get User's Current Project 107
Purpose 107
Prototype 107
Parameters. 107
Return Codes 107
Sample 107
PcmsInitSpec - Get Dimensions CM Object Details by Specification 108
Purpose 108
Prototype 108

Parameters 108
Return Codes 108
Comments 108
Related Functions 108
PcmsInitUid - Get Dimensions CM Object Details by Uid 109
Purpose 109
Prototype 109
Parameters 109
Return Codes 109
Comments 109
Related Functions 109
PcmsLovFree - Free a List of Values 110
Purpose 110
Prototype 110
Parameters 110
Return Codes 110
Related Functions 110
PcmsObjFree - Free Dimensions CM Object Structures 111
Purpose 111
Prototype 111
Parameters 111
Return Codes 111
PcmsObjGetBackRels - Get Dimensions CM Object Reverse Relationships . . 112
Purpose 112
Prototype 112
Parameters 112
Return Codes 113
Comments 113
Related Functions 113
PcmsObjGetRels - Get Dimensions CM Object Relationships 114
Purpose 114
Prototype 114
Parameters 114
Return Codes 115
Comments 115
Related Functions 115
PcmsObjInSecondary - Is Request Object in Secondary Catalog 116
Purpose 116
Prototype 116
Parameters 116
Return Codes 116
PcmsPendGet - Retrieve Dimensions CM Object Inboxes for a User 117
Purpose 117
Prototype 117
Parameters 117
Return Codes 118
Comments 118
PcmsPendWhoGet - Retrieve Users for Object 119

Purpose . . . 119
Prototype . . . 119
Parameters . . . 119
Return Codes . . . 120
Related Functions . . . 120
Comments . . . 120
PcmsPopulate - Populate an Object's Attributes Values . . . 121
Purpose . . . 121
Prototype . . . 121
Parameters . . . 121
Return Codes . . . 121
PcmsQuery - Find Dimensions CM Objects, Returning Uids . . . 122
Purpose . . . 122
Prototype . . . 122
Parameters . . . 122
Return Codes . . . 122
Sample . . . 123
Comments . . . 124
Related Functions . . . 125
PcmsSendCommand - Execute Dimensions CM Command Asynchronously . 126
Purpose . . . 126
Prototype . . . 126
Parameters . . . 126
Return Codes . . . 126
Comments . . . 126
Related Functions . . . 127
PcmsSetAttrs - Set Dimensions CM Object Attributes . . . 128
Purpose . . . 128
Prototype . . . 128
Parameters . . . 128
Return Codes . . . 128
Comments . . . 128
Related Functions . . . 129
PcmsSetCallback - Set Dimensions CM API Server Callback . . . 130
Purpose . . . 130
Prototype . . . 130
Parameters . . . 130
Return Codes . . . 131
Comments . . . 131
Related Functions . . . 131
PcmsSetDbErrorCallback - Set Server Error Callback . . . 132
Purpose . . . 132
Prototype . . . 132
Parameters . . . 132
Return Codes . . . 132
Comments . . . 133
Related Functions . . . 133
PcmsSetDirectory - Change Dimensions CM Default Directory . . . 134

Purpose . . . 134
Prototype . . . 134
Parameters . . . 134
Return Codes . . . 134
Comments . . . 134
Related Functions . . . 134
PcmsSetIdleChecker - Install Idle Checker . . . 135
Purpose . . . 135
Prototype . . . 135
Parameters . . . 135
Return Codes . . . 135
Comments . . . 135
PcmsSetWsetObj - Set User's Current Project . . . 136
Purpose . . . 136
Prototype . . . 136
Parameters . . . 136
Return Codes . . . 136
PcmsWriteAttachments - Write Specified Request Attachments To Disk . . . 137
Purpose . . . 137
Prototype . . . 137
Parameters . . . 137
Return Codes . . . 137
Attribute Macros . . . 138
Initialize PcmsObjStruct attrs . . . 138
Add attrDef Structures . . . 138
Single-Value Attributes (SVA) . . . 138
Multi-Value Attributes (MVA) . . . 139

Chapter 4 **DTK API Functions for Windows Client Installations . . . 141**

Introduction . . . 142
Building Client Applications . . . 142
Sample Code Fragment . . . 143
PcmsClntApiConnect - Connect to a Dimensions CM Database . . . 144
Purpose . . . 144
Prototype . . . 144
Parameters . . . 144
Return Codes . . . 144
Related Functions . . . 144
PcmsClntApiDisconnect - Disconnect from a Dimensions CM Database . . . 145
Purpose . . . 145
Prototype . . . 145
Parameters . . . 145
Return Codes . . . 145
Related Functions . . . 145
PcmsClntApiExecCommand - Execute a Dimensions CM Command . . . 146
Purpose . . . 146
Prototype . . . 146
Parameters . . . 146

Return Codes 146
PcmsClntApiFree – Free Memory 147
Purpose 147
Prototype 147
Parameters 147
PcmsClntApiGetLastError - Get the Last Dimensions CM Message 148
Purpose 148
Prototype 148
Parameters 148
Return Codes 148
Related Function 148
PcmsClntApiGetLastErrorEx - Get the Last Dimensions CM Message (Dynamic Buffer) 149
Purpose 149
Prototype 149
Parameters 149
Return Codes 149
Related Function 149
PcmsClntApiModeBinary - Set File Transfer Mode to Binary 150
Purpose 150
Prototype 150
Parameters 150
Return Codes 150
Comments 150
PcmsClntApiModeText - Set File Transfer Mode to ASCII 151
Purpose 151
Prototype 151
Parameters 151
Return Codes 151
Comments 151
PcmsClntApiSilentConnect - Connect Silently to a Dimensions CM Database 152
Purpose 152
Prototype 152
Parameters 152
Return Codes 152
Related Functions 153
Additional Supported DTK Functions 154

Part 2 Events Callout 155

Chapter 5 **Dimensions CM Events Callout Interface 157**

Description 158
Shared Libraries 158
MBCS Versus UTF-8 Support 158
Large File Support 159
Public Function Call 159
Event Callout Interface 160
Validate Events 160

Pre-Events . . . 160
Post-Events . . . 161
Event Types . . . 161
Determining the Event You Want . . . 163
First and Second Event Calls . . . 164
Event Call Summary . . . 165
Writing a DTK Callout Event . . . 165
Is an Event the Solution for You? . . . 165
Designing Your Event . . . 166
Writing Your Event . . . 167
DTK Event Internals . . . 168
Changing System-Attributes on Validate Events . . . 169
Changing User-Attributes on Validate Events . . . 170
Recommendations on How to Change Attribute Values . . . 170
Calling DTK Functions Within Events . . . 171
Specialist DTK Event Functions . . . 171
Unsupported DTK Function Calls from Within an Event . . . 171
Using the ptrEventInfo in Events . . . 172
Recompiling With New Versions of Dimensions CM . . . 173
Event Examples . . . 173
Events - A Final Word and a Warning . . . 174

Part 3 Java API . . . 175

Chapter 6 **dmclient . . . 177**
Introduction . . . 178
Using dmclient . . . 179
Entry Point to dmclient . . . 180
Getting Started . . . 180
Time Format Strings in Java-API . . . 180
Javadoc . . . 181
Examples . . . 181

Part 4 The Dimensions CM Templating Language . . . 183

Chapter 7 **The Templating Language and Processor . . . 185**
Introduction . . . 186
About Templates . . . 186
Default Template Characters . . . 187
Directives . . . 188
)ADJUST . . . 188
)ATTR . . . 188
)CALL . . . 189
)CALLBACK . . . 189
)CM . . . 189
)CERTLOGON . . . 189
)COMMAND . . . 190

)COPYSYM 190
)DELETE 190
)DUMP 191
)ELSE 191
)ENDIF 192
)ENDEXPAND 193
)ENDR 193
)ENDSCRIPT 193
)EXPAND 193
)IF 194
)IFDEF 194
)IFGROUPDEF 194
)IFGROUPNDEF 195
)IFNDEF 195
)IM 195
)LOAD 196
)LOGON 196
)REP 196
)SAVE 198
)SCRIPT 198
)SET 198
)SETL 198
)SET_PATH 199
)SET_PATH_MVS 199
)SET_PATH_NT 200
)SET_PATH_UNIX 200
)SETU 200
)SLICE 201
)VECTOR 201
Special Directives 201
General Predefined Symbols 202
DMCERTIFICATE 202
DMCURRENTDIRECTORY 202
DMCYGWINCWD 202
DMDAY 202
DMFOLDERSEPARATOR 203
DMHOUR 203
DMMICROSEC 203
DMMINUTE 203
DMMONTH 203
DMOSTDERR 203
DMOSTDOUT 204
DMOSTYPE 204
DMOXTMPLT 204
DMPASSWORD 204
DMPATHSEPARATOR 204
DMSECOND 204
DMUNIQUE 205

DMUSER 205
DMUSERU 205
DMYEAR 205
Control Symbols 206
DMSAVESTDERR 206
DMSAVESTDOUT 206
DMCOMBINEOUTERR 207
Remote Job Execution Predefined Symbols 207
DMCERTIFICATE 207
DMJOBID 207
Dimensions Build Standard Symbols 208
DMPATH 208
DMINPUT 210
DMTARGET 210
DMSTEP 210
DMRUNMODE 210
DMUSER and DMPASS 211
DMSERVER 211
DMPBEMNODE and DMPBEMPORT 211
DMTOKEN 211
DMJOBID 211
DMXFERSOURCE 211
DMBUILDJOB 211
Dimensions Build User-Defined Optional Symbols 212
DMEXECENV 212
DMEXPAND 213
DMTASKNAME 213
DMTIMEOUT 213
DMSTEPMODE 214
DMMAXRC 214
DMNOCACHE 214
DMREBUILDALL 214
DMMUSTRUN 214
DMFINAL 214
DMTTLOG 215
DMALTSERVER 215
DM_SP_START_STAGE 215
DM_BUILDER_PROGRESS_REPORTING 215
Dimensions for z/OS Predefined Symbols 216
User ID and Account Predefined Symbols 216
Job Sequence Number Predefined Symbols 216
Template Expansion and Scripts 217
Complex Symbol References 220
Template Inline Functions 220
Overview 220
MDHDSN 220
MDHEXTRACT 223
substring 224

pos . . . 225
Testing Templates . . . 225
MVS Template Testing Program . . . 227
Adding Template Variables to the Dimensions Configuration File . . . 227
Extending the Template Processor with User Written Functions . . . 229
Template Processing Call-Out Functionality . . . 229
Template Processing: Passing Data Between Steps . . . 234
The Template Processor Interface . . . 242

Chapter 8 **Structured Information Return . . . 245**
What is SIR? . . . 246
Key features . . . 246
SIR Commands . . . 247
SSPM . . . 247
SAVE . . . 248
Examples . . . 248

Chapter 9 **Introduction to Build Templates . . . 249**
Understanding Build . . . 250
What is a Build Template? . . . 250
Where are Build Templates Located? . . . 251
Using Build Templates with Build Configurations . . . 252
Executing a Single Build Step . . . 252
Simple Build Template Example . . . 253
Build Template Types . . . 254
High Level Templates . . . 254
Compilation Templates . . . 254
Collection Templates . . . 254
Coordination Templates . . . 254
Build Template Options . . . 254
Expanding Wildcards . . . 256
Using a Build Type with Wildcards in Build Templates . . . 256
Using Build Configuration Inputs and Outputs in Templates . . . 257
Handling Outputs and Error Logs . . . 258
Generating a Bill of Materials . . . 258
Search Paths . . . 259
Initial Execution Path . . . 259
Footprinting . . . 259
Communicating between Templates . . . 259
Callouts . . . 260
Managing Build and Deployment Job Logs . . . 260

Chapter 10 **Build Templates for Distributed Platforms . . . 261**
Introduction . . . 262
High Level Templates . . . 263
Compilation Templates . . . 263
Collection Templates . . . 264
Using Mixed Inputs in Build Configurations . . . 264

Using Search Paths 265
Asynchronous Templates 266
Configuring Messaging 266
Templating Techniques 267
Using Special Characters in Windows 267
Handling Array Variables 268
Picking a Specific Array Value 269
Decomposing a Path or File Specification 269
Combining STDERR and STDOUT Streams 270
Submitting Formatted Text after Preprocessing 270
Using Make and Creating a Bill Of Materials 271

Chapter 11 **Build Templates for MVS Platforms 275**
Introduction 276
Secondary Build Execution Monitor 276
MVS Templates 277
Basic MVS Example Template 277
Terminating a Build Step 278
Using the SBEM 279
Writing Plain JCL Templates 279
Wrapping Generated JCL into the SBEM 280
Using Outputs in a Build Template 281
Using Outputs from Build Steps 281
Using Listings and Reports 281
Using Mixed Inputs 281
Build Step Warning Messages 282
Computing the Final Return Code 282
Creating a Bill of Materials from the SBEM 283
Source Types 284
Mapping between MVS and Dimensions CM 284
Using REXX in a Build Template 286
Controlling Simultaneous Conflicting ENQs in Data Sets 290

Chapter 12 **Openmake Templates 293**
Introduction 294
Variables 295

Chapter 13 **Remote Job Execution Templates 299**
Introduction 300
Templates 300
Parameters 300
Simple Synchronous Templates 300
Feedback 301
Asynchronous Remote Jobs 302
Remote Job Template Example 303

Chapter 14 **Deployment Area Scripts 309**
Introduction 310
Changes Introduced in Dimensions CM 12.2.1 311
Variables 312
Testing Variables 313
Differences in Relation to Build Templates 314
Debugging 314
Other Features 314
Specifying User Attributes in Deployment Area Scripts 315
Introduction 315
Example 317

Part 5 Web Services Reference 319

Chapter 15 **Getting Started 321**
Introduction 322
About the Dimensions CM Web Services API 322
Dimensions CM Web Services Architecture 322
Introduction 322
Dimensions CM WSDL 323
Dimensions CM Web Services Logging Mechanism 324
Dimensions CM Web Service Error Handling 326
Before You Begin 328
System Requirements 328
Licensing 328
Security 328
Setting Up a Development Environment 328
Installing the Dimensions CM Web Services API 330
Setting the Idle Connection Timeout 330
What is the Idle Connection Timeout? 330
Changing the Idle Connection Time-Out 330

Chapter 16 **Tips for Using Dimensions CM Web Services 331**
Authentication Methods 332
Supported Character Encoding 332
Supported Datetime Formats 332
General Datetime Format 332
Dimensions CM Attributes Datetime Format 333

Chapter 17 **Application Lifecycle Framework Events 335**
Introduction to ALF Events 336
Dimensions CM ALF Events Architecture 336
Introduction 336
Dimensions CM ALF Events Declaration 336
Configuring ALF Events 339
Generation of ALF events 340
ALF Events Available 340

Baseline ALF Events . . . 341
Item ALF Events . . . 341
Project ALF Events . . . 342
Request ALF Events . . . 342
Miscellaneous ALF Events . . . 343
Logging of ALF Events . . . 343
Supported Datetime Formats . . . 343
Error Message Logs . . . 343

Chapter 18 **Web Services Reference . . . 345**

actionRequest . . . 347
Description . . . 347
Arguments . . . 347
Response . . . 347
Usage . . . 347
Example SOAP Request . . . 348
Example SOAP Response . . . 348
buildBaseline . . . 349
Description . . . 349
Arguments . . . 349
Response . . . 349
Usage . . . 350
Example SOAP Request . . . 350
Example SOAP Response . . . 351
buildProject . . . 352
Description . . . 352
Arguments . . . 352
Response . . . 353
Usage . . . 353
Example SOAP Request . . . 354
Example SOAP Response . . . 355
checkInItem . . . 356
Description . . . 356
Arguments . . . 356
Response . . . 357
Usage . . . 357
Example SOAP Request . . . 358
Example SOAP Response . . . 359
checkOutItem . . . 360
Description . . . 360
Arguments . . . 360
Response . . . 362
Usage . . . 362
Example SOAP Request . . . 362
Example SOAP Response . . . 363
createDeploymentArea . . . 364
Description . . . 364
Arguments . . . 364

Response . . . 365
Usage . . . 365
Example SOAP Response . . . 366
createDesignPart . . . 367
Description . . . 367
Arguments . . . 367
Response . . . 367
Usage . . . 367
Example SOAP Request . . . 368
Example SOAP Response . . . 368
createDesignPartBaseline . . . 369
Description . . . 369
Arguments . . . 369
Response . . . 371
Usage . . . 371
Example SOAP Request . . . 371
Example SOAP Response . . . 372
createProject . . . 373
Description . . . 373
Arguments . . . 373
Response . . . 374
Usage . . . 374
Example SOAP Request . . . 375
Example SOAP Response . . . 376
createProjectBaseline . . . 377
Description . . . 377
Arguments . . . 377
Response . . . 378
Usage . . . 378
Example SOAP Request . . . 379
Example SOAP Response . . . 379
createRequest . . . 380
Description . . . 380
Arguments . . . 380
Response . . . 381
Usage . . . 381
Example SOAP Request . . . 382
Example SOAP Response . . . 382
createRevisedBaseline . . . 383
Description . . . 383
Arguments . . . 383
Response . . . 384
Usage . . . 384
Example SOAP Request . . . 385
Example SOAP Response . . . 385
createScheduleJob . . . 386
Description . . . 386
Arguments . . . 386

Response 387
Usage 387
Example SOAP Request 387
Example SOAP Response 388
createStream 389
Description 389
Arguments 389
Response 390
Usage 390
Example SOAP Response 391
createWorkArea 392
Description 392
Arguments 392
Response 392
Usage 393
Example SOAP Response 393
delegateRequest 394
Description 394
Arguments 394
Response 394
Usage 394
Example SOAP Request 395
Example SOAP Response 395
delegateRequestForReplication 396
Description 396
Arguments 396
Response 396
Usage 396
Example SOAP Request 397
Example SOAP Response 397
demoteBaselines 398
Description 398
Arguments 398
Response 398
Usage 398
Example SOAP Request 399
Example SOAP Response 400
demoteItems. 401
Description 401
Arguments 401
Response 401
Usage 402
Example SOAP Request 402
Example SOAP Response 403
demoteRequests 404
Description 404
Arguments 404
Response 404

Usage . . . 405
Example SOAP Request . . . 405
Example SOAP Response . . . 406
deployBaseline . . . 407
Description . . . 407
Arguments . . . 407
Response . . . 407
Usage . . . 407
Example SOAP Request . . . 408
Example SOAP Response . . . 408
deployRequest . . . 409
Description . . . 409
Arguments . . . 409
Response . . . 409
Usage . . . 409
Example SOAP Request . . . 410
Example SOAP Response . . . 410
getItems . . . 411
Description . . . 411
Arguments . . . 411
Response . . . 412
Usage . . . 413
Example SOAP Request . . . 413
Example SOAP Response . . . 414
getRequestDescription . . . 415
Description . . . 415
Arguments . . . 415
Response . . . 415
Usage . . . 415
Example SOAP Request . . . 415
Example SOAP Response . . . 416
getRequests . . . 417
Description . . . 417
Arguments . . . 417
Response . . . 417
Usage . . . 417
Example SOAP Request . . . 418
Example SOAP Response . . . 418
getVersion . . . 422
Description . . . 422
Arguments . . . 422
Response . . . 422
Usage . . . 422
Example SOAP Request . . . 423
Example SOAP Response . . . 423
listDeploymentAreas . . . 424
Description . . . 424
Arguments . . . 424

Response 424
Usage 424
Example SOAP Response 425
listProjectItems 426
Description 426
Arguments 426
Response 426
Usage 426
Example SOAP Request 427
Example SOAP Response 427
listProjectRequests 429
Description 429
Arguments 429
Response 429
Usage 429
Example SOAP Request 430
Example SOAP Response 430
listProjects 431
Description 431
Arguments 431
Response 431
Usage 431
Example SOAP Request 432
Example SOAP Response 432
listScheduleJobs 434
Description 434
Arguments 434
Response 435
Usage 435
Example SOAP Request 436
Example SOAP Response 436
listWorkAreas 439
Description 439
Arguments 439
Response 439
Usage 439
Example SOAP Response 440
moveItemToPart 441
Description 441
Arguments 441
Response 441
Usage 441
Example SOAP Request 442
Example SOAP Response 442
promoteBaselines 443
Description 443
Arguments 443
Response 444

Usage 444
Example SOAP Request 445
Example SOAP Response 446
promoteItems 447
Description 447
Arguments 447
Response 448
Usage 448
Example SOAP Request 449
Example SOAP Response 449
promoteRequests 450
Description 450
Arguments 450
Response 451
Usage 451
Example SOAP Request 452
Example SOAP Response 452
relateItemsToParts 453
Description 453
Arguments 453
Response 453
Usage 453
Example SOAP Request 454
Example SOAP Response 455
relateRequestToRequests 456
Description 456
Arguments 456
Response 456
Usage 457
Example SOAP Request 457
Example SOAP Response 458
removeDeploymentArea 459
Description 459
Arguments 459
Response 459
Usage 459
Example SOAP Response 460
removeWorkArea 461
Description 461
Arguments 461
Response 461
Usage 461
Example SOAP Response 462
runCommand 463
Description 463
Arguments 463
Response 463
Usage 463

Example SOAP Request . . . 464
Example SOAP Response . . . 464
submitDeployBaselines . . . 465
Description . . . 465
Arguments . . . 465
Response . . . 465
Usage . . . 466
Example SOAP Request . . . 466
Example SOAP Response . . . 467
submitDeployItems . . . 468
Description . . . 468
Arguments . . . 468
Response . . . 468
Usage . . . 469
Example SOAP Request . . . 470
Example SOAP Response . . . 470
submitDeployRequests . . . 471
Description . . . 471
Arguments . . . 471
Response . . . 471
Usage . . . 472
Example SOAP Request . . . 473
Example SOAP Response . . . 473
submitRollbackBaselines . . . 474
Description . . . 474
Arguments . . . 474
Response . . . 474
Usage . . . 474
Example SOAP Request . . . 475
Example SOAP Response . . . 476
submitRollbackItems . . . 477
Description . . . 477
Arguments . . . 477
Response . . . 477
Usage . . . 477
Example SOAP Request . . . 478
Example SOAP Response . . . 479
submitRollbackRequests . . . 480
Description . . . 480
Arguments . . . 480
Response . . . 480
Usage . . . 480
Example SOAP Request . . . 481
Example SOAP Response . . . 482
unrelateItemsFromParts . . . 483
Description . . . 483
Arguments . . . 483
Response . . . 483

Usage . . . 483
Example SOAP Request . . . 484
Example SOAP Response . . . 485
unrelateRequestFromRequests . . . 486
Description . . . 486
Arguments . . . 486
Response . . . 486
Usage . . . 486
Example SOAP Request . . . 487
Example SOAP Response . . . 488
updateDeploymentArea . . . 489
Description . . . 489
Arguments . . . 489
Response . . . 490
Usage . . . 490
Example SOAP Response . . . 491
updateRequest . . . 492
Description . . . 492
Arguments . . . 492
Response . . . 493
Usage . . . 493
Example SOAP Request . . . 493
Example SOAP Response . . . 494
updateRequestDescription . . . 495
Description . . . 495
Arguments . . . 495
Response . . . 495
Usage . . . 495
Example SOAP Request . . . 495
Example SOAP Response . . . 496
updateWorkArea . . . 497
Description . . . 497
Arguments . . . 497
Response . . . 497
Usage . . . 498
Example SOAP Response . . . 498

Chapter 19 **Application Lifecycle Framework Events Reference . . . 499**

Introduction to ALF Events . . . 500
ALF Event Structure . . . 500
Baseline ALF Events . . . 502
Event Extension for Baseline . . . 502
Build-Submitted and Build-Completed Baseline ALF Events . . . 503
Deploy Baseline ALF Event . . . 506
Deployment ALF Events . . . 508
Event Extension for Deployment . . . 508
Promote/Demote Item Events . . . 509
Deploy Baseline/Item/Request to Area ALF Events . . . 510

Rollback Baseline/Item/Request ALF Events . . . 513
Rollback Area Version ALF Event . . . 517
Item ALF Events . . . 519
Event Extension for Item . . . 519
Check-In Item ALF Event . . . 520
Check-Out Item ALF Event . . . 522
Create Item ALF Event . . . 524
Undo-Check-Out Item ALF Event . . . 526
Project ALF Events . . . 528
Event Extension for Project . . . 528
Build-Submitted and Build-Completed Project ALF Events . . . 529
Project deliver Events . . . 532
Project create Events . . . 535
Request ALF Events . . . 536
Event Extension for Request . . . 536
Action Request ALF Event . . . 536
Create Request ALF Event . . . 538
Deploy Request ALF Event . . . 541
Update Request ALF Event . . . 542
Schedule Job ALF Events . . . 545
Event Extension for Schedule Job . . . 545
Run Schedule Job ALF Event . . . 546

Chapter 20 **RESTful Web Services . . . 549**
Introduction . . . 550
Path Parameters . . . 551
Query Parameters . . . 551
Authentication Methods . . . 552
Example JSON Responses . . . 553
Calling a RESTful Web Service . . . 553
Listing Groups . . . 553
Listing Projects . . . 554
Listing Baselines . . . 555
WADL Definition File . . . 555
Traces and Logs . . . 556
RESTful Logs . . . 556
Connection Pool . . . 556
Configuring Default Server Parameters . . . 557
Reference . . . 558
getDeploymentAreas . . . 558
getDeploymentAreaDetails . . . 558
getGroups . . . 559
getProducts . . . 559
getProductDetails . . . 560
getBaselines . . . 560
getBaselineDetails . . . 561
getBaselineTemplates . . . 561
getBaselineTemplateDetails . . . 562

getDesignParts . . . 562
getDesignPartDetails . . . 563
getChildDesignParts . . . 563
getProjects . . . 564
getProjectDetails . . . 564
getProjectBaselines . . . 565
getProjectBuildConfigurations . . . 565
getProjectFileAreas . . . 566
getProjectStages . . . 566
getProductRoleAssignments . . . 567
getRoles . . . 567
getUsers . . . 568
getVersion . . . 568

Part 6 **Appendixes . . . 569**

Appendix A **Known DTK Event Issues . . . 571**
Missing Events . . . 572
Running External Executables . . . 572
Running dmcli from Event Triggers . . . 572
Windows Example . . . 573
UNIX Example . . . 573

Appendix B **Using the Bill of Materials API . . . 575**
Introduction . . . 576
Errors . . . 576
Persistence . . . 576
Connection . . . 576
Filenames . . . 576
Bill Of Materials API Reference . . . 577

Index . . . 581

Welcome to Serena Dimensions CM

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

Before you Begin

The Dimensions CM readme contains the following information:

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

The readme is available online at:

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

Contacting Serena Technical Support

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

http://www.serena.com/support

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

You can use the Serena Support web page to:

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

Videos

Videos of Dimensions CM features can be viewed online at:

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

License and Copyright Information for Third-Party Software

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

http://www.serena.com/support

Part 1

C/C++ API DTK

Part 1: C/C++ API DTK contains the following chapters:

What is the Dimensions CM Toolkit Interface?	33
Writing Dimensions CM DTK Applications	37
DTK API Functions for C/C++	57
DTK API Functions for Windows Client Installations	141

Chapter 1

What is the Dimensions CM Toolkit Interface?

Overview	34
Positioning Your Solutions within the DTK	34
Scope of the DTK Architecture	35

Overview

The *Serena*® Dimensions® CM Developer's Toolkit Interface (DTK) is a powerful C and C++ Applications Programming Interface (API) that allows you to access data held within a Dimensions CM repository.

The DTK provides a way in which you can:

- Design and implement your own applications that can interact with Dimensions CM.
- Implement your own specific customizations using a rich event callout interface.

This chapter takes you through the architecture of the DTK and how you can use it to expand on the functionality offered by Dimensions CM.

Positioning Your Solutions within the DTK

The DTK provides two comprehensive architectures that allows you to integrate your solutions with Dimensions CM in the following ways:

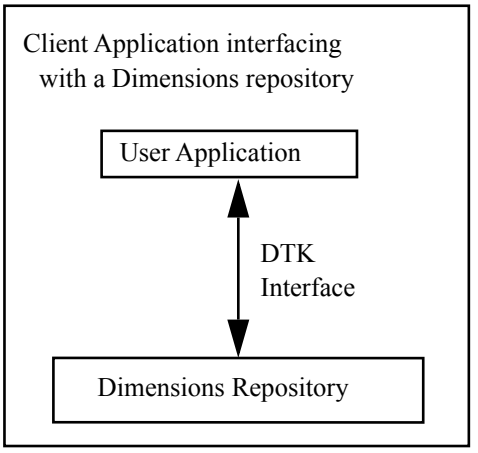
1. As a separate client application that uses the DTK to access and manipulate objects held within a Dimensions CM repository
2. As a rich event callout interface that allows you to perform your own customized operations when certain Dimensions CM commands are run.

When you are looking at your requirements keep in mind where within the DTK architecture you wish to position your solution. If, for example, you have a requirement where you need to assess the impact to one of your projects of implementing a number of application changes, then you would use the format for Client Architecture as stated in the following subsection.

Client Architecture

Using the *Client Architecture* you would design an application that would:

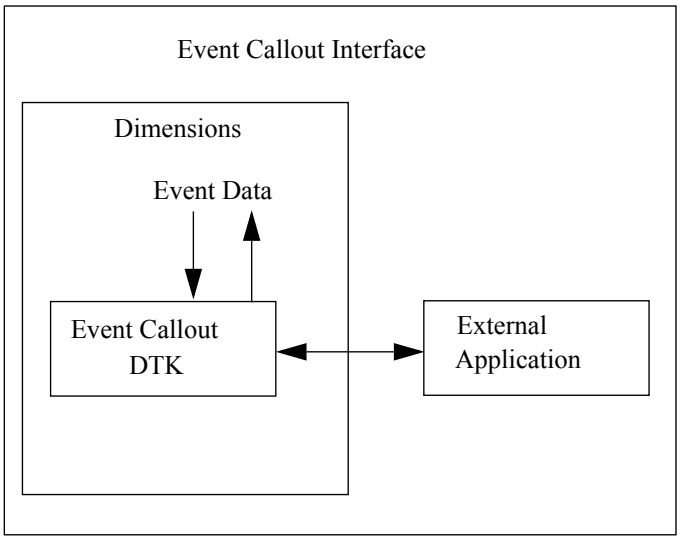
- Connect to your Dimensions CM repository.
- Query or manipulate the data from that repository.
- Disconnect from the repository and take some action based on that data.



This scheme enables you to use the DTK as an input/output interface into the Dimensions CM repository. If, however, you wanted a specific operation or action to be performed when certain Dimensions CM commands are run, then you would use the *Event Architecture* described below.

Event Architecture

Using the *Event Architecture* you can design a set of customizations that are applied explicitly when a user issues a certain Dimensions CM command. A public interface is provided that allows you to pass information back and forth between the event and the Dimensions CM server. The result of this is that you can manipulate the outcome of the command in certain ways.



For more details on how events operate, please refer to Chapter 5, "Dimensions CM Events Callout Interface" on page 157.

Interactions between the Two Architectures

When you design and implement a customization, using the *Event Architecture*, this customization is applied to all the standard Dimensions CM interfaces. This customization is also applied to all those applications that you have developed using the *Client Architecture.* Events, when they are deployed, literally become part of the Dimensions CM, and as a result are used by all Dimensions CM components.

Scope of the DTK Architecture

When you look at designing your applications for either of the specific architectures described above, there are a number of points that you must keep in mind:

- **DTK Applications using the Client Architecture**

  These applications are stand-alone utilities that interface with the Dimensions CM repository through the use of `PcmsConnect()` and other DTK functions.

- **DTK Applications using the Event Architecture**

  These applications interface with Dimensions CM through a public C function call named `userSuppliedFunction()`. These applications are built as shared libraries that are dynamically loaded by Dimensions CM. These applications are able to share information with the Dimensions CM server and so do not need to call `PcmsConnect()` or `PcmsDisconnect()`.

  For events to become active they only need to be deployed on the Dimensions CM server. As a result of this, each client that accesses that Dimensions CM server will use these events. You do not need to be concerned with deploying and controlling events on multiple client installations.

Chapter 2

Writing Dimensions CM DTK Applications

Introduction	38
DTK Return Codes	39
DTK Data Structures	40
DTK System Attribute Definitions	47
DTK Constant Definitions	52
Memory Allocation within the DTK	53

Introduction

This chapter outlines the data structures, manifest constants, and return codes that are used by the DTK. These structures and constants are defined in the provided `include` file `pcms_api.h` in the directory:

- `$DM_ROOT%/pcms_api` for UNIX.
- `%DM_ROOT%\pcms_api` for Windows.

Any source file that references DTK functions or constants *must* include this file.

CAUTION! As part of supporting MBCS (see Notes below), Serena® Dimensions® CM 10.1.2 or later comes with MBCS libraries enabled as default. This means that any API application—either on UNIX or Windows—must be compiled with the _MBCS define flag enabled. Failure to do so may cause your DTK application to exhibit memory errors when run. Please see the DTK examples for samples of how to define this flag—basically, on Windows you need to define /D "_MBCS" and on UNIX you need to define -D_MBCS as part of the compile line.

NOTES

1 In addition to `pcms_api.h`, there is also an include file called `pcms_len.h` for character array sizes.

2 The Dimensions CM 10.1.2 or later API supports MBCS (Multi-Byte Character Set). As part of this support, the legacy primitive type `char` has been replaced by `_TCHAR`. If you do not intend to utilize any MBCS or DBCS (Double-Byte Character Set) functionality, you can continue to use `char` as in previous releases of Dimensions. Note, UTF-8 is supported on non-Windows platforms.

3 The API library files are versioned to the number of the Dimensions CM release. For example, in Dimensions CM 10.1.3 the API libraries are named `pcms_api10m.lib` (Windows) and `pcms_api10.so (UNIX)`.

4 The Dimensions CM 2009 R1 or later API has added support for large files (4GBytes or greater)—so, for example, you will now be able to check in and fetch back a DVD ISO image. Consequently, the length of the corresponding item library file in such circumstances will need to exceed 4Gbytes.

To provide support for such item library files, internally Dimensions CM 2009 R1 or later uses 64-bit integers to represent library file lengths. Those 64-bit library lengths are stored as pairs of low-order 32-bits and high-order 32-bits.

The `PCMS_ITEM_DATA` published view (see the *Reports Guide*) has been enhanced to reflect this by providing a new column, `LIB_FILE_LENGTH_HI`, containing the high-order 32-bits of the file length. The low-order 32-bits of the file length continue to be stored in the `LIB_FILE_LENGTH` column as in releases of Dimensions CM prior to 2009 R1.

[continued on next page]

NOTES *[continued]*

This also means that the `LIB_FILE_LENGTH` and `FILE_LENGTH` item attributes can potentially refer to 64-bit values. If you store files larger than 4Gbytes in Dimensions CM, and use DTK applications or event callouts, then you need to ensure that your code uses correct C functions for converting attribute values into 64-bit integers and vice versa. Please refer to your compiler/C library vendor reference for information on how to convert 64-bit integers into null-terminated strings and vice versa.

DTK Return Codes

In general, when you call a DTK function the results of that function call can be determined by the return code given. There are three codes that a DTK function can return:

PCMS_OK	indicates that the function call succeeded, and objects were processed (for example, a query returned some information).
PCMS_FAIL	indicates that while the function call succeeded, no objects were actually processed (for example, a query returned no objects).
PCMS_ERROR	indicates that an error occurred trying to process the function call. If an error is encountered, the reason for the error can be assessed by examining the following variables.

`int PcmsErrorNo` `char *PcmsErrorStr`	If the error occurred due to a programming error, for example, invalid parameters were specified to a DTK function, these variables will be set detailing the reason for the error.
`int PcmsDbErrorNo` `char *PcmsDbErrorStr`	If the error occurred due to a database error, for example, the database is full, these variables will be set detailing the reason for the error. `PcmsDbErrorNo` will indicate the database error number (if any). `PcmsDbErrorStr` will indicate the SQL error string.

NOTE `ErrorNo` variables and `ErrorStr` variables will be set only when an error occurs.

DTK Data Structures

The results of function calls are generally stored in the data structures that are defined in the following sub-sections. These structures represent the abstraction of Dimensions CM objects and other information, and can be accessed through standard C constructions.

PcmsCallbackStruct

Definition

```
typedef struct PcmsCallbackStruct
{
   PcmsCallbackProc callback;
   void *clientData;
} PcmsCallbackStruct;
```

Description

This structure is used to hold information regarding registered callbacks. For more information please refer to `PcmsSetCallback()`.

PcmsChdAttachmentStruct

Definition

```
typedef struct PcmsChdAttachmentStruct
{
   int uid;
   _TCHAR opt;
   _TCHAR filename[(PCMS_L_FILENAME + 1)*CHARSIZEMAX];
   _TCHAR *userFile;
   _TCHAR dateTime[(PCMS_L_DATE_TIME + 1)*CHARSIZEMAX];
   _TCHAR userId[(PCMS_L_USER + 1)*CHARSIZEMAX];
   _TCHAR description[(PCMS_L_REMARK + 1)*CHARSIZEMAX];
}
PcmsChdAttachmentStruct;
```

Description

To support integration with the Application Lifecycle Framework (ALF), this structure is used to provide Dimensions CM event level access to Dimensions CM request attachments.

PcmsEventStruct

Definition

```
typedef struct
{
   char             *database;/* Dimensions RDBMS database */
                    /* identification */
   char             *baseDb;/* Base Database */
   int              eventId;/* PCMS_EVENT_XXX */
                    /* See userSuppliedFunction */
   int              objType;/* PCMS_ITEM, PCMS_PART, */
                    /* PCMS_CHDOC */
   int              noAttrsChanged; /*The number of attributes */
                    /* that changed */
   PcmsObjAttrStruct  *attrsChanged;
                    /* Attribute number of field that changed */
   int              whenCalled;/* PCMS_EVENT_VALIDATE_OP, */
                    /* PCMS_EVENT_PRE_OP or PCMS_EVENT_POST_OP */
} PcmsEventStruct;
```

Description

This structure is used exclusively for events and defines which event is being fired and with what parameters.

PcmsLcStruct

Definition

```
typedef struct
{
   char             normalPath;
   int              phase;
   char             status [PCMS_L_STATUS + 1];
   char             role [PCMS_L_ROLE + 1];
} PcmsLcStruct;
```

Description

This structure holds lifecycle definition information.

PcmsObjAttrDefStruct

Definition

```
typedef struct
{
   int              attr;/* Attribute number */
   int              valueMaxLen;/* The maximum length of the */
                    /* attribute */
   char             variable [PCMS_L_ATTR_VARIABLE + 1];
                    /*The attribute name */
   char             prompt [PCMS_L_ATTR_PROMPT + 1];
                    /* The screen prompt*/
   char             attrType;/* PCMS_ATTR_DATE = 'D' */
                    /* PCMS_ATTR_UNDEFINED = 'U' */
                    /* PCMS_ATTR_INTEGER = 'I' */
                    /* PCMS_ATTR_NUMBER = 'N' */
                    /* PCMS_ATTR_CHAR = 'C' */
   char             scope;/* PCMS_ATTR_ITEM='I' */
                    /* PCMS_ATTR_PART='P'*/
   char             display;/* Y or N */
   char             allRevisions;/* Y or N */
   char             manOpt;/* MANDATORY = 'Y' */
                    /* OPTIONAL = 'N' */
   char             fldUpd;/* 'Y' or 'N' */
   char             roleCheck [PCMS_L_ROLE + 1];
   char             uniqueVal;/* 'Y' *or 'N' /
   char             defaultVal[PCMS_L_ATTR_DEFAULT_VAL + 1];
   char             helpMess [PCMS_L_ATTR_HELP_MESS + 1];
   char             validationOn;/* Is validation enabled, */
                    /* Y or N */
   int              definedBy;/* PCMS_ATTR_PCMS/PROG/USER */
   char             hasLov;/* 'Y' or 'N'. List of Values */
                    /* must be used to set */
   void             **pp;/* Reserved */
   char             mva;/* 'Y' or 'N'.Multi-Valued */
                    /* use PcmsMva... macros to */
                    /* interpret. * /
   char             mvaType;/* not used currently */
   char             blockName [PCMS_L_ID + 1];
                    /* attr may belong to a */
                    /* display Block */
   int              blockColNo;/* Column number in the */
                    /* display block */
   int              displayWidth;/* Recommended display width */
   int              displayHeight;/* Recommended display height */
   char             multiLine;
                    /* 'Y' - use displayHeight ie. */
                    /* displayHeight > 0 */
                    /* 'N' - displayHeight not used */
   char             valueCase;  /* 'L' - Lower 'U' - Upper, */
                    /* or 'M' - Mixed */
   char             catalogDisplay;/* 'Y' or 'N' */
} PcmsObjAttrDefStruct;
```

Description

This structure is used to hold information relating to the attribute definition. Some of the fields for requests are currently hard-wired to the following values.

```
allRevisions   'N'
manOpt'        'N'
fldUpd         'Y'
roleCheck      '\0'
uniqueVal      'N'
```

PcmsObjAttrStruct

Definition

```
typedef struct
{
   int               attr;/* Attribute number */
   void              *value;/* The value of the attribute. */
                     /* See section on Attribute Macros */
   PcmsObjAttrDefStruct  *attrDef;
                     /* Pointer to the definition */
                     /* of the attribute */
} PcmsObjAttrStruct;
```

Description

This structure is used to hold information regarding the attributes that an object has. The `attr` member details the attribute number, while the `*attrDef` pointer contains details on the attribute definition. The value of the attribute is accessed through the `PcmsMvaGetVal()` and `PcmsSvaGetVal()` attribute macros. For more information about these macros, please refer to the "Attribute Macros" on page 138.

PcmsObjStruct

Definition

```
typedef struct PcmsObjStruct
{
    int         uid;
    int         specUid;
    int         objType; /* PCMS_ITEM, PCMS_PART, PCMS_CHDOC, etc */
    int         typeUid;
    _TCHAR      typeName[(PCMS_L_TYPE_NAME + 1)*CHARSIZEMAX];
    _TCHAR      productId[(PCMS_L_PRODUCT_ID + 1)*CHARSIZEMAX];
    _TCHAR      objId[(PCMS_MAX_L_ID + 1)*CHARSIZEMAX];
    _TCHAR      variant[(PCMS_L_VARIANT + 1)*CHARSIZEMAX];
    _TCHAR      revision[(PCMS_L_REVISION + 1)*CHARSIZEMAX];
    _TCHAR      description[(PCMS_L_DESCRIPTION + 1)*CHARSIZEMAX];
    _TCHAR      userName[(PCMS_L_USER + 1)*CHARSIZEMAX];
    _TCHAR      status[(PCMS_L_STATUS + 1)*CHARSIZEMAX];
    _TCHAR      dateTime[(PCMS_L_DATE_TIME + 1)*CHARSIZEMAX];
    _TCHAR      isExtracted;
    int         noAttrs; /* The number of attributes for this object. */
    PcmsObjAttrStruct *attrs; /* Pointer to the array of attributes. */
    int         specUid;
}   PcmsObjStruct;
```

Description

This is the generic structure used for Dimensions CM objects such as items, parts, requests, and baselines. The type of object is defined by the member field `objType` being set to a specific constant (for example, `PCMS_ITEM`). The `*attrs` pointer can be used to access attribute information if it is defined. The `specUid` field contains the object's specification `uid`, where applicable, or `-1`.

PcmsPendingUserStruct

Definition

```
typedef struct
{
   char             user [PCMS_L_USER+ 1];/* User */
   char             role [PCMS_L_ROLE + 1];/* Role */
   char             capability;/* The user's capability in */
                    /* role, either */
                    /* 'P' - Primary or */
                    /* 'S' - Secondary or */
                    /* 'L' – Leader */
char nextStatus [PCMS_L_STATUS + 1];
                    /* next possible status */
   int              nextPhase;/* next phase */
   char             actionable;
                    /* PCMS_ACT_NOT_LEADER = '1' */
                    /* (Can't action not leader) */
                    /* PCMS_ACT_OK = '2' */
                    /* (Can action no leaders) */
```

```
                    /* PCMS_ACT_LEADER = '3' */
                    /* (Can action I am a leader) */
} PcmsPendingUserStruct;
```

Description

This structure holds information relating to who can action an object and to what states.

PcmsPendStruct

Definition

```
typedef struct
{
   int              objUid;
   int              objType;
   char             capability;/* 'P' - Primary, */
                    /* 'S'- Secondary, */
                    /* 'L' - Leader*/
   char             actionable;/* PCMS_ACT_NOT_LEADER = '1'*/
                    /* (can't action not leader), */
                    /* PCMS_ACT_OK = '2' */
                    /* (Can action no leaders), */
                    /* PCMS_ACT_LEADER = '3' */
                    /* (Can action I am a leader) */
} PcmsPendStruct;
```

Description

This structure holds object inbox information.

PcmsRelStruct

Definition

```
typedef struct
{
   int              uid;
   int              objType;
   int              relType;
   int              relSubTypeUid;
                    /* Refers to the uid field of a */
                    /* PcmsRelTypeStruct. For requests, users */
                    /* may setup up specialization's of the basic */
                    /* relTypes to add attributes, etc. See */
                    /* section on PcmsGetUserRels */
} PcmsRelStruct;
```

Description

This structure is used in conjunction with `PcmsRelTypeStruct` and stores the relationships that an object has.

PcmsRelTypeStruct

Definition

```
typedef struct
{
   int              uid;/* of this relationship subType */
   char             name [ PCMS_L_ID + 1];
                    /* Name for this rel subtype*/
   int              relType;
                    /* The parent relationship type */
                    /* for, example, PCMS_REL_INFO */
   char             productId [ PCMS_L_PRODUCT_ID  + 1];
                    /* Product name */
} PcmsRelTypeStruct;
```

Description

This structure is used to hold information relating to the user-defined types used in Dimensions CM. For example, 'request to request' relationships.

PcmsRoleStruct

Definition

```
typedef struct PcmsRoleStruct
{
   char             role [PCMS_L_ROLE + 1];/* Role */
   int              uid;/* uid of the part */
   char             leader;/* 'Y'  = Leader only role */
} PcmsRoleStruct;
```

Description

This structure holds information of the roles defined on a Dimensions CM product.

PcmsTypeStruct

Definition

```
typedef struct
{
   int              uid;
   char             productId [PCMS_L_PRODUCT_ID + 1];
   char             typeName [PCMS_L_TYPE_NAME + 1];
   int              objType;/* PCMS_PART, PCMS_ITEM, */
                    /* PCMS_CHDOC etc*/
   char             lifecycle [PCMS_L_ID + 1];
   char             typeDescription [PCMS_L_DESCRIPTION + 1];
   char             superType;/* '1', '2', '3', or '4'*/
                    /* for requests only */
} PcmsTypeStruct;
```

Description

This structure holds object type definition information.

PcmsUserRoleStruct

Definition

```
typedef struct
{
   char             user [PCMS_L_USER + 1];/* User */
   char             role [PCMS_L_ROLE + 1];/* Role */
   char             capability;/* The users capability in */
                    /* this role, either */
                    /* 'P' - Primary or */
                    /* 'S' - Secondary or */
                    /* 'L' - Leader */
   char             applyDeny;/* applyDeny flag not */
                    /* currently used */
   char             treeWalk;/* treeWalk flag not */
                    /* currently used */
   char             actionable;
                    /* PCMS_ACT_NOT_LEADER = '1' */
                    /* (Can't action not leader) */
                    /* PCMS_ACT_OK = '2' */
                    /* (Can action no leaders) */
                    /* PCMS_ACT_LEADER = '3' */
                    /* (Can action I am a leader) */
   char             fromTree;/* 'Y' This role was found */
                    /* from the Part Structure */
                    /* 'N' This role was delegated */
                    /* using the DLGC command */
} PcmsUserRoleStruct;
```

Description

This structure is used to hold information relating to role assignments.

DTK System Attribute Definitions

A Dimensions CM object can have two kinds of attributes:

- User-defined attributes.

  These are attributes defined by you in the process model.

- System-defined attributes.

  These are attributes definitions that are internal to Dimensions CM. These attributes are provided to allow you to access information that might be useful.

The table below details the system attributes that are available for each object type.

NOTE The number of system-defined attributes are defined by the constant `PCMS_NUM_<objtype>_ATTRS`, for example, `PCMS_NUM_ITEM_ATTRS`.

Object Type **PCMS_ :**	Number of System Attributes **PCMS_NUM_ :**	Attribute Reference **PCMS_ATTR_ :**	Definition
BASELINE	BLN_ATTRS	BASELINE_METHOD	Indicates how the baseline was created. Possible values are: Created = Created through CBL. Revised = Created through CRB. Merged = Created through CMB.
		BASELINE_TYPE	The type of baseline created 1 = Design 2 = Release 3 = Archive
		CREATE_DATE	Create date of the baseline
		SENDER_ID	This attribute is only populated if this baseline was created as a result of replication. This will correspond to the database that sent the baseline.
		TEMPLATE	Template name used to create the baseline.
CHDOC	CHD_ATTRS	CHSEQ	Sequence number of the Dimensions request.
		CREATE_DATE	Create date of the request.
		DELEGATED_ OWNER_SITE	Issue replication attribute: delegated owner site.
		DELEGATED_ REFERENCE_ONLY	Issue replication attribute: reference only request.
		LIFECYCLE	Lifecycle assigned to the request.
		NO_ACTIONS	Number of actions on the request.
		ORIGINATOR	Originator of the request.

Object Type PCMS_ :	Number of System Attributes PCMS_NUM_ :	Attribute Reference PCMS_ATTR_ :	Definition
		OWNER_SITE	Issue replication attribute: current owner site.
		PHASE	Current phase of the request.
		SUPER_TYPE	Super type of the request.
		UPDATE_DATE	Update date of the request.
ITEM	ITEM_ATTRS	CHECKSUM	Checksum of the project file.
		COMPRESSED	If the item is compressed.
		CREATE_DATE	Date of item creation.
		DIR_UID	Project directory uid.
		DIRPATH	Project directory path.
		EDITABLE	If the item is editable.
		FILE_LENGTH	Length of the project file. See Step 4 of "Introduction" on page 38 for a discussion of support for files greater than 4Gbytes.
		FILE_VERSION	File version in the library.
		FILENAME	Project filename (current project).
		FORMAT	Item format.
		ITEM_SPEC_UID	Item spec uid.
		LIB_CHECKSUM	Library item file checksum.
		LIB_FILE_LENGTH	Length of the item file in the library (low-order 32-bits). See Step 4 of "Introduction" on page 38 for a discussion of support for files greater than 4Gbytes.
		LIB_FILENAME	Library item filename. See Step 4 of "Introduction" on page 38 for a discussion of support for files greater than 4Gbytes.

Object Type PCMS_ :	Number of System Attributes PCMS_NUM_ :	Attribute Reference PCMS_ATTR_ :	Definition
		LIB_FILENAME_HI	Length of the item file in the library (high-order 32-bits). See Step 4 of "Introduction" on page 38 for a discussion of support for files greater than 4Gbytes.
		LIFECYCLE	Lifecycle id that is followed by the item.
		LOCKED_USER	The user who last locked/ unlocked this file.
		LOCKED_DATETIME	The date that the file was last locked/unlocked.
		LOCKED_FILE	Indicates if the file is currently locked.
		ORIGINATOR	Who created the item.
		PHASE	The item's current phase.
		REVISED_DATE	The item's last revised date in Julian date format (this is the equivalent of Oracle's date format: 'J.SSSSS'). A Julian date is the number of days since Jan 1, 4712 BC. Julian dates allow continuous dating from a common reference.
		SENDER_ID	This attribute is only populated if this item was created as a result of replication. If an item has been remotely replicated, then this will correspond to the database that sent the item. If an item has been replicated locally, then this will correspond to the replication configuration identifier.
		SHARED_BRANCH	This attribute is reserved for future use.
		STATUS	Status of the item.
		USER_FILENAME	The user filename resulting from check out, update, check in, etc, operations.

Object Type PCMS_ :	Number of System Attributes PCMS_NUM_ :	Attribute Reference PCMS_ATTR_ :	Definition
PART	PART_ATTRS	LOCALNO	Local Part Number.
		PARTNO	Part Number.
		PHASE	Part Phase.
WORKSET	WORKSET_ATTRS	CMPATHCONTROL	If request path control is enabled for this project.
		CMRULES	IF CM Rules are enabled for this project.
		DEFAULTCHDOC	Your default working request for this project.
		DEPLOYMETHOD	The project’s deployment method.
		ENFORCE_REV	If revision generation is enforced.
		LOCKED	Indicates if a project or stream is currently locked.
		PHASE	The project's current phase.
		STREAMPROJECT	If this project is a stream.
		TRUNK	If this is a trunk project.
		WS_DIR	User's default project directory.
		WS_PART	The project’s default product part.
USER	USER_ATTRS (Additional user properties that are defined when the user is created.)	BASEDB	The name of the base database.
		DEPT	The user’s department.
		EMAIL	The user's email address.
		FULL_USERNAME	The user’s full name.
		GROUP	The user’s group.
		PHONE	The user’s phone number.

Object Type PCMS_ :	Number of System Attributes PCMS_NUM_ :	Attribute Reference PCMS_ATTR_ :	Definition
		PRIVILEGE_LEVEL	The user's privilege.
		SITE	The user's site name.

DTK Constant Definitions

The following constant definitions are used within the DTK to represent Dimensions CM objects, relationships, and phases.

- **Object types (objType)**

  Object types are constants used in the DTK to indicate classes of objects (for example, items).

Constant	Object Definition
PCMS_BASELINE	Dimensions CM Baselines
PCMS_CHDOC	Dimensions CM Request
PCMS_CUSTOMER	Dimensions CM Customers
PCMS_ITEM	Dimensions CM Items
PCMS_PART	Dimensions CM Design Parts
PCMS_USER	Dimensions CM Users
PCMS_WORKSET	Dimensions CM Projects/Streams
PCMS_DIRECTORY	Dimensions CM Directories

- **Relationships (relType)**

  Relationship types are constants used in the DTK to indicate relationships between objects (for example, usage).

Constant	Relationship Definition
PCMS_REL_AFF	Affected
PCMS_REL_BREAKDOWN	Owner
PCMS_REL_DEP	Dependent
PCMS_REL_DERIVED	Built item
PCMS_REL_INFO	Information
PCMS_REL_IRT	In Response To
PCMS_REL_OWN	Same as for PCMS_REL_BREAKDOWN
PCMS_REL_PRED	Predecessor

Constant	Relationship Definition
PCMS_REL_SUCC	Successor
PCMS_REL_TOP	Top owner object
PCMS_REL_USE	Usage
PCMS_REL_PARENT	Object parent
PCMS_REL_CHILD	Object child

- **Relationship sub-types (relSubType)**

  Relationship sub-types are constants used in the DTK to indicate additional information about relationships between objects.

Constant	Relationship Definition
PCMS_REL_PERMANENT	A sub-class of relationship between a request and a baseline that details a relationship created as a result of a revised baseline operation. This relationship sub-class is used in the `relSubTypeUid` bit mask that is part of the `PcmsRelStruct` structure. This relationship information is only retrieved as a result of a `PcmsObjGetRels()` or `PcmsObjGetBackRels()` operation.

- **Phases**

Phases are generic indicators that are used to show where a particular object is in its lifecycle. Please refer to the related document *Introduction to Dimensions CM* for more information.

Memory Allocation within the DTK

Some platforms, such as Windows, require that the shared library that allocated memory is also responsible for freeing that memory. Because of this a number of wrapper functions to the standard C memory functions have been provided.

DTK Function	Wrapper to function
PcmsEvntFree()	free()
PcmsEvntMalloc()	malloc()
PcmsEvntCalloc()	calloc()
PcmsEvntRealloc()	realloc()

Usage of the Functions

- General Usage of these Functions to Allocate Memory

  If any application, either in events or clients, requires you to allocate memory that will be used by DTK functions, then you must use the wrappers as listed above. For example, `PcmsQuery()` allows you to dynamically allocate memory to the `PcmsObjStruct attrs` pointer to define user-defined filters. This memory must be allocated and re-allocated through the use of the wrappers listed above. If you are allocating memory that is not used by the DTK, then you do not have to use these wrappers. However, it is strongly recommended for consistency that you use these wrappers for any memory allocation that you make.

- General Usage of these Functions to De-allocate Memory

  If any memory has been allocated by the DTK, or by the wrapper functions described above, then this memory must be freed through `PcmsEvntFree()`.

- Within DTK client applications, any memory that has been allocated by DTK functions, such as `PcmsQuery()`, must be freed by the function `PcmsEvntFree()`. For example, if you have the following call:

  ```
  int            *uids = 0;
  int            noUids = 0;

  if (PcmsQuery(conId, &queryObj,0,&noUIds,&uids)!=PCMS_OK)
         PcmsEvntFree(uids);
  ```

  then you would use `PcmsEvntFree()` function to free this memory. You do not need to use these functions if you are allocating memory that is not used by the DTK. However, to be consistent in the memory functions that you do use, it is strongly recommended that you use these functions for all memory allocation and de-allocation.

- Within DTK events these functions must always be used to allocate or de-allocate memory. This includes both memory usage with events and with reference to DTK function calls, for example:

  ```
  char  *txt = (char *)PcmsEvntMalloc(15);

  status = PcmsQuery(conId, &queryObj,0,&noUIds,&uids);
  PcmsEvntFree(uids);
  PcmsEvntFree(txt);
  ```

To minimize the impact of any code changes, it is suggested that you redefine the standard C functions to use the new functions through the use of `#define(s)`, for example:

```
#define     free            PcmsEvntFree
#define     malloc          PcmsEvntMalloc
....
```

and then recompile the event code. Please note, however, that the prototype of the function `PcmsEventCalloc()` is not the same as `calloc()`. Please consult Chapter 3, "DTK API Functions for C/C++" for more information.

NOTE These functions must always be used when freeing memory that has been allocated by DTK function calls (such as `PcmsQuery()`) both within events and within DTK client programs.

The table below is provided as a guideline to help you identify which functions and pointers are dynamically assigned memory by the DTK, and what functions you should use to free that memory. Ensure that you refer to the DTK function description for more information on when this memory may be assigned.

Function Name	Pointer to Free	Free through
PcmsAttrDefInit()	PcmsObjAttrDefStruct *ptrDef	PcmsEvntFree()
PcmsAttrGetLov()	char ***ptrVal char **ptrMess	PcmsLovFree() PcmsEvntFree()
PcmsClntApiGetLastErrorEx()	char **errorBuffer	PcmsClntApiFree()
PcmsCntrlPlanGet()	void **ptr	PcmsEvntFree()
PcmsExecCommand()	char **ptrResponse	PcmsEvntFree()
PcmsFullQuery()	PcmsObjStruct **ptrObjs	PcmsObjFree()
PcmsGetAttrFile()	char **ptrFile	PcmsEvntFree()
PcmsGetAttrs()	PcmsObjStruct *ptrObj	PcmsObjFree()
PcmsGetCandidates()	char ***ptrCans	PcmsEvntFree()
PcmsGetPendingUsers()	PcmsPendingUserStruct **ptrUsers	PcmsEvntFree()
PcmsGetRSAttrs()	int **ptrAttrs char **ptrDefRole	PcmsEvntFree()
PcmsGetRSNames()	char **ptrMessage char ***ptrVal	PcmsEvntFree()
PcmsGetUserRelTypes()	PcmsRelTypeStruct **ptrRels	PcmsEvntFree()
PcmsGetUserRoles()	PcmsGetUserRoles **ptrRoles	PcmsEvntFree()
PcmsObjGetBackRels()	PcmsRelStruct **ptrRels	PcmsEvntFree()
PcmsObjGetRels()	PcmsRelStruct **ptrRels	PcmsEvntFree()
PcmsPendGet()	PcmsPendStruct **ptrPend	PcmsEvntFree()
PcmsPendWhoGet()	PcmsPendStruct **ptrPend	PcmsEvntFree()
PcmsPopulate()	PcmsObjStruct **ptrObj	PcmsObjFree()
PcmsQuery()	int **uids	PcmsEvntFree()

Chapter 3

DTK API Functions for C/C++

Introduction	59
Memory Allocation by DTK Functions	59
Access Security in DTK Functions	59
PcmsAttrDefInit - Get Attribute Definition	61
PcmsAttrGetLov - Get Attribute's List of Values	62
PcmsAttrValidate - Validate an Attribute Value	65
PcmsCheckMessages - Check Results of Dimensions CM Command	66
PcmsCntrlPlanGet - Get Dimensions CM Process Model Information	68
PcmsConnect - Connect to Dimensions CM Database	71
PcmsDisconnect - Disconnect from a Dimensions CM Database	73
PcmsEvntCalloc – Allocate Zero Initialized Memory	74
PcmsEvntFree – Free Memory	75
PcmsEvntMalloc – Allocate Memory	76
PcmsEvntRealloc – Re-Allocate Memory	77
PcmsExecCommand - Execute Dimensions CM Command Synchronously	78
PcmsFullQuery - Find Dimensions CM Objects, Returning Complete Objects	80
PcmsGetAttachments - Obtain Request Attachment Structures	84
PcmsGetAttrDefNum - Get Attribute Definition Number	85
PcmsGetAttrFile - Get Request Descriptions	86
PcmsGetAttrs - Get Dimensions CM Object Attributes	88
PcmsGetBaseDbOptions - Retrieve Base Database Options	89
PcmsGetCandidates - Retrieve Candidates for Delegation	90
PcmsGetCommandLine – Get the Dimensions CM Command	91
PcmsGetConnectDesc - Get Input File Descriptor	92
PcmsGetDimensionsVersions - Obtain List of Loaded DLLs and Dates and Times	93
PcmsGetPendingUsers - Obtain Inbox User Structures	94
PcmsGetRSAttrs - Retrieve Attribute Numbers in a Role Section	96
PcmsGetRSNames - Obtain Role Section Names for a Product	98
PcmsGetStringSymbolValue - Obtain Values Associated with a Symbol	100
PcmsGetSymbolInfo - Look Up Symbol in Structured Information Return Table	101
PcmsGetUserRelTypes - Obtain User Relationship Subtypes	103
PcmsGetUserRoles - Obtain User Role Structures	105
PcmsGetWsetObj - Get User's Current Project	107
PcmsInitSpec - Get Dimensions CM Object Details by Specification	108

PcmsInitUid - Get Dimensions CM Object Details by Uid	109
PcmsLovFree - Free a List of Values	110
PcmsObjFree - Free Dimensions CM Object Structures	111
PcmsObjGetBackRels - Get Dimensions CM Object Reverse Relationships	112
PcmsObjGetRels - Get Dimensions CM Object Relationships	114
PcmsObjInSecondary - Is Request Object in Secondary Catalog	116
PcmsPendGet - Retrieve Dimensions CM Object Inboxes for a User	117
PcmsPendWhoGet - Retrieve Users for Object	119
PcmsPopulate - Populate an Object's Attributes Values	121
PcmsQuery - Find Dimensions CM Objects, Returning Uids	122
PcmsSendCommand - Execute Dimensions CM Command Asynchronously	126
PcmsSetAttrs - Set Dimensions CM Object Attributes	128
PcmsSetCallback - Set Dimensions CM API Server Callback	130
PcmsSetDbErrorCallback - Set Server Error Callback	132
PcmsSetDirectory - Change Dimensions CM Default Directory	134
PcmsSetIdleChecker - Install Idle Checker	135
PcmsSetWsetObj - Set User's Current Project	136
PcmsWriteAttachments - Write Specified Request Attachments To Disk	137
Attribute Macros	138

Introduction

This chapter describes each of the functions that are available in the Serena® Dimensions® CM DTK for C/C++ programs. The description of each function has the following components.

Purpose	What the function does.
Prototype	The function prototype.
Parameters	Description of the parameters used in the function.
Return Codes	Codes returned (please refer to "DTK Return Codes" on page 39 for further details).
Sample	Sample function call (if applicable).
Comments	Additional relevant information (if applicable).
Related Functions	Any related DTK function calls.

Before reading this chapter, ensure that you have familiarized yourself with the contents of Chapter 2, "Writing Dimensions CM DTK Applications" because it contains important information relating to data structures, return codes, and manifest constants that are referenced in this chapter.

Memory Allocation by DTK Functions

A number of the DTK functions allocate memory to pointers that then becomes the responsibility of the calling application to free. On some operating systems, such as Windows and Solaris, memory that has been allocated by a shared library must be freed by that same shared library. You must free this memory through the function call `PcmsEvntFree()`. If you do not use this function, you may experience memory corruption. For more information please refer to "Memory Allocation within the DTK" on page 53.

Access Security in DTK Functions

If your Dimensions CM installation has been configured to support product-level security, then the following DTK functions will filter out or limit access to only those Dimensions CM objects that a user has the rights to access:

- `PcmsAttrDefNum()`
- `PcmsAttrGetLov()`
- `PcmsAttrValidate()`
- `PcmsCntrlPlanGet()`
- `PcmsFullQuery()`
- `PcmsGetAttachments()`

- PcmsGetAttrFile()
- PcmsGetAttrs()
- PcmsGetCandidates()
- PcmsGetPendingUsers()
- PcmsGetRSAttrs()
- PcmsGetRSNames()
- PcmsGetUserRelTypes()
- PcmsGetUserRoles()
- PcmsInitSpec()
- PcmsInitUid()
- PcmsObjGetBackRels()
- PcmsObjGetRels()
- PcmsPendWhoGet()
- PcmsPopulate()
- PcmsQuery()
- PcmsSetAttrs()
- PcmsWriteAttachments()

PcmsAttrDefInit - Get Attribute Definition

Purpose

This function retrieves an attribute definition for a specified `typeUid`, object type, and attribute number.

Prototype

```
int
PcmsAttrDefInit (
    int                 connectId,
    int                 typeUid,
    int                 objType,
    int                 attrNum,
    PcmsObjAttrDefStruct **ptrDefStruct
);
```

Parameters

`connectId`	is the database connection identifier.
`typeUid`	is the type (the `typeUid` field of the `PcmsObjStruct`) to which the attribute applies.
`objType`	is the type of the object. You can use `PCMS_PART`, `PCMS_ITEM`, `PCMS_BASELINE`, `PCMS_USER`, `PCMS_CHDOC`, or `PCMS_WORKSET`.
`attrNum`	is the attribute number for which the definition is to be retrieved.
`ptrDefStruct`	is the address of a pointer to a `PcmsObjAttrDefStruct` in which the attribute definition will be stored. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsAttrDefInit()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find the specified attribute number for the given type and object type.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo` and `PcmsErrorStr`.

Related Functions

`PcmsCntrlPlanGet()`.

PcmsAttrGetLov - Get Attribute's List of Values

Purpose

This function retrieves the list of valid set values that are allowed for a specified object and attribute. The list of values is returned as an array of `char *`(s).

The `PcmsObjStruct` that is used in this function must have at least the following member fields defined.

- The `typeUid` set to the Dimensions CM type against which the attribute has been assigned.
- The `noAttrs` field set to at least 1.
- The `*attrs` pointer set to a `PcmsObjAttrStruct` structure, which must have the member fields set as follows:
  - `attr`: attribute number you are querying
  - `value`: a `" "` string through `PcmsSvaSetVal()`
  - `attrDef`*:* the corresponding `PcmsObjAttrDefStruct`*.*

It is possible to obtain the `typeUids` for a product and `objType` through `PcmsCntrlPlanGet()`. A certain attribute definition can then be obtained by calling `PcmsAttrDefInit()`.

Prototype

```
int
PcmsAttrGetLov
 (
    int             connectId,
    PcmsObjStruct  *objPtr,
    int             attrNum,
    char            **message,
    int             *noStrings,
    char            ***ptrArrayOfStrings
);
```

Parameters

`connectId`	is the database connection identifier.
`objPtr`	is the object containing the type and attribute details.
`attrNum`	is the attribute number.
`message`	is the error message returned if the status is not `PCMS_OK`. It is the caller's responsibility to free this allocated memory if not `NULL`.
`noStrings`	is the address of an integer variable to contain the number of strings in the array.
`ptrArrayOfStrings`	is the address of a pointer to the array of returned strings. This array must be freed through `PcmsLovFree()`*.*

Return Codes

PcmsAttrGetLov() returns:

PCMS_OK	on success.
PCMS_FAIL	on failure to find the specified attribute number for the given type and object type.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Sample

```
/*
    *-------------------------------------------------------------
    *       FUNCTION SPECIFICATION
    *       Name:
    *        GetLovs
    *       Description:
    *        Get LOVs for a specified attribute and typeUid
    *       Parameters:
    *        int  conId
    *        int  objType
    *        int  typeUid
    *        int  attr
    *       Return:
    *        int
    *       Notes:
    *-------------------------------------------------------
    */
int GetLovs(int conId, int objType, int typeUid, int attr)
{
   PcmsobjStruct obj = { 0 };
   PcmsObjAttrDefStruct *attrDef = 0;
   char **vals = 0;
   int  noVals = 0;
   int  x = 0;
   char *ptrError = 0;
   int  status = 0;
   obj.typeUid = typeUid;
   obj.noAttrs = 1;
   /* Get the details on the specified attribute */
   if ((status = PcmsAttrDefInit(conId,typeUid
     objType,attr,&attrDef))!=PCMS_OK)
     return status;
   /* Put together a dummy object structure */
   obj.attrs =
     (PcmsObjAttrStruct*)PcmsEvntMalloc(sizeof(PcmsObjAttrStruct)*1);
   obj.attrs[0].attr = attr;
   PcmsSvaSetVal(obj.attrs[0].value,NULL,0);
   obj.attrs[0].attrDef = attrDef;
   /* Get the LOV values */
   if ((status = PcmsAttrGetLov(conId,&obj,
                   attr,&ptrError,
                   &noVals,&vals))==PCMS_OK)
```

```
    {
       int i = 0;
       /**
       ** Scan list of LOVs and report

                          **/
       for(i=O;i<noVals;i++)
            (void)fprintf(stdout,
                       "\nLov[%d/%d] - %s",
                       i,noVals,vals[i]);
       PcmsLovFree(noVals,vals);
    }
    void)PcmsObjFree(&obj);
    return(status);
}
```

Related Functions

`PcmsAttrDefInit()`, `PcmsAttrValidate()`.

PcmsAttrValidate - Validate an Attribute Value

Purpose

This function verifies that any attribute values specified on a given object structure do not conflict with any valid sets that may have been defined.

Prototype

```
int
PcmsAttrValidate (
    int             connectId,
    PcmsObjStruct  *objPtr,
    int             attrNum,
    char            **message
);
```

Parameters

`connectId`	is the database connection identifier.
`objPtr`	is the object containing the attribute values to be validated.
`attrNum`	is the attribute number against which the object attributes will be validated.
`message`	is the error message returned if the status is not `PCMS_OK`. It is the caller's responsibility to free this allocated memory if not `NULL`.

Return Codes

PcmsAttrValidate() returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Related Functions

`PcmsAttrGetLov()`, `PcmsLovFree()`.

PcmsCheckMessages - Check Results of Dimensions CM Command

CAUTION! This function is deprecated. It no longer submits commands asynchronously. You should, instead, use the synchronous function PcmsExecCommand, see page 78.

Purpose

This function may be used to check whether the results from Dimensions CM commands previously submitted, using PcmsSendCommand(), are available.

Prototype

```
int
PcmsCheckMessages (
    int     connectId,
    int     flag
);
```

Parameters

connectId	is the database connection identifier.
flag	is used to determine whether the operation is to be blocking (PCMS_MSG_WAIT) or non-blocking (PCMS_MSG_NOWAIT).

Return Codes

PcmsCheckMessages() returns:

PCMS_OK	results of a command are available and the callback function has been invoked.
PCMS_FAIL	results of a command are not available (non-blocking operation).
PCMS_ERROR	on failure and sets PcmsErrorNo and PcmsErrorStr.

Comments

If there are no commands being processed on this connection, PCMS_FAIL will be returned. If there are outstanding commands, the operation of this function will depend on the value of flag.

- If flag is equal to PCMS_MSG_WAIT, the function will block (by calling the function set with PcmsSetIdleChecker()) until the results of the next command are available. The function will then invoke the callback function (see PcmsSetCallback() on page 130) and return PCMS_OK.

- If `flag` equals `PCMS_MSG_NOWAIT`, the function will return immediately if no results are available (after calling the function set with `PcmsSetCallback()`), and the return value will be `PCMS_FAIL`. If results are available, the callback function will be invoked and the value `PCMS_OK` returned.

Related Functions

`PcmsSendCommand()`, `PcmsSetCallback()`, `PcmsSetIdleChecker()`, `PcmsExecCommand()`.

PcmsCntrlPlanGet - Get Dimensions CM Process Model Information

Purpose

This function queries the Dimensions CM process model and returns the data in various different structures.

Prototype

```
int
PcmsCntrlPlanGet (
    int    connectId,
    int    reserved,
    int    options,
    char   *fromId,
    int    objUid,
    char   *startContext,
    int    *noStructs,
    void   **ptrStructs
);
```

Parameters

- When `options = PCMS_CHD_TYPE`:

`objUid`	is `0` for all request types or contains a *typeUid* from a `PcmsObjStruct` structure.
`fromId`	is the Dimensions CM product from which to retrieve the request types when `objUid` is `0`. This parameter is ignored when `objUid` is not 0.
`startContext`	is `NULL` or a valid Dimensions CM super_type cast to a `char *`. This parameter is ignored when `objUid` is not `0`.
`ptrStructs`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsTypeStruct`.

- When `options = PCMS_PART_TYPE` or `PCMS_PART_CATEGORY`:

`objUid`	is `0` for all part types or contains a `typeUid` from a `PcmsObjStruct` structure.
`fromId`	is the Dimensions CM product from which to retrieve the design part types when `objUid` is `0`. This parameter is ignored when `objUid` is not `0`.
`startContext`	is ignored.
`ptrStructs`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsTypeStruct`.

- When `options = PCMS_BASELINE`:

`objUid`	is `0` for all baseline types or contains a `typeUid` from `PcmsObjStruct` structure.
`fromId`	is the Dimensions CM product from which to retrieve the baseline types when `objUid` is `0`. This parameter is ignored when `objUid` is not `0`.
`startContext`	is ignored.
`ptrStructs`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsTypeStruct`.

- When `options = PCMS_LC`:

`fromId`	is the lifecycle-id.
`objUid`	is ignored.
`startContext`	is `NULL` (in which case the first lifecycle state is returned) or is a valid state within the lifecycle (in which case the next possible states will be returned).
`ptrStructs`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsLcStruct`.

- When `options = PCMS_ATTRIBUTE` OR-ed with `PCMS_OBJ_TYPE` OR-ed with (`PCMS_ITEM` or `PCMS_PART` or `PCMS_CHDOC` or `PCMS_USER` or `PCMS_BASELINE` or `PCMS_WORKSET`):

`fromId`	is the Dimensions CM product from which to retrieve the attribute definitions when `objUid` is `0`. This parameter is ignored when `objUid` is not `0`.
`objUid`	is `0` or the uid of the type for which to retrieve the attribute definitions.
`startContext`	is `NULL` or a valid name of a Dimensions CM type (the `typeName` field of the `PcmsTypeStruct`). If a `startContext` is specified, then only attribute definitions that have been specified in the documentation plan will be returned. This parameter is ignored when `objUid` is not `0`.
`ptrStructs`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsObjAttrDefStruct`.

- The following parameters apply whatever the definition of `options`.

`connectId`	is the database connection identifier.
`reserved`	is reserved for future use.
`noStructs`	is a pointer to an integer variable in which to store the number of structures returned in `ptrStructs`. If no objects are found `noStructs` is set to zero and `ptrStructs` is set to `(void *) zero`.

It is the responsibility of the calling application to free the `ptrStructs` pointer when it is no longer required.

Return Codes

`PcmsCntrlPlanGet()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	when no objects were found.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

PcmsConnect - Connect to Dimensions CM Database

Purpose

This function provides you with a connection to the Dimensions CM database (for example, `intermediate`) specified by the input parameters. You can use this function to open multiple connections on the same or different Dimensions CM databases. This function will return a `connectId` (integer) that represents your database connection. The database parameters reflect the same values as you would specify for a `DMDB` symbol.

Prototype

```
int
    PcmsConnect (
    char    *database,
    char    *password,
    char    *node
);
```

Parameters

`database`	is the name of the Dimensions CM database to connect to.
`password`	is the password of the database. Use `NULL` if the user is secure.
`node`	is the RDBMS Service Name assigned to the node where the RDBMS database is located.

Return Codes

`PcmsConnect()` returns:

`int connectId`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

This function will not return until a successful connection has been made, or an error is encountered while attempting the connection.

If you wish to use the default Dimensions CM database for this user (DMDB), then invoke `PcmsConnect()` with `NULL` parameters for database, password, and node.

This function will apply all the same pre-login user-verification checks as if the user had typed `dmcli` at the command prompt.

Sample

```
/*
 * connect to Dimensions
 *
 */
int connect()
{
    int conId = PCMS_ERROR
    /* CONNECT to DMDB */
    conId = PcmsConnect(NULL, NULL, NULL);
    return conId;
}
```

Related Function

PcmsDisconnect().

PcmsDisconnect - Disconnect from a Dimensions CM Database

Purpose

This function disconnects from the Dimensions CM database as specified by the `connectId`. This `connectId` must be a valid `connectId` returned by `PcmsConnect()`.

Prototype

```
int
PcmsDisconnect (
    int    connectId
);
```

Parameters

`connectId`	is the database connection identifier.

Return Codes

`PcmsDisconnect()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure, and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Sample

```
/* Disconnect and exit with success… */
(void) PcmsDisconnect(conID);
return (EXIT_SUCCESS);
```

Comments

On some operating systems if you do not call this function before the application exits, the connection to the repository may never terminate.

This function does not return until the disconnection is complete.

Related Function

`PcmsConnect()`.

PcmsEvntCalloc – Allocate Zero Initialized Memory

Purpose

This function is a wrapper to the C function `calloc()`. It must be used to allocated zero initialized memory within a DTK application. The reason for this is that on some platforms, like Windows, the memory that is allocated within a shared library *must* be freed in the same shared library. If this is not done, then memory errors begin to occur. This function is thus provided as a convenience to ensure that all the memory is allocated within the context of the same shared library.

Prototype

```
void* PcmsEvntCalloc (int size);
```

Parameters

`size`	is the size of the memory to allocate.

PcmsEvntFree – Free Memory

Purpose

This function is a wrapper to the C function `free()`. It must be used to free memory allocated by DTK functions, such as `PcmsQuery()`. The reason for this is that on some platforms, like Windows, the memory that is allocated within a shared library *must* be freed by the same shared library. If this is not done, then memory errors begin to occur.

Prototype

```
void PcmsEvntFree (void *ptr);
```

Parameters

`ptr`	is a pointer to the memory block that will be freed.

PcmsEvntMalloc – Allocate Memory

Purpose

This function is a wrapper to the C function `malloc`. It must be used to allocate memory within a DTK application. The reason for this is that on some platforms, like Windows, the memory that is allocated within a shared library *must* be freed by the same shared library. If this is not done, then memory errors begin to occur. This function is thus provided as a convenience to ensure that all the memory is allocated within the context of the same shared library.

Prototype

```
void* PcmsEvntMalloc (int size);
```

Parameters

`size`	is the size of the memory to allocate.

PcmsEvntRealloc – Re-Allocate Memory

Purpose

This function is a wrapper to the C function `realloc()`. It must be used to re-allocate memory within a DTK application. The reason for this is that on some platforms, like Windows, the memory that is allocated within a shared library *must* be re-allocated by the same shared library. If this is not done, then memory errors begin to occur. This function is thus provided as a convenience to ensure that all the memory is maintained within the context of the same shared library.

Prototype

```
void* PcmsEvntRealloc (void *ptr, int size);
```

Parameters

`ptr`	is a pointer to the block of memory that will be resized.
`size`	is the size of the new memory.

PcmsExecCommand - Execute Dimensions CM Command Synchronously

Purpose

This function sends a command to Dimensions CM and waits for it to complete. See the *Command-Line Reference* for information on legal syntax for command mode applications.

Prototype

```
int
PcmsExecCommand (
    int    connectId,
    char   *command,
    char   **response
);
```

Parameters

`connectId`	is the database connection identifier.
`command`	is the command to be executed.
`response`	is the address of a *char** variable that will be set to point to a dynamically allocated buffer containing the diagnostic messages generated during the execution of the command. It is the responsibility of the calling application to free this buffer when it is no longer required.

Return Codes

`PcmsExecCommand ()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on the Dimensions CM command failing and sets the *response* parameter.
`PCMS_ERROR`	on other failures and sets `PcmsErrorNo` and `PcmsErrorStr`.

Sample

See the examples on the release media.

Comments

All commands that have been queued previously to Dimensions CM using `PcmsSendCommand()` will be processed first. When the results from all those commands have been received, the current command will be executed by Dimensions CM. Therefore the time taken for `PcmsExecCommand()` to process a single simple command may depend on the number of commands previously queued to Dimensions CM using `PcmsSendCommand()`.

Related Function

`PcmsSendCommand()`.

PcmsFullQuery - Find Dimensions CM Objects, Returning Complete Objects

Purpose

This function, like PcmsQuery(), will return a set of Dimensions CM objects based on a user-specified filter. However, unlike PcmsQuery(), this function returns fully populated PcmsObjStructs with both object and attribute details loaded. This function is faster than PcmsQuery() for returning large amounts of data.

Prototype

```
int
PcmsFullQuery (
    int             connectId,
    PcmsObjStruct  *ptrPcmsObjStruct,
    int             options,
    int             *noObjs,
    PcmsObjStruct  **ptrObjs
);
```

Parameters

connectId	is the database connection identifier.
ptrPcmsObjStruct	is a pointer to a PcmsObjStruct that contains the fields to query for. The objType field in the PcmsObjStruct can currently only be PCMS_CHDOC.
options	if this is set to PCMS_OPT_SECONDARY_CATALOGUE, then the function will process requests in the secondary catalog. By default this function processes requests in the primary catalog.
noObjs	is a pointer to an PcmsObjStruct variable in which to store the objects returned in ptrObjs.
ptrObjs	is a pointer to a contiguous block of allocated memory that contains the structures that the query returned. If no objects are found noObjs is set to zero and ptrObjs is set to (PcmsObjStruct *) zero. It is the responsibility of the calling application to free this memory when it is no longer required.

Return Codes

PcmsFullQuery() returns:

PCMS_OK	on success.
PCMS_FAIL	when no objects were found.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Sample

```
/*
     *-------------------------------------------------------------
     * FUNCTION SPECIFICATION
     * Name:
     *  CountPcmsChdocs
     * Description:
     *  Count chdocs
     * Return:
     * Return no of chdocs
     *-------------------------------------------------------------
     */
    static int
    CountPcmsChdocs(int conId, char *chdoc)
    {
        /*
         * Query the database for chdocs with the chdoc passed in...
         */
    int *uids = 0;
    int noUids = 0;
    int noAttrs = 5;
    int i = 0;
    int xx = PCMS_OK;
    PcmsObjStruct obj = { 0 };
    PcmsObjStruct *ptrObjs = 0;
    #define SET_ATTR(_attr,_value,p) \
    {\
            register int x = p;\
            x--;\
            obj.attrs[x].attr = _attr;\
            PcmsSvaSetVal(obj.attrs[x].value,_value,0);\
            p++;\
    }
    obj.objType = PCMS_CHDOC;
    obj.noAttrs = noAttrs;
    obj.attrs = 0;
    obj.attrs =
            (PcmsObjAttrStruct *)
             PcmsEvntCalloc(sizeof(PcmsObjAttrStruct)
            * obj.noAttrs);
    (void)strcpy(obj.objId,chdoc);

    i=1;
```

```
    /* Search for chdocs with fixed attributes */
    SET_ATTR(PCMS_ATTR_CREATE_DATE,"%%",i);
    SET_ATTR(PCMS_ATTR_ORIGINATOR,"%%",i);
    SET_ATTR(PCMS_ATTR_PHASE,"%%",i);
    SET_ATTR(PCMS_ATTR_SUPER_TYPE,"%%",i);
    SET_ATTR(PCMS_ATTR_UPDATE_DATE,"%%",i);
    if ((xx = PcmsFullQuery(conId,&obj,0, &noUids, &ptrObjs))==PCMS_OK)
    {
            /* Free memory */
            if (ptrObjs && noUids > 0)
            {
                 int xc = 0;
                 for(xc=0;xc<noUids;xc++)
                     PcmsObjFree(&ptrObjs[xc]);
            }
    }
    else
    {
             (void)fprintf(stdout,"\nNo objects found - %s",
                    (xx == PCMS_ERROR) ? "Error" : "Fail");
             if (xx == PCMS_ERROR)
                 (void)fprintf(stdout,PcmsErrorStr);
    }
    /* Free memory */
    void)PcmsObjFree(&obj);
    return ((xx == PCMS_ERROR) ? xx : noUids);
}
```

Comments

By manipulating the `*attrs` pointer and associated `noAttrs` members of the `PcmsObjStruct` structure it is possible to use system and user attributes as additional components within the query. If you wish to make use of this functionality, then you only need to specify values in the `attr` and `value` members of the `PcmsObjAttrStruct` structure. All other member fields are ignored.

Both the members of the `PcmsObjStruct` structure and the `value` field member of the `PcmsObjAttrStruct` support the use of wildcard characters. There are two different kinds of wildcard that you can use:

- `' % '` (per cent) that allows pattern matching on many characters
- `' _ '` (underscore) that allows pattern matching against one character.

This function currently supports only objects of `PCMS_CHDOC`.

You can only use the following as system attribute filters:

- `PCMS_ATTR_CREATE_DATE`
- `PCMS_ATTR_ATTR_ORIGINATOR`
- `PCMS_ATTR_PHASE`
- `PCMS_ATTR_SUPER_TYPE`
- `PCMS_ATTR_UPDATE_DATE`
- `PCMS_ATTR_LIFECYCLE`.

If you use a multi-valued attribute as a filter, then only the first element of the attribute list will be used, the other elements will be ignored.

Related Functions

`PcmsQuery()`.

PcmsGetAttachments - Obtain Request Attachment Structures

Purpose

This function allows users to retrieve the list of attachments for a specified request object.

Each of the structures that are returned detail an individual attachment for that request. The "userFile" member of each returned structures is NULL.

This function supports only uids taken from PCMS_CHDOC objects.

Prototype

```
int
PcmsGetAttachments (
   int              connectId,
   PcmsObjStruct    object,
   int              *noAttachments,
   PcmsChdAttachmentStruct  **ptrAttachments
);
```

Parameters

connectId	is the database connection identifier.
object	is the request - PcmsObjStruct that has been initialized through PcmsInitSpec() or PcmsInitUid().
noAttachments	is the address of an integer variable to contain the number of attachment structures returned.
ptrAttachments	is a pointer to a contiguous block of allocated memory that lists the structures of type PcmsChdAttachmentStruct. If no attachments are found noAttachments is set to zero and ptrAttachments is set to (PcmsChdAttachmentStruct *)zero. It is the responsibility of the called application to free this pointer when it is no longer required.

Return Codes

PcmsGetAttachments() returns:

PCMS_OK	on success.
PCMS_ERROR	on failure or when no attachments are found.

PcmsGetAttrDefNum - Get Attribute Definition Number

Purpose

This function returns the attribute number for a specified attribute definition.

Prototype

```
int
PcmsGetAttrDefNum (
    int     connectId,
    char    *productId,
    int     objType,
    char    *attrName,
    int     *attrNum
);
```

Parameters

`connectId`	is the database connection identifier.
`productId`	is the product name.
`objtype`	is the type of the object. You can use `PCMS_PART`, `PCMS_ITEM`, `PCMS_BASELINE`, `PCMS_USER`, `PCMS_CHDOC`, or `PCMS_WORKSET`.
`attrName`	is the name of the attribute (the *variable* field in the `PcmsObjAttrDefStruct`).
`attrNum`	is the returned attribute number.

Return Codes

`PcmsGetAttrDefNum()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find the specified attribute name for the given product and object type.
`PCMS_ERROR`	on failure and sets *PcmsErrorNo* and `PcmsErrorStr`.

PcmsGetAttrFile - Get Request Descriptions

Purpose

This function enables you to obtain either the detailed description, the current action description, or full action for a specified Dimensions CM request.

Prototype

```
int
PcmsGetAttrFile (
    int             connectId,
    PcmsObjStruct  *objPtr,
    int             options,
    int             attrNo,
    char            **toFile,
    int             *size
);
```

Parameters

`connectId` is the database connection identifier.

`objPtr` is the pointer to the request that the function will use.

`options` Options 0 to 2 and 4 are used as follows:

- 0 Copies the selected description into a file specified by the variable `toFile`
- 1 Copies the selected description into the variable `toFile` and populates the size of the description file into the variable `size`.

  NOTE that if `toFile` is a `NULL` pointer, then this function will allocate memory to hold the description. It is then the caller's responsibility to free this memory.
- 2 Places only the size of the description into the variable `size`.
- 4 Do *not* convert the file to the local character set of the client.

`attrNo` specifies the type of description requested, the following values are valid.

`PCMS_ATTR_ACTION_DESC`	Returns the full action description of the request.
`PCMS_ATTR_THIS_ACTION_DESC`	Returns the current action description for the request.
`PCMS_ATTR_DETAIL_DESC`	Returns the detailed description of the request.

`toFile` is a pointer to where the description is to be copied. If `*toFile` is a `NULL` pointer, this function will allocate memory to hold the description. It is the responsibility of the caller to free this memory after use.

`size` is a pointer to an integer into which the description size is returned.

Return Codes

`PcmsGetAttrFile()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

PcmsGetAttrs - Get Dimensions CM Object Attributes

Purpose

This function populates a specific PcmsObjStruct with the attribute details for that object. Calling this function will result in the noAttrs and *attrs member elements being populated with the full attribute details and attribute definitions. If you wish to access the information on the attribute definitions, use the PcmsObjAttrDefStruct pointer (attrDef) from the PcmsObjAttrStruct (attrs) pointer.

Prototype

```
int
PcmsGetAttrs (
    int             connectId,
    PcmsObjStruct  *ptrPcmsObjStruct
);
```

Parameters

connectId	is the database connection identifier.
ptrPcmsObjStruct	is a pointer to a structure of type *PcmsObjStruct* into which the attribute information will be populated.

Return Codes

PcmsGetAttrs() returns:

PCMS_OK	on success.
PCMS_FAIL	on not finding the object.
PCMS_ERROR	on failure and sets PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Comments

Before calling this function ensure that you have populated a valid PcmsObjStruct through the PcmsInitUid() or PcmsInitSpec() functions.

When you have populated an object structure using this function, remember to free the memory associated with it through PcmsObjFree() when that object is no longer required.

Related Functions

PcmsInitUid(), PcmsInitSpec(), PcmsObjFree()

PcmsGetBaseDbOptions - Retrieve Base Database Options

Purpose

This function populates a specific PcmsObjStruct with the details of the defined base database options. Calling this function will result in the noAttrs and *attrs member elements being populated with the full option details and option definitions. If you wish to access the information on the database option definitions, use the PcmsObjAttrDefStruct pointer (attrDef) from the PcmsObjAttrStruct (attrs) pointer.

Prototype

```
int
PcmsGetBaseDbOptions(
            int connectId,
            PcmsObjStruct *ptrPcmsObjStruct
);
```

Parameters

connectId	is the database connection identifier.
ptrPcmsObjStruct	is a pointer to a structure of type PcmsObjStruct into which the base database options will be populated.

Return Codes

PcmsGetBaseDbOptions() returns:

PCMS_OK	on success.
PCMS_ERROR	on failure and sets PcmsErrorStr.

Comments

This function requires that the objType member element of the supplied PcmsObjStruct pointer be initialized to PCMS_BASEDB_OPTIONS.

PcmsInitUid() and PcmsInitSpec() should *not* be used to populate the supplied PcmsObjStruct.

When you have populated an object structure using this function, remember to free the memory associated with it via PcmsObjFree() when that object is no longer required.

Related Functions

PcmsObjFree()

PcmsGetCandidates - Retrieve Candidates for Delegation

Purpose

This function returns a list of those users who are valid candidates for the object, role, and capability supplied. Currently this function supports only objects of the type PCMS_CHDOC.

Prototype

```
int
PcmsGetCandidates (
    int            connectId,
    int            options,
    PcmsObjStruct  *objPtr,
    char           *role,
    char           capability,
    char           *noCandidates,
    char           ***candidates
);
```

Parameters

connectId	is the database connection identifier.
options	is reserved for future use.
objPtr	is pointer to a PcmsObjStruct that contains an object that has been initialized with PcmsInitSpec() or PcmsInitUid().
role	is the role for which to retrieve candidates, for example, DEVELOPER.
capability	is the capability that the candidate has – 'L' (Leader),'P' (Primary) or 'S' (Secondary).
noCandidates	is a pointer to an integer in which to store the number of returned values.
candidates	is the address of a pointer in which the returned list of users will be returned. The function will allocate the associated memory. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

PcmsGetCandidates() returns:

PCMS_OK	on success.
PCMS_FAIL	on failure to find any data.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

PcmsGetCommandLine – Get the Dimensions CM Command

Purpose

This function will return a constant string pointer to a copy of the command that was submitted to Dimensions CM. This is intended to allow you, from within a DTK event, to determine what Dimensions CM command was actually run.

Prototype

```
const char* PcmsGetCommandLine (void);
```

Parameters

None.

PcmsGetConnectDesc - Get Input File Descriptor

Purpose

This function returns the input file descriptor for the specified connection identifier. The purpose of this function is for an X based application to add the file descriptor as an input for the X application using `XtAddInput()`. When the X application receives notification that there is input available on the file descriptor, the application should call `PcmsCheckMessages()`. This will then activate any callback functions that have been setup by `PcmsSetCallback()`. Using this method of processing Dimensions CM messages, the need for `PcmsSetIdleChecker()` is eliminated.

Prototype

```
int
PcmsGetConnectDesc (
    int    connectId
);
```

Parameters

`connectId`	is the database connection identifier.

Return Codes

`PcmsGetConnectDesc ()` returns:

`int fd`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrno` and `PcmsErrorStr`.

Comments

This function is applicable only to UNIX.

Related Functions

`PcmsExecCommand()`,`PcmsCheckMessages()`, `PcmsSetCallback()`.

PcmsGetDimensionsVersions - Obtain List of Loaded DLLs and Dates and Times

Purpose

To provide a means of determining exactly which DLLs have been loaded to support this DTK program.

Prototype

```
int
PcmsGetDimensionsVersions(
   _TCHAR **ppszResult
   )
```

Parameters

Pointer to a string pointer that will return a single string of text describing the DLLs loaded. This string should be copied and freed using the usual mechanisms for freeing resources allocated by the DTK.

Return Codes

`PcmsGetDimensionsVersions()` returns NULL if the call fails.

Comments

None.

Related Functions

None.

PcmsGetPendingUsers - Obtain Inbox User Structures

Purpose

PRIVILEGES Manage and View Other Users' Lists.

This function allows users with the above privilege to retrieve the list of inbox users (with their roles and capabilities) for a specified request object.

Each of the structures that are returned detail the next status and phase possible for a user on that request.

This function supports only objects of `PCMS_CHDOC` type.

Prototype

```
int
PcmsGetPendingUsers (
    int                     connectId,
    int                     options,
    int                     reserved,
    PcmsObjStruct          *objPtr,
    int                     *noPendingUsers,
    PcmsPendingUserStruct  **ptrPendingUsers
);
```

Parameters

`connectId`	is the database connection identifier.
`options`	is reserved for future use.
`reserved`	is reserved for future use.
`objPtr`	is a pointer to a request - `PcmsObjStruct` that has been initialized through `PcmsInitSpec()` or `PcmsInitUid()`.
`noPendingUsers`	is the address of an integer variable to contain the number of `connectId` structures returned.
`ptrPendingUsers`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsPendingUser`. If no objects are found `noPendingUsers` is set to zero and `ptrPendingUsers` is set to `(PcmsPendingUser *) zero`. It is the responsibility of the caller application to free this pointer when it is no longer required.

Return Codes

`PcmsGetPendingUsers()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Related Functions

`PcmsInitSpec()`, `PcmsInitUid()`, `PcmsGetUserRoles()`.

PcmsGetRSAttrs - Retrieve Attribute Numbers in a Role Section

Purpose

This function returns the attribute numbers that are used by a given role section object/type uid combination. The order in at these attribute numbers are returned is in the display order described in the process model. If you specify an object uid, then only those roles section attributes applicable to that object are returned. If you specify a type uid, then all the role section attributes associated with that type's lifecycle are returned.

Object and type uids are mutually exclusive. If you specify object uid (`objUid`), this will cause the function to ignore any type uids that you may additionally specify.

This function currently supports items, requests, baselines, parts, and projects.

Prototype

```
int
PcmsGetRSAttrs (
    int    connectId,
    int    reserved,
    int    options,
    int    objUid,
    int    typeUid,
    char   *roleName,
    char   *userName,
    int    *noAttrs,
    int    **attrs,
    char   **defaultRole
);
```

Parameters

`connectId`	is the database connection identifier.
`reserved`	is reserved for future use.
`options`	is reserved for future use.
`objUid`	is the uid of the object that the function will use.
`typeUid`	is the uid of the type name that the function will use.
`roleName`	is the role name to request the attribute numbers for. If a `NULL` string is used, the default role section name will be calculated and returned through the parameter `defaultRole`.
`userName`	is reserved for future use.
`noAttrs`	is the total number of attribute numbers returned.

`attrs`	is a pointer to the list of attribute numbers contained in this role section. It is the responsibility of the calling application to free this pointer when it is no longer required.
`defaultRole`	is the address of a `char *` that is used to store the default role section name when the `roleName` parameter is a `NULL` string. It is the caller's responsibility to free the memory if this string is not `NULL`.

Return Codes

`PcmsGetRSAttrs()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

You can use the `roleName` parameter to filter on those attributes that apply to a certain role section name. In addition, two special filters can also be used.

1	`$ALL`	returns all the attributes that are associated with that object or type uid's lifecycle.
2	`$ALL_ROLE_SECTIONS`	returns all the attributes used by any role sections for this object or type uid's lifecycle.

If you specify the `roleName` parameter as a `NULL` string, the function will calculate the default role section name and populate this into the `defaultRole` parameter.

Related Functions

`PcmsGetRSNames()`

PcmsGetRSNames - Obtain Role Section Names for a Product

Purpose

This function retrieves the list of role section names corresponding to a given object uid. This uid can be for a request, an item, a part, a baseline, a workset, or a type name. If you specify an object uid, then only those role sections applicable to that object are returned. If you specify a type uid, then all role sections associated with that type's lifecycle are returned.

Prototype

```
int
PcmsGetRSNames (
    int    connectId,
    int    reserved,
    int    options,
    int    uid,
    char   **message,
    int    *noValues,
    char   **values
);
```

Parameters

`connectId`	is the database connection identifier.
`reserved`	is reserved for future use.
`options`	is reserved for future use.
`uid`	is the object uid or type uid that will be used.
`message`	is the error message returned if the status is not `PCMS_OK`. It is the caller's responsibility to free this allocated memory if not `NULL`.
`noValues`	is the address of an integer corresponding to the number of role section names returned.
`values`	is a `char *` array of role section names returned. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsGetRSNames()` returns:

`PCMS_OK`	on success.

`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Related Functions

`PcmsGetRSAttrs()`.

PcmsGetStringSymbolValue - Obtain Values Associated with a Symbol

Purpose

Obtain a value for a symbol from the symbol table.

Prototype

```
int
PcmsGetStringSymbolValue(
   int Pcms_connection,
     const _TCHAR *pszTable,
     const _TCHAR *pszSymbol,
     int iIndex,
     _TCHAR **ppszResult)
```

Parameters

Pcms_connection	This is the integer identifying the connection the command was sent on.
pszTable	A string that defines the tables to search. `P` denotes search the persistent table, and `T` the temporary table. The order is observed, and the first match returned.
pszSymbol	The symbol to search for.
iIndex	The occurrence of the symbol for which a value is to be returned.
ppszResult	The string value requested, or `NULL` if no match. This string is a copy of that from the symbol table, and must be freed using the usual DTK mechanism once consumed.

Return Codes

`PcmsGetStringSymbolValue()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to locate a symbol of the specified name.
`PCMS_ERROR`	on failure.

Comments

None.

Related Functions

`PcmsGetSymbolInfo()`

PcmsGetSymbolInfo - Look Up Symbol in Structured Information Return Table

Purpose

To provide a means of checking for a symbol in the Structured Information Return (SIR) symbol tables.

Prototype

```
int
PcmsGetSymbolInfo(
     int Pcms_connection,
     const _TCHAR *pszTable,
     const _TCHAR *pszSymbol,
     int *pcMaxOccurs,
     int *pType)
```

Parameters

Pcms_connection	This is the integer identifying the connection the command was sent on.
pszTable	A string that defines the tables to search. `P` denotes search the persistent table, and `T` the temporary table. The order is observed, and the first match returned.
pszSymbol	The symbol to search for.
pcMaxOccurs	A pointer to the integer to hold the maximum number of times this field occurs. `NULL` is valid, and will result in this parameter not being returned.

ptype	A pointer to the type of this field. `NULL` is valid, and will result in this parameter not being returned.
	
	Values returned are drawn from:
	
	`#define PCMS_STYPE_CHAR 0`
	`#define PCMS_STYPE_INT 1`
	`#define PCMS_STYPE_ST 2`
	`#define PCMS_STYPE_PVOID 3`
	
	These values are defined in `pcms_api.h`.

Return Codes

`PcmsGetSymbolInfo()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to locate a symbol of the specified name.
`PCMS_ERROR`	on failure.

Comments

None.

Related Functions

`PcmsGetStringsSymbolValue()`

PcmsGetUserRelTypes - Obtain User Relationship Subtypes

Purpose

This function returns all the user-relationship sub-types for a specified product. These relationship types include affected, information, dependent, and user-defined item-to-item relationships.

Prototype

```
int
PcmsGetUserRelTypes (
    int                 connectId,
    int                 reserved,
    int                 options,
    char                *productId,
    int                 *noRelTypes,
    PcmsRelTypeStruct  **relTypes
);
```

Parameters

`connectId`	is the database connection identifier.
`reserved`	is reserved for future use.
`options`	is reserved for future use.
`productId`	is the product Id that the function will use.
`noRelTypes`	is pointer to an integer variable in which to store in `relTypes`.
`relTypes`	is a pointer to a contiguous block of allocated memory that lists the an array of `PcmsRelTypeStructs` returned. If no objects are found as a result of the function call, then `noRelTypes` is set to zero and `relTypes` is set to `(PcmsRelTypeStruct *) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsGetUserRelTypes()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

For completeness, definitions of the standard Dimensions CM defined relationships are also given.

PcmsGetUserRoles - Obtain User Role Structures

Purpose

This function returns those users who are found to have certain roles for this object. The returned structures indicate whether or not the user was delegated the role, or if the role was inherited from the design tree. These structures will also indicate if the user's capability is primary or secondary.

The actionable field of the returned `PcmsUserRoleStruct` is not populated.

Prototype

```
int
PcmsGetUserRoles (
    int                 connectId,
    int                 reserved,
    int                 options,
    PcmsObjStruct       *objPtr,
    int                 partUid,
    int                 noRoles,
    char                *roles[],
    char                *userName,
    int                 *noUserRoles,
    PcmsUserRoleStruct  **ptrUserRoles
);
```

Parameters

`connectId`	is the database connection identifier.
`reserved`	is reserved for future use.
`options`	is reserved for future use.
`objPtr`	is a pointer to `PcmsObjStruct` that has been initialized through `PcmsInitSpec()` or `PcmsInitUid()`.
`partUid`	is the `partUid` for which to obtain roles.
`noRoles`	is the optional number of roles on which you wish to filter.
`roles`	is an optional array of `char *` roles that are used to filter the data returned.
`userName`	is an optional username.
`noUserRoles`	is the address of an integer variable to contain the number of `PcmsUserRole` structures returned.
`ptrUserRoles`	is a pointer to a contiguous block of allocated memory that lists the structures of type `PcmsUserRole`. If no objects are found `noUserRoles` is set to zero and `ptrUserRoles` is set to `(PcmsUserRoleStruct *) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsGetUserRoles()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure to find any data.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

If `partUid` is zero, then the appropriate parent part is calculated and used by the function for the tree walk.

You can use the `userName` parameter to act as a filter for the user against which this function will apply. If you specify a `NULL` value, then all users are retrieved.

If you know which particular roles that interest you, you can use the `noRoles` and `roles` parameters to filter for these roles. If `noRoles` is zero (0), all the roles will be retrieved.

Related Functions

`PcmsGetPendingUsers()`.

PcmsGetWsetObj - Get User's Current Project

Purpose

This function returns the current project in which this Dimensions CM session is active.

Prototype

```
int
PcmsGetWsetObj (
    int             connectId,
    int             options,
    PcmsObjStruct  **ptrPcmsObjStruct);
```

Parameters

connectId	is the database connection identifier.
options	is not currently supported (use 0).
ptrPcmsObjStruct	is a dynamically allocated buffer containing a PcmsObjStruct, whose objType field will be PCMS_WORKSET.

Return Codes

PcmsGetWsetObj() returns:

PCMS_OK	on success.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Sample

```
static char*
GetWorkset(int conId)
{
    static char workset_id[PCMS_L_PRODUCT_ID + PCMS_L_CD_ID + 5];
    PcmsObjStruct   *wsobj = (PcmsObjStruct *)0;
    workset_id[0] = '\0';
    switch(PcmsGetWsetObj(conId, 0, &wsobj))
   {
       case    PCMS ERROR:
       case    PCMS FAIL:
                    return((char *)0);
       default:
                    break;
   }

   (void)sprintf(workset_id,"\"%s\":\ "%s\"",
       wsobj->productId,wsobj->objId);
   return(workset_id);
}
```

PcmsInitSpec - Get Dimensions CM Object Details by Specification

Purpose

This function populates a PcmsObjStruct with the details on a specific object.

Prototype

```
int
PcmsInitSpec (
    int             connectId,
    char            *objSpec,
    int             objType,
    PcmsObjStruct  *ptrPcmsObjStruct
);
```

Parameters

connectId	is the database connection identifier.
objSpec	is the textual specification of an object, for example, "FS:HITOMI_C.A-SRC;main#1".
objType	is the type of the object that the specification refers to. You can use PCMS_PART, PCMS_ITEM, PCMS_BASELINE, PCMS_USER, PCMS_CHDOC, PCMS_WORKSET, or PCMS_DIRECTORY.
ptrPcmsObjStruct	is a pointer to a structure of type PcmsObjStruct in which to store the details on the object specified by objSpec.

Return Codes

PcmsInitSpec() returns:

PCMS_OK	on success.
PCMS_FAIL	on not finding the object.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Comments

This function does not populate the attribute details within the PcmsObjStruct structure. This has to be done separately by calling PcmsGetAttrs().

Related Functions

PcmsInitUid(), PcmsGetAttrs().

PcmsInitUid - Get Dimensions CM Object Details by Uid

Purpose

This function populates a PcmsObjStruct with the details on a specific object.

Prototype

```
int
PcmsInitUid (
    int             connectId,
    int             objUid,
    int             objType,
    PcmsObjStruct  *ptrPcmsObjStruct
);
```

Parameters

connectId	is the database connection identifier.
objUid	is the integer uid of the object.
objType	is the type of the object that the specification refers to. You can use PCMS_PART, PCMS_ITEM, PCMS_BASELINE, PCMS_USER, PCMS_CHDOC, PCMS_WORKSET, or PCMS_DIRECTORY.
ptrPcmsObjStruct	is a pointer to a structure of type PcmsObjStruct in which to store the details on the object specified by the uid.

Return Codes

PcmsInitUid() returns:

PCMS_OK	on success.
PCMS_FAIL	on not finding the object.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Comments

This function does not populate the attribute details within the PcmsObjStruct structure. This has to be done separately by calling PcmsGetAttrs().

Related Functions

PcmsInitSpec(), PcmsGetAttrs().

PcmsLovFree - Free a List of Values

Purpose

This function frees an array of strings returned by `PcmsAttrGetLov()`.

Prototype

```
int
PcmsLovFree (
    int    noValues,
    char   **values
);
```

Parameters

`noValues`	is the number of strings in the array.
`values`	is the array name.

Return Codes

`PcmsLovFree()` returns:

`PCMS_OK`	on success.

Related Functions

`PcmsAttrGetLov()`.

PcmsObjFree - Free Dimensions CM Object Structures

Purpose

This function frees any memory that may have been allocated internally to the `PcmsObjStruct` structure. This includes any attributes or attribute definition structures.

Prototype

```
int
PcmsObjFree (
    PcmsObjStruct  *ptrPcmsObjStruct
);
```

Parameters

`ptrPcmsObjStruct`	is a pointer to a structure of type `PcmsObjStruct` from which to free the memory.

Return Codes

`PcmsObjFree()` returns:

`PCMS_OK`	this value is always returned.

PcmsObjGetBackRels - Get Dimensions CM Object Reverse Relationships

Purpose

This function can be used to navigate objects and their relationships to other objects. For example, to return predecessor revisions of an item. This function performs the inverse navigation of `PcmsObjGetRels()`.

Prototype

```
int
PcmsObjGetBackRels (
    int    connectId,
    int    fromObjUid,
    int    objType,
    int    options,
    int    contextUid,
    int    *noRels,
    PcmsRelStruct  **ptrPcmsRelStruct);
```

Parameters

`connectId`	is the database connection identifier.
`fromObjUid`	is the integer uid for a Dimensions CM object.
`objType`	is the type of the Dimensions CM object. This type must be one of `PCMS_PART`, `PCMS_ITEM`, `PCMS_CHDOC`, `PCMS_BASELINE`, or `PCMS_DIRECTORY`.
`options`	is a collection of bits set that indicates the type of objects to return in `ptrPcmsRelStruct`. If this value is zero, then all object relationship types are returned. You can restrict the list of objects returned by specifying one or more of the types `PCMS_PART`, `PCMS_ITEM`, `PCMS_CHDOC`, `PCMS_BASELINE`, or `PCMS_DIRECTORY`. **NOTE** For directories, you *must* specify PCMS_DIRECTORY.
`contextUid`	can be used to limit the objects navigated to a specific `baseline_uid`.
`noRels`	is a pointer to an integer variable in which to store the number of structures returned in `ptrPcmsRelStruct`.
`ptrPcmsRelStruct`	is a pointer to a contiguous block of allocated memory that contains a number of structures of type `PcmsRelStruct`. If no objects are found then *noRels* is set to zero and `ptrPcmsRelStruct` is set to `(PcmsRelStruct *) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsObjGetBackRels()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	when no objects where found.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

If `PCMS_OPT_PRED` is set, then `contextUid` will be ignored and the `fromObjUid`'s predecessor revision will be returned. This option is only valid when `objType` is set to either `PCMS_PART` or `PCMS_ITEM`.

The value of `fromObjUid` cannot be specified as zero.

The option `PCMS_OPT_MERGED` can be used in conjunction with `PCMS_OPT_PRED` to return objects that have been used in a merge to create the object specified in `fromObjUid`. Currently this option is only available for items.

The following table lists combinations of object reference, object type and query options that are valid for this function. Note that the PCMS_BASELINE option is always invalid when obtaining the reverse relationships recorded in a baseline.

	Options			
objType	PART	ITEM	B'LINE	CHDOC
PART	V	I	V	V
ITEM	V	V	V	V
BASELINE	I	I	I	V
CHDOC	I	I	V	V

Key: V = Valid I = Invalid

Related Functions

`PcmsObjGetRels()`.

PcmsObjGetRels - Get Dimensions CM Object Relationships

Purpose

This function can be used to navigate objects and their relationships to other objects. For example, to return successor revisions of an item.

Prototype

```
int
PcmsObjGetRels (
    int    connectId,
    int    fromObjUid,
    int    objType,
    int    options,
    int    contextUid,
    int    *noRels,
    PcmsRelStruct  **ptrPcmsRelStruct);
```

Parameters

`connectId`	is the database connection identifier.
`fromObjUid`	is the integer uid for a Dimensions CM object.
`objType`	is the type of the Dimensions CM object. This type must be one of `PCMS_PART`, `PCMS_ITEM`, `PCMS_CHDOC`, `PCMS_BASELINE`, or `PCMS_DIRECTORY`.
`options`	is a collection of bits set that indicates the type of objects to return in `ptrPcmsRelStruct`. If this value is zero, then all object relationship types are returned. You can restrict the list of objects returned by specifying one or more of the following types `PCMS_PART`, `PCMS_ITEM`, `PCMS_CHDOC`, `PCMS_BASELINE`, or `PCMS_DIRECTORY`. **NOTE** For directories, you *must* specify PCMS_DIRECTORY.
`contextUid`	can be used to limit the objects navigated to a specific `baseline_uid`.
`noRels`	is a pointer to an integer variable in which to store the number of structures returned in `ptrPcmsRelStruct`.
`ptrPcmsRelStruct`	is a pointer to a contiguous block of allocated memory that contains a number of structures of type `PcmsRelStruct`. If no objects are found then `noRels` is set to zero and `ptrPcmsRelStruct` is set to `(PcmsRelStruct *) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

PcmsObjGetRels() returns:

PCMS_OK	on success.
PCMS_FAIL	when no objects were found.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Comments

- If fromObjUid is zero, then the top of the Dimensions CM database is used as the starting point for the navigation.
- If fromObjUid is zero and options is set to PCMS_PART, then only the Dimensions CM products will be returned.
- If PCMS_OPT_LATEST is set, then only the latest version of any related objects is returned. This option is only valid when objType is set to either PCMS_ITEM or PCMS_PART, and only applies to 'item to design part' relationships.
- If PCMS_OPT_SUCC is set, then contextUid will be ignored and the fromObjUid's successor revision will be returned. This option is only valid when objType is set to either PCMS_PART or PCMS_ITEM.
- The option PCMS_OPT_MERGED can be used in conjunction with PCMS_OPT_SUCC to return objects that have resulted as a merge based on the object specified in fromObjUid. Currently this option is only available for items.
- The following table lists the combinations of object references, object types and query options that are valid for this function. Note that the PCMS_BASELINE option is always invalid when obtaining the relationships recorded in a baseline.

		Options			
fromObjUid	objType	PART	ITEM	BASELINE	CHDOC
Zero	Ignored	V	V	V	V
Non-zero	PART	V	V	I	I
Non-zero	ITEM	I	V	I	I
Non-zero	BASELINE	V	V	I	V
Non-zero	CHDOC	V	V	V	V

Key: V = Valid I = Invalid

Related Functions

PcmsGetBackRels().

PcmsObjInSecondary - Is Request Object in Secondary Catalog

Purpose

This macro will return an integer indicating whether the object specified by the parameter `objPtr` is a request in the secondary catalog.

Prototype

```
int
    PcmsObjInSecondary (
    PcmsObjStruct  *objPtr
    );
```

Parameters

`objPtr`	is a pointer to a `PcmsObjStruct`.

Return Codes

`PcmsObjInSecondary()` returns:

`PCMS_OK`	if this request is in the secondary catalog.
`PCMS_FAIL`	if this request is in the primary catalog.

PcmsPendGet - Retrieve Dimensions CM Object Inboxes for a User

Purpose

This function retrieves the inbox of items and/or requests for the current or a specified user.

CAUTION! The parameter `userName` will turn into an object `uid` for a Dimensions CM user object in the future.

Prototype

```
int
PcmsPendGet (
    int             connectId,
    char            *userName,
    char            *reserved,
    int             options,
    int             *noStructs,
    PcmsPendStruct  **ptrPcmsPendStructs
);
```

Parameters

`connectId`	is the database connection identifier.
`userName`	is the user to query the objects inbox for. If this parameter is `NULL` then the current user's inbox is queried.
`reserved`	is reserved for future use.
`options`	this determines which objects are to be returned by this function. You can specify one of the following: ■ Zero (0) that returns all inbox object types ■ `PCMS_CHDOC` that returns all inbox requests ■ `PCMS_ITEM` that returns all inbox items.
`noStructs`	is a pointer to an integer variable in which to store the number of structures of type `PcmsPendStruct` returned in `ptrPcmsPendStruct`.
`ptrPcmsPendStructs`	is a pointer to a contiguous block of allocated memory that lists the objects that the function returned. If no objects are found, then this value is set to 0 and `ptrPcmsPendStructs` is set to `(PcmsPendStruct*) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsPendGet()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	when no objects are found.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

PRIVILEGES Manage and View Other Users' Lists.

This function allows users with the above privilege to use the `userName` parameter to query another user's item or request inbox.

PcmsPendWhoGet - Retrieve Users for Object

Purpose

This function retrieves the users who will have a specified object in their inbox at a user-defined status.

Prototype

```
int
PcmsPendWhoGet (
    int             connectId,
    int             objUid,
    int             objType,
    char            *status,
    int             options,
    int             *noStructs,
    PcmsPendStruct  **ptrPcmsPendStructs
);
```

Parameters

`connectId`	is the database connection identifier.
`objUid`	is the uid for a Dimensions CM object against which this function will be run. **NOTE** `objUid` cannot be zero (0).
`objType`	is the type of the Dimensions CM object. This type must be one of `PCMS_PART`, `PCMS_BASELINE`, `PCMS_ITEM`, or `PCMS_CHDOC`.
`status`	is the status in the lifecycle for which you wish to return the list of users.
`options`	is reserved for future use.
`noStructs`	is a pointer to an integer variable in which to store the number of structures of type `PcmsPendStruct` returned in `ptrPcmsPendStructs`.
`ptrPcmsPendStructs`	is a pointer to a contiguous block of allocated memory that lists the users that the function has found. If no objects are found `noStructs` is set to zero and `ptrPcmsPendStructs` is set to `(PcmsPendStructs *) zero`. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsPendWhoGet()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	when no objects were found.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Related Functions

`PcmsPendGet()`.

Comments

PRIVILEGES Manage and View Other Users' Lists.

This function allows users with the above privilege to use the `userName` parameter to query another user's item or request inbox.

PcmsPopulate - Populate an Object's Attributes Values

Purpose

This function populates a given PcmsObjStruct with attributes, attribute definitions and values. The values are copied from an existing object that you supply, and are merged with the attributes generic to a specified type uid.

Prototype

```
int
PcmsPopulate (
    int             connectId,
    int             options,
    int             objType,
    int             typeUid,
    PcmsObjStruct  *primeObj,
    PcmsObjStruct  **outObject
);
```

Parameters

connectId	is the database connection identifier.
options	is reserved for future use.
objType	is the type of the outObject, for example, PCMS_ITEM.
typeUid	is the typeUid for the outObject.
primeObj	is a pointer to another object to prime the values of the outObject from.
outObject	is the address of a pointer to a PcmsObjStruct that will be allocated and populated by this function.

Return Codes

PcmsPopulate() returns:

PCMS_OK	on success.
PCMS_FAIL	on failure to find any data.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

PcmsQuery - Find Dimensions CM Objects, Returning Uids

Purpose

This function finds a list of Dimensions CM objects from the fields specified in the PcmsObjStruct. If values are present in the fields of the ptrPcmsObjStruct, they will be used to further refine the query. The only field in ptrPcmsObjStruct that must be filled in is objType. This means that the returned object uids will all be of the same type.

Prototype

```
int
PcmsQuery (
    int             connectId,
    PcmsObjStruct  *ptrPcmsObjStruct,
    int             options,
    int             *noObjs,
    int             **ptrObjUids
);
```

Parameters

connectId	is the database connection identifier.
PtrPcmsObjStruct	is a pointer to a PcmsObjStruct that will be used to further refine the query. The objType field in the PcmsObjStruct must be one of PCMS_ITEM, PCMS_PART, PCMS_CHDOC, PCMS_USER, PCMS_BASELINE, or PCMS_WORKSET.
options	is a collection of bits that is used to change the default behavior of this function.
noObjs	is a pointer to an integer variable in which to store the number of integer uids returned in ptrObjUids.
ptrObjUids	is a pointer to a contiguous block of allocated memory that lists the uids that the query returned. If no objects are found as a result of the function call, then noObjs is set to zero and ptrObjUids is set to (int *) zero. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

PcmsQuery() returns:

PCMS_OK	on success.
PCMS_FAIL	when no objects were found.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Sample

```
 /*
     *-------------------------------------------------------------
     * FUNCTION SPECIFICATION
     * Name:
     *  CountPcmsItems
     * Description:
     *  Count items in Dimensions
     * Parameters:
     *  char *itemId
     * Return:
     *  no of items
     * Notes:
     *-------------------------------------------------------------
     */
static int
CountPcmsItems(int conId, char *itemId)
{
   /*
   * Query the database for items with the itemId passed in.
   */
   int *uids = 0;
   int noUids = 0;
   int noAttrs = 2;
   int i = 0;
   int xx = PCMS_OK;
   PcmsObjStruct obj = { 0 };
   #define SET_ATTR(_attr,_value,p) \
   {\
            register int x = p;\
            x--;\
            obj.attrs[x].attr = _attr;\
            PcmsSvaSetVal(obj.attrs[x].value,_value,0);\
            p++;\
   }
   obj.objType = PCMS_ITEM;
   obj.noAttrs = noAttrs;
   obj.attrs = 0;
   obj.attrs =
       (PcmsObjAttrStruct *)
       PcmsEvntCalloc(sizeof(PcmsObjAttrStruct)
       *obj.noAttrs);
   (void)strcpy(obj.objId,itemId);
   i=1;
    /* Search for items with TXT as format and of SPEC_UID 121 */
    SET_ATTR(PCMS_ATTR_FORMAT,"TXT",i);
    SET_ATTR(PCMS_ATTR_ITEM_SPEC_UID,"121",i);
    /* Run the query */
    if ((xx = PcmsQuery(conId,&obj,0,&noUids,&uids))==PCMS_OK)
    {
     /* Free memory */
     if (uids && noUids > 0)
         PcmsEvntFree((int *)uids);
    }
    else
```

```
    {
     (void)fprintf(stdout,"\nNo objects found - %s",
           (xx == PCMS_ERROR) ? "Error" : "Fail");
     if (xx == PCMS_ERROR)
         (void)fprintf(stdout,PcmsErrorStr);
    }
    /* Free memory */
    (void)PcmsObjFree(&obj);
    return ((xx == PCMS_ERROR) ? xx : noUids);
}
```

Comments

By manipulating the `*attrs` pointer and associated `noAttrs` members of the `PcmsObjStruct` structure it is possible to use system and user attributes as additional components within the query. If you wish to make use of this functionality, then you only need to specify values in the `attr` and `value` members of the `PcmsObjAttrStruct` structure. All other member fields are ignored.

Both the members of the `PcmsObjStruct` structure and the `value` field member of the `PcmsObjAttrStruct` structure support the use of wildcard characters. There are two different kinds of wildcard that you can use:

- `'%'` (per cent) that allows pattern matching on many characters
- `'_'` (underscore) that allows pattern matching against one character.

If the `objType` field member of the `PcmsObjStruct` is set to `PCMS_CHDOC`, you can use the `options` parameter (`PCMS_OPT_SECONDARY_CATALOGUE`) to make the function query against the secondary request catalog instead of the primary catalog. By default, the function will always query the primary request catalog.

If the `objType` field member of the `PcmsObjStruct` is set to `PCMS_ITEM`, you can use the options parameter (`PCMS_OPT_LATEST`) to return only the `latest` revisions of the items that match the query.

If you use attribute filters or the `options` parameter to further restrict the list of uids returned, the speed of the query will be affected.

Only the following system attributes are supported in this function.

Object	Attribute
PCMS_PART	PCMS_ATTR_PARTNO PCMS_ATTR_LOCALNO
PCMS_ITEM	PCMS_ATTR_FORMAT PCMS_ATTR_FILENAME PCMS_ATTR_ITEM_SPEC_UID PCMS_ATTR_LIB_FILENAME PCMS_ATTR_COMPRESSED PCMS_ATTR_SENDER_ID PCMS_ATTR_CREATE_DATE PCMS_ATTR_ORIGINATOR PCMS_ATTR_PHASE PCMS_ATTR_LIFECYCLE
PCMS_BASELINE	PCMS_ATTR_TEMPLATE PCMS_ATTR_BASELINE_TYPE

Object	Attribute
PCMS_CHDOC	PCMS_ATTR_CREATE_DATE PCMS_ATTR_ORIGINATOR PCMS_ATTR_PHASE PCMS_ATTR_SUPER_TYPE PCMS_ATTR_UPDATE_DATE PCMS_ATTR_LIFECYCLE
PCMS_USER	PCMS_ATTR_GROUP PCMS_ATTR_FULL_USERNAME PCMS_ATTR_PHONE PCMS_ATTR_DEPT PCMS_ATTR_SITE PCMS_ATTR_BASEDB PCMS_ATTR_EMAIL PCMS_ATTR_PRIVILEGE
PCMS_WORKSET	PCMS_ATTR_TRUNK (PCMS_ATTR_TRUNC) PCMS_ATTR_ENFORCE_REV

If you use a multi-valued attribute as a filter, only the first element of the attribute list will be used. The other elements will be ignored.

The `PCMS_ATTR_PRIVILEGES` system-attribute of `PCMS_USER` object is available for Serena Command Center users to determine whether or not a Dimensions CM user has been deleted or disabled. This system-attribute is populated with the current privilege (type of user access) for a Dimensions CM user. Public values of this system-attribute are as in the table below (but note that these may change in future releases of Dimensions CM).

Attribute	Attribute Value	
PCMS_ATTR_PRIVILEGE	-1	User has been disabled/dropped.
	0	User is an enabled "normal" user.

Related Functions

`PcmsFullQuery()`.

PcmsSendCommand - Execute Dimensions CM Command Asynchronously

CAUTION! This function is deprecated. It no longer submits commands asynchronously. You should, instead, use the synchronous function `PcmsExecCommand`, see page 78.

Purpose

This function sends a command to Dimensions CM and returns without waiting for it to complete. See the *Command-Line Reference* for information on legal syntax for command mode applications.

Prototype

```
int
PcmsSendCommand (
    int     connectId,
    char    *command,
    int     *cmdId
);
```

Parameters

`connectId`	is the database connection identifier.
`command`	is the command to execute.
`cmdId`	is a unique command identifier that is returned to the user.

Return Codes

`PcmsSendCommand ()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo` and `PcmsErrorStr`.

Comments

You must call the function `PcmsCheckMessages()` to check whether the results of commands submitted using `PcmsSendCommand()` are available. If results are available, your callback function will be invoked (see `PcmsSetCallback` on page 130).

NOTE If your application does not call `PcmsCheckMessages()` periodically, commands sent with `PcmsSendCommand()` may not be executed by Dimensions CM.

The commands EXIT and exit will result in an error being returned. A null string will also result in an error. Use `PcmsDisconnect()` to terminate the connection with the Dimensions CM Server.

Related Functions

`PcmsExecCommand`, `PcmsSetCallback()`, `PcmsCheckMessages()`.

PcmsSetAttrs - Set Dimensions CM Object Attributes

Purpose

This function uses the attribute details defined in a PcmsObjStruct and sets these attributes on the appropriate Dimensions CM object.

By using this function you can populate a PcmsObjStruct with the details on an object, such as an item, manipulate the noAttrs and *attrs member fields and then apply these attributes to the Dimensions CM object.

Prototype

```
int
PcmsSetAttrs (
    int             connectId,
    PcmsObjStruct  *ptrPcmsObjStruct
);
```

Parameters

connectId	is the database connection identifier.
ptrPcmsObjStruct	is a pointer to a structure of type PcmsObjStruct that has been initialized through PcmsInitUid() or PcmsInitSpec() and into which the new attributes have been defined. Each element of the array pointed to by the *attrs field must have the attr and value member fields set.

Return Codes

PcmsSetAttrs() returns:

PCMS_OK	on success.
PCMS_FAIL	on not setting the attributes successfully. PcmsErrorStr will contain the message returned from Dimensions CM.
PCMS_ERROR	on failure and sets PcmsErrorNo, PcmsErrorStr, PcmsDbErrorNo, and PcmsDbErrorStr.

Comments

IMPORTANT! If the Dimensions "electronic signatures" facility is enabled, authentication is required for sensitive changes to a request's or item's lifecycle state and attributes. Consequently, PcmsSetAttrs() cannot be used to modify attributes in such situations.

- PcmsSetAttrs() can set user-defined attributes on parts, items, and requests, but non-visible attributes (those that are not displayed in client interfaces) can be set only on requests.

- The `*attrs` pointer in the `PcmsObjStruct` must only be populated with attributes that you wish to add or modify.
- This function can be used only to setup user-defined attributes on objects of type `PCMS_ITEM`, `PCMS_CHDOC`, and `PCMS_PART`.
- This function enforces the same checks used when setting attribute values as performed by any other interface.
- This function is intended for applications using the *Client Architecture Model*. It is not supported when called from within DTK events. If you wish to change attributes from within an event, then please use the Validate event as described in Chapter 5, "Dimensions CM Events Callout Interface" on page 157.

Related Functions

`PcmsInitUid()`, `PcmsInitSpec()`, `PcmsGetAttrs()`.

PcmsSetCallback - Set Dimensions CM API Server Callback

Purpose

This function sets up a callback function for the specified *connectId*. The previous callback function and associated `clientData` are returned in the `ptrOldPcmsCallback` pointer. The callback function is used to register the results of Dimensions CM commands submitted asynchronously by the function `PcmsSendCommand()`. The callback function will be invoked when a user calls `PcmsCheckMessages()`.

Prototype

```
int
PcmsSetCallback (
    int       connectId,
    PcmsCallbackStruct  *ptrNewPcmsCallback,
    PcmsCallbackStruct  *ptrOldPcmsCallback
);

typedef struct
{
   PcmsCallbackProc callback;
   void             *clientData;
} PcmsCallbackStruct;
```

Parameters

`connectId`	is the database connection identifier.

`ptrNewPcmsCallback`	is a pointer to a structure of type `PcmsCallbackStruct`. In this structure the member field `callback` is a pointer to the callback function. This function must have the following prototype.
	`void sampleCallbackProc(`
	`int connectId,`
	`void *clientData,`
	`int commandStatus,`
	`int commandId,`
	`char *commandStr,`
	`char *callData,`
	`...`
	`)`
	The `clientData` field of the `PcmsCallbackStruct` will be passed as one of the parameters to the callback function. Specifying a `NULL`
	
	`ptrNewPcmsCallback` parameter installs the default callback for the connection. This is a null function.

`ptrOldPcmsCallback`	is a pointer to a structure of type `PcmsCallbackStruct`, which will hold the previous callback function and client data. If this information is of no interest then you may specify `NULL`.

Return Codes

`PcmsSetCallback ()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo` and `PcmsErrorStr`.

Comments

The parameters passed to the callback function are:

`connectId`	is the database connection identifier.
`clientData`	is the pointer value specified in the `PcmsCallbackStruct` when the callback function was installed using `PcmsSetCallback()`.
`commandStatus`	is the status of the Dimensions CM command.
`commandId`	is the unique command identifier associated with the command. This value corresponds to that returned by `PcmsSendCommand()`.
`commandStr`	is the text of the command submitted.
`callData`	is the text output that has resulted from the command execution.
`...`	is a variable argument list that is used internally by Dimensions CM. You must not attempt to use this list.

Related Functions

`PcmsExecCommand()`,`PcmsSendCommand()`, `PcmsCheckMessages()`, `PcmsSetNoErrorCallback()`.

PcmsSetDbErrorCallback - Set Server Error Callback

Purpose

This function sets up a callback function to be executed when the RDBMS is no longer available. The callback function will be invoked when an application invokes a DTK function and the DTK detects that the Dimensions CM repository is no longer available. For example, this error may occur if the Dimensions CM server has been powered down.

Prototype

```
int
PcmsSetDbErrorCallback (
    int                 connectId,
    PcmsCallbackStruct  *ptrPcmsCallback
);

typedef struct
{
   PcmsCallbackProc callback;
   void             *clientData;
} PcmsCallbackStruct;
```

Parameters

`connectId`	is the database connection identifier.

`ptrPcmsCallback`	is a pointer to a structure of type `PcmsCallbackStruct`. In this structure the member field *callback* is a pointer to the callback function. This function must have the following prototype.
	`void sampleCallbackProc(`
	`intconnectId,`
	`void*clientData,`
	`intcommandStatus,`
	`intcommandId,`
	`char*commandStr,`
	`char*callData,`
	`...`
	`)`
	The `clientData` field of the `PcmsCallbackStruct` will be passed as one of the parameters to the callback function.

Return Codes

`PcmsSetDbErrorCallback ()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo` and `PcmsErrorStr`.

Comments

The parameters passed to the callback function are:

`connectId`	Database connection identifier associated with the command.
`clientData`	Null.
`commandStatus`	The RDBMS error code detected.
`commandId`	Zero.
`commandStr`	Null.
`callData`	The text output formatted to include the RDBMS error code(s) that has resulted from the command execution.
`...`	is a variable argument list that is used internally by Dimensions CM. You must not attempt to use this list.

Related Functions

`PcmsSetCallback()`.

PcmsSetDirectory - Change Dimensions CM Default Directory

Purpose

This function changes the default directory of the Dimensions CM process being managed by your application.

Prototype

```
int
PcmsSetDirectory (
    int    connectId,
    char   *new_directory
);
```

Parameters

connectId	is the database connection identifier.
new_directory	is the full specification of the default directory the Dimensions CM process is to use.

Return Codes

PcmsSetDirectory() returns:

PCMS_OK	on success.
PCMS_ERROR	on failure and sets PcmsErrorNo and PcmsErrorStr.

Comments

The function does not change directory until all commands in the PcmsSendCommand() queue (if any) are executed.

The directory change effects this connection only.

Related Functions

PcmsExecCommand(),PcmsSendCommand().

PcmsSetIdleChecker - Install Idle Checker

Purpose

This function installs an application function to be called before any Dimensions CM DTK functions are blocked on a read. `PcmsSetIdleChecker()` will be called before making a connection to a database, and may be useful in X applications to process events from the X event queue while the Dimensions CM DTK is waiting for input.

Prototype

```
int
PcmsSetIdleChecker(
    int    (*userIdleChecker )(int fd, int flag)
);
```

Parameters

`userIdleChecker`	is the address of the application function that will be called.

Return Codes

`PcmsSetIdleChecker()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

Comments

The prototype of the idle checker function is:

```
int idleCheckerFunction(int fd, int flag)
```

where

`fd`	is the file descriptor for the data connection that is about to be read.
`PCMS_ERROR`	is either `PCMS_MSG_WAIT` or `PCMS_MSG_NOWAIT`.

If the `flag` is set to `PCMS_MSG_WAIT`, the function should only return when data is available to be read from the file descriptor, in which case the return value will be `PCMS_OK`. If the flag is set to `PCMS_MSG_NOWAIT`, the function will return immediately with a return value of `PCMS_OK` if data is available; and a return value of `PCMS_FAIL` if no data is available.

For X applications, `XtAddInput()` can be used to set a callback procedure when input is pending on `fd`.

PcmsSetWsetObj - Set User's Current Project

Purpose

This function allows the user to reset the current project in which this Dimensions CM session is active.

Prototype

```
int
PcmsSetWsetObj (
    int    connectId,
    int    options,
    PcmsObjStruct  *ptrPcmsObjStruct,
    char   *dir);
```

Parameters

`connectId`	is the database connection identifier.
`options`	is not currently supported (use 0).
`ptrPcmsObjStruct`	is a pointer to a structure of type `PcmsObjStruct` that is populated by the user to indicate what project to change to.
`dir`	is the project directory to use.

Return Codes

`PcmsSetWsetObj()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on failure and sets `PcmsErrorNo`, `PcmsErrorStr`, `PcmsDbErrorNo`, and `PcmsDbErrorStr`.

PcmsWriteAttachments - Write Specified Request Attachments To Disk

Purpose

This function allows users to write request attachments to disk. It will process the array of attachments, writing the attachments out to a location specified by the "`userFile`" members.

The "`userFile`" member will be user-allocated memory, and the user will be responsible for freeing it.

Prototype

```
int PcmsWriteAttachments (
   int conId,
   PcmsObjStruct *objPtr,
   int archived,
   int noAttachs,
   PcmsChdAttachmentStruct **ptrAttachment
)
```

Parameters

`connectId`	is the database connection identifier.
object	is the request - `PcmsObjStruct` that has been initialized through `PcmsInitSpec()` or `PcmsInitUid()`.
archived	is the integer used to indicate if the specified request object can be found in the main catalog or the secondary catalog. `0` indicates the request can be found in the main catalog, `1` indicates the request can be found in the secondary catalog.
noAttachs	is the integer containing the number of attachments structures supplied.
ptrAttachment	is a pointer to a contiguous block of allocated memory that lists the attachment structures.

Return Codes

`PcmsSetWsetObj()` returns:

`PCMS_OK`	on success.
`PCMS_FAIL`	on failure or when no attachments are found.

Attribute Macros

Defined in the file `pcms_api.h` are a set of macros that have been provided to help you in writing your application. These macros are public but the structures that they use may change in the future and should not be used directly.

Initialize *PcmsObjStruct* attrs

- ***PcmsInitAttrStruct(objPtr, number)***

  This macro allocates *number* zero initialized memory structure of type `PcmsObjAttrStruct` to the `attrs` pointer and updates *noAttrs* accordingly.

  The parameters are:

`objPtr`	a pointer to a `PcmsObjStruct`.
`number`	the number of `PcmsObjAttrStruct`(s) for which to allocate memory.

NOTE This macro works only on `PcmsObjStructs` that have not had attributes already setup.

Add *attrDef* Structures

- ***PcmsAddAttrDefs(objPtr)***

  This macro allocates zero memory for all the `PcmsObjAttrDefStruct`(s) within a `PcmsObjStruct`.

  The parameter is:

`objPtr`	a pointer to a `PcmsObjStruc`.

NOTE This macro works only on `PcmsObjStructs` that have `NULL objPtr.attr[n].attrDefs`.

Single-Value Attributes (SVA)

- ***PcmsSvaSetVal (valuePtr, string, reserved)***

  This macro sets an attribute value in a single-value attribute.

  The parameters are:

`valuePtr`	the value structure being initialized and set (for a `PcmsObjStruct` object `objPtr` and attribute number n, `valuePtr` is `objPtr.attrs[n].value`).
`string`	the value itself expressed as a `char *` string.
`reserved`	is an integer field reserved for future use (use 0).

- ***PcmsSvaReSetVal (valuePtr, string, reserved)***

  This macro resets an attribute value in a single-value attribute.

  The parameters are:

`valuePtr`	the value structure being initialized and set (for a `PcmsObjStruct` object `objPtr` and attribute number n, `valuePtr` is `objPtr.attrs[n].value`).
`string`	the value itself expressed as a `char *` string.
`Reserved`	is an integer field reserved for future use (use 0).

- ***PcmsSvaGetVal (valuePtr)***

  This macro returns a `char *` corresponding to the string value of this attribute.

  The parameter is:

`valuePtr`	the value structure pointer being queried (for an object `objPtr` and attribute number n, `valuePtr` is `objPtr.attrs[n].value`).

Multi-Value Attributes (MVA)

Instead of a single value, multiple-valued attributes maintain a value-set that is accessed through the use of an index. You can use the following macros to access and set this value-set.

- ***PcmsMvaSetVal (valueSetPtr, index, string, reserved)***

  This macro appends a value to a value-set.

  The parameters are:

`valueSetPtr`	the value set being added to (for an object `objPtr` and attribute number n the `valueSetPtr` is `objPtr.attrs[n].value`).
`index`	the index into the list. This index is incremented within the macro. For the first value in the `value-set`, this index is zero.
`string`	the value itself as a `char *` string.
`reserved`	is an integer field reserved for future use (use 0).

- ***PcmsMvaNumVals(valueSetPtr)***

  This macro returns an integer value corresponding to the number of values currently in the value-set.

  The parameter is:

`valueSetPtr`	the value set to query.

- ***PcmsMvaGetVal (valueSetPtr, index)***

  This macro returns the value of the value-set at a specific `index` as a `char *` string.

  The parameters are:

`valueSetPtr`	the `valueSet` to query.
`index`	the integer index in the list of this attribute's value-set.

- ***PcmsMvaReSetVal(valueSetPtr, index, string,reserved)***

  This macro frees a certain indexed value and writes the new string into the same position given by the index.

  The parameters are:

`valueSetPtr`	the value-set to manipulate.
`index`	the integer index in the list of this attribute's value-set value.
`string`	the new string.
`reserved`	reserved

- ***PcmsMvaFree (valueSetPtr)***

  This macro frees a complete `PcmsMva`, that is, frees the whole list.

  The parameters is:

`valueSetPtr`	the `valueSet` to free.

NOTE It is useful to note that the SVA macros are only a convenience. It is possible to access all attribute value structures with `PcmsMvaNumVals()` and `PcmsMvaGetVal()`. For single value attributes `PcmsMvaNumVals()` will return 1.

Chapter 4

DTK API Functions for Windows Client Installations

Introduction	142
Building Client Applications	142
Sample Code Fragment	143
PcmsClntApiConnect - Connect to a Dimensions CM Database	144
PcmsClntApiDisconnect - Disconnect from a Dimensions CM Database	145
PcmsClntApiExecCommand - Execute a Dimensions CM Command	146
PcmsClntApiFree – Free Memory	147
PcmsClntApiGetLastError - Get the Last Dimensions CM Message	148
PcmsClntApiGetLastErrorEx - Get the Last Dimensions CM Message (Dynamic Buffer)	149
PcmsClntApiModeBinary - Set File Transfer Mode to Binary	150
PcmsClntApiModeText - Set File Transfer Mode to ASCII	151
PcmsClntApiSilentConnect - Connect Silently to a Dimensions CM Database	152
Additional Supported DTK Functions	154

Introduction

This chapter describes a set of functions that have been specifically written for Windows client machines that have had only the Serena® Dimensions® CM Windows clients installed. These functions give you access to the full functionality of the DTK but are specifically written for Windows clients.

The description of each function has the following components

Purpose	What the function does.
Prototype	The function prototype.
Parameters	Description of the parameters used in the function.
Return Codes	Codes returned (please refer to DTK return codes on "DTK Return Codes" on page 39 for further details).
Sample	Sample function call (if applicable).
Comments	Additional relevant information (if applicable).
Related Functions	Any related DTK function calls.

Before reading this chapter, ensure that you have familiarized yourself with the contents of Chapter 2, "Writing Dimensions CM DTK Applications" because it contains important information relating to data structures, return codes, and manifest constants that are referenced in this chapter.

Building Client Applications

The client application functions are available in the supplied:

- `clientapi.h` include file and
- `clientapi10m.lib` library file (for MBCS applications)
- `clientapi10u.lib` library file (for Unicode applications)

All the above files are located in the directory `%DM_ROOT%\pcms_api\`.

Any source file that references the functions or constants must include `clientapi.h`.

If your application references connection functions, such as `PcmsClntApiConnect()`, you must include the standard WinD2K file `windows.h` to properly compile the application.

Sample Code Fragment

```
PcmsObjStruct pObj = { 0 };
int conId = 0;
char errBuff[1024];
/* API will now connect and show the login dialog */
conId = PcmsClntApiConnect((HWND)NULL);
/* This is an example public API call looking for the Project
    "TEST_WORKSET" */
if (PcmsInitSpec(conId, "TEST:TEST_WORKSET",
    PCMS_WORKSET, &pObj)!=PCMS_OK)
{
    /* API called failed so get the error and display it */
    (void)PcmsClntApiGetLastError(conId, errBuff,
           sizeof(errBuff));
    (void)fprintf(stderr, errBuff);
    return;
}
/* List the directories in the Project "TEST_WORKSET" */
(void)PcmsClntApiExecCommand(conId, "LWSD TEST:TEST_WORKSET");
/* Now display the output */
(void)PcmsClntApiGetLastError(conId, errBuff, sizeof(errBuff));
(void)fprintf(stdout, errBuff);
/* Now disconnect */
(void)PcmsClntApiDisconnect(conId);
```

PcmsClntApiConnect - Connect to a Dimensions CM Database

Purpose

This function provides you with a login window allowing you to connect to a Dimensions CM database. This function will return a `connectId` (integer) that represents your database connection.

Prototype

```
int
PcmsClntApiConnect
(
    HWND parent=NULL
);
```

Parameters

`HWND parent`	is the parent window for the login dialog.

Return Codes

`PcmsClntApiConnect()` returns:

`connectId`	on successfully completing the connection to the Dimensions CM server.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

Related Functions

`PcmsClntApiSilentConnect()`

PcmsClntApiDisconnect - Disconnect from a Dimensions CM Database

Purpose

This function disconnects from the Dimensions CM database as specified by the `conId`. The `connectId` must be a valid `connectId` returned by `PcmsClntApiConnect()` or `PcmsClntApiSilentConnect()`.

Prototype

```
int
PcmsClntApiDisconnect
(
    int     connectId
);
```

Parameters

`connectId`	is the database connection identifier.

Return Codes

`PcmsClntApiDisconnect()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

Related Functions

`PcmsClntApiConnect()`, `PcmsClntApiSilentConnect()`

PcmsClntApiExecCommand - Execute a Dimensions CM Command

Purpose

This function sends a command to the Dimensions CM server specified by `connectId`.

Prototype

```
int
PcmsClntApiExecCommand
(
    int     conId,
    char    *command
);
```

Parameters

`connectId`	is the database connection identifier.
`command`	is a pointer to a user allocated character array that is populated with the command to be executed.

Return Codes

`PcmsClntApiExecCommand()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

PcmsClntApiFree – Free Memory

Purpose

This function is a wrapper to the C function `free()`. It must be used to free memory allocated by `PcmsClnt` API DTK functions, such as `PcmsClntApiGetLastErrorEx()`. The reason for this is that on some platforms, like Windows, the memory that is allocated within a shared library *must* be freed by the same shared library. If this is not done, then memory errors begin to occur.

Prototype

```
void PcmsClntApiFree (void *buffer);
```

Parameters

`buffer`	is a pointer to the memory block that will be freed.

PcmsClntApiGetLastError - Get the Last Dimensions CM Message

Purpose

This function allows you to access the output from the last Dimensions CM command that was run on the server through `PcmsClntApiExecCommand()`.

Prototype

```
int
PcmsClntApiGetLastError
(
    int     connectId,
    char    *errorBuffer,
    int     maxLength
);
```

Parameters

`connectId`	is the database connection identifier. If the error prevented the establishment of a connection, pass a `connectID` value of `-1`.
`errorBuffer`	is a pointer to a user allocated character array that is populated with the text of the server message.
`maxLength`	is the maximum length of the server message to be displayed.

Return Codes

`PcmsClntApiGetLastError()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

Related Function

`PcmsClntApiGetLastErrorEx()`

PcmsClntApiGetLastErrorEx - Get the Last Dimensions CM Message (Dynamic Buffer)

Purpose

This function allows you to access the output from the last Dimensions CM command that was run on the server through `PcmsClntApiExecCommand()`. The functionality of this command is the same as for `PcmsClntApiGetLastError()` except that the buffer size is dynamically allocated.

Prototype

```
int
PcmsClntApiGetLastErrorEx
(
    int     connectId,
    char    **errorBuffer
);
```

Parameters

`connectId`	is the database connection identifier. If the error prevented the establishment of a connection, pass a `connectID` value of `-1`.
`errorBuffer`	is a pointer to a contiguous block of allocated memory that is populated with the message text. It is the responsibility of the calling application to free this pointer when it is no longer required.

Return Codes

`PcmsClntApiGetLastErrorEx()` returns:

`PCMS_OK`	on success.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

Related Function

`PcmsClntApiGetLastError()`

PcmsClntApiModeBinary - Set File Transfer Mode to Binary

Purpose

This function sets the file transfer format to binary for any subsequent item commands that are issued.

Prototype

```
void
PcmsClntApiModeBinary
(
   int connectId
);
```

Parameters

`connectId` is the database connection identifier.

Return Codes

None.

Comments

You must use this command if you are going to perform operations that require file transfer from the client to the server (or vice versa) to occur in binary mode. An example of this might be getting a binary item into your PC. When you perform any file transfer operations, such as check in (`RI`), Dimensions CM will not validate the format of the file being transferred.

PcmsClntApiModeText - Set File Transfer Mode to ASCII

Purpose

This function sets the file transfer format to ASCII for any subsequent item commands that are issued.

Prototype

```
void
PcmsClntApiModeText
(
   int connectId
);
```

Parameters

`connectId` is the database connection identifier.

Return Codes

None.

Comments

You must use this command if you are going to perform operations that require file transfer from the client to the server (or vice versa) to occur in ASCII mode. An example of this might be getting an ASCII item into your PC. When you perform any file transfer operations, such as check in (`RI`), Dimensions CM will not validate the format of the file being transferred.

PcmsClntApiSilentConnect - Connect Silently to a Dimensions CM Database

Purpose

This function provides you with a connection to a Dimensions CM database as specified by your input parameters. This function will return a `conId` that represents your connection to the Dimensions CM server.

Prototype

```
int
PcmsClntApiSilentConnect
    (
    char    *user,
    char    *password,
    char    *host,
    char    *pcms_install,
    char    *db_name,
    char    *db_pword,
    char    *db_node
);
```

Parameters

`user`	is your operating system login name.
`password`	is your operating system password.
`host`	is the Dimensions CM server node name that you wish to connect to.
`pcms_install`	is the Dimensions CM server installation directory, for example, `/opt/serena/dimensions/10.1/cm`. If a null entry is submitted, this parameter is ignored.
`db_name`	is the name of the Dimensions CM database that you wish to connect to, for example, `intermediate`.
`db_pword`	is the password of the database. If a null entry is submitted, this parameter is ignored.
`db_node`	is the RDBMS service name assigned to the node where the RDBMS database is located.

Return Codes

`PcmsClntApiSilentConnect()` returns:

`int connectId`	on successful connection.
`PCMS_ERROR`	on error.
`PCMS_FAIL`	on failure.

Related Functions

PcmsClntApiConnect()

Additional Supported DTK Functions

In addition to the specific Win32 functions described in this chapter, the following standard Dimensions CM DTK functions are also available. These functions are fully documented in Chapter 3, "DTK API Functions for C/C++".

```
PcmsAttrDefInit()
PcmsAttrGetLov() *
PcmsAttrValidate() *
PcmsCntrlPlanGet() *
PcmsEvntCalloc()
PcmsEvntFree()
PcmsEvntMalloc()
PcmsEvntRealloc()
PcmsFullQuery() *
PcmsGetAttrDefNum() *
PcmsGetAttrs()
PcmsGetCandidates() *
PcmsGetPendingUsers() *
PcmsGetRSAttrs() *
PcmsGetRSNamest() *
PcmsGetUserRelTypes() *
PcmsGetUserRoles() *
PcmsGetWsetObj() *
PcmsInitSpec()
PcmsInitUid()
PcmsLovFree()
PcmsObjFree()
PcmsObjGetBackRels() *
PcmsObjGetRels() *
PcmsPendGet() *
PcmsPendWhoGet()
PcmsPopulate() *
PcmsQuery() *
```

NOTE Functions marked with an asterisk (*) require that the following file to be installed on each client: `%DM_ROOT%\msg\pcms_api_sql_uk.msb`. This can be accomplished by a default client installation or by copying the file from the Dimensions CM server, where it resides in the same directory.

Part 2

Events Callout

Part 2: Events Callout contains the following chapter:

Dimensions CM Events Callout Interface 157

Chapter 5

Dimensions CM Events Callout Interface

Description	158
Shared Libraries	158
MBCS Versus UTF-8 Support	158
Large File Support	159
Public Function Call	159
Event Callout Interface	160
Determining the Event You Want	163
First and Second Event Calls	164
Event Call Summary	165
Writing a DTK Callout Event	165
DTK Event Internals	168
Changing System-Attributes on Validate Events	169
Changing User-Attributes on Validate Events	170
Recommendations on How to Change Attribute Values	170
Calling DTK Functions Within Events	171
Using the ptrEventInfo in Events	172
Recompiling With New Versions of Dimensions CM	173
Event Examples	173
Events - A Final Word and a Warning	174

Description

This chapter outlines the functionality, design and implementation of applications using the Serena® Dimensions® CM Event Callout Interface. Before reading this chapter ensure that you familiarize yourself with the concept of shared libraries because it is through this mechanism that this event interface is implemented.

Shared Libraries

The Dimensions CM Event Callout Interface is provided by giving you access to a public function call that is invoked when certain Dimensions CM commands are run. This function is called `userSuppliedFunction()` and is resolved in a shared library called `libpcmsu`. On a default Dimensions CM installation a stub version of this library is provided. If you wish to implement your own event callout, you will need to build your own shared library and use this in place of the stub.

For more information on how to build shared libraries please refer to your system documentation or for guidelines refer to the examples provided in:

- UNIX: `$DM_ROOT/pcms_api/examples/`
- Windows: `%DM_ROOT%\pcms_api\examples\`

MBCS Versus UTF-8 Support

Dimensions CM supports both Multi-Byte Character Sets (MBCS) and UTF-8 Character Sets. This means, however, that depending on what type of database you are connecting to, the way you compile your events may have to be different, namely:

- If you are connecting to a UTF-8 database, then your events will have to be compiled using the compiler directive `-DDM_UNICODE`.
- If you are connecting to any type of database other than UTF-8, then your events will have to be compiled using the compiler directive `-D_MBCS`.

You can determine the nature of the database you are connecting to by selecting from the `DM_CHARACTERSET` view in the `PCMS_SYS` schema. If this view returns null or a value other than AL32UTF8, then you are connecting to an MBCS database.

IMPORTANT! A SQL Server RDBMS is *always* a MBCS database.

The example DTK events provided generate UTF-8 compatible shared libraries by default, that is, the libraries ending end in `'u'`. To change this behavior, use the `MBCS_NEEDED=y` make flag when you run the Dimensions CM `dm_make` command. This will generate a shared library with `'m'` at the end of the library name.

Copy the appropriate library you require to your usual installation directory:

- If your database has a Unicode/UTF-8 character set, rename the generated library file from `libpcmsuu.so` to `libpcmsu.so`.
- For any other kind of database character set, you must *not* rename the library name, that is, you must leave it with the 'm' extension.

If you use an MBCS shared library against a UTF-8 database or vice versa, the results will be unpredictable as the server will probably experience memory issues.

Large File Support

The Dimensions CM 2009 R1 or later API has added support for large files (4GBytes or greater). See Step 4 of "Introduction" on page 38 for a discussion of this topic.

Public Function Call

The prototype for the public function call `userSuppliedFunction()` is:

```
int userSuppliedFunction(
            PcmsEventStruct *ptrPcmsEventStruct,
            PcmsObjStruct *ptrObj,
            PcmsObjStruct *ptrUser,
            char  **ptrErrorMessage,
            int  *noEventInfo,
            void  **ptrEventInfo);
```

where the parameters are:

`ptrPcmsEventStruct`	is a pointer to a `PcmsEventStruct` in which the details on the current event being fired are held.
`ptrObj`	is a pointer to a `PcmsObjStruct` that holds the object details pertaining to the Dimensions CM object currently being processed.
`ptrUser`	is a pointer to a `PcmsObjStruct` that holds the details on the user currently running the event.
`ptrErrorMessage`	is a pointer to a pointer that allows you to setup an error message to be printed instead of the default Dimensions CM message.

noEventInfo	is an integer variable that is used in context with ptrEventInfo to access members of that pointer.
ptrEventInfo	is a pointer to a void *pointer that is populated with different information depending on the event called. For PCMS_EVENT_MAIL, PCMS_EVENT_DLGI, and PCMS_EVENT_DLGC events this pointer will point to an array of PcmsUserRoleStructs. For a PCMS_EVENT_RELATE or PCMS_EVENT_UNRELATE event this pointer will point to an array of PcmsRelStructs.

Event Callout Interface

As indicated previously, when certain Dimensions CM commands are run, the public function userSuppliedFunction() is invoked with a number of parameters. These parameters are used to specify what event is being fired; what Dimensions CM object is being affected, and finally which type of Dimensions CM command is being run. Each time an event is fired the following hierarchy of calls is made to userSuppliedFunction():

- Validate event call (fired *only* when a user or system attribute has changed).
- Pre-event call.
- Post-event call.

Each of the event calls allows you to perform a number of operations on the Dimensions CM object on which the event has been called.

You can access the type of event being fired by examining the ptrPcmsEventStruct as described on page 163.

Validate Events

Validate events are called prior to any Dimensions CM validation being run on the data supplied. Typically, you can use validation events to inspect information (such as the object details, default, and user attributes) and then change this information. You could, for example, use this event to implement your own automatic item identity generator or perform extra validation on user-defined date attributes. Once this event has been fired Dimensions CM will then proceed with its normal validation checks. This event is indicated by the use of the constant PCMS_EVENT_VALIDATE_OP.

NOTE Validate events are fired *only* when user or system attributes change as a result of a command. This means that events captured on the validate event will be Type PCMS_EVENT_UPDATE events

Pre-Events

Pre-events are called prior to the Dimensions CM command being executed. You can use this event to stop the execution of this command by returning the failure status

PCMS_FAIL. If you populate ptrErrorMessage, then this error string will be displayed through whatever interface invoked the command. Typically, you can use this event to perform any specific validation before deciding to let the command proceed. This event is indicated by the use of the constant PCMS_EVENT_PRE_OP.

Post-Events

Post-events are called after the Dimensions CM command has been run. Typically you can use this event to perform any 'clean up' or post-command logging to other applications. This event is indicated by the use of the constant PCMS_EVENT_POST_OP.

Event Types

In addition to the event hierarchy described in the previous sub-sections, each event fired also has an event type that relates to the type of Dimensions CM command being run. These event types are:

Event Type	Activity to a Dimensions CM Object
PCMS_EVENT_ACTION	Actioned.
PCMS_EVENT_ADD	Addition.
PCMS_EVENT_CANCEL	Check out of object is canceled.
PCMS_EVENT_CREATE	Created.
PCMS_EVENT_DELETE	Deleted.
PCMS_EVENT_DELIVER	Delivered (project or stream)*.
PCMS_EVENT_DEMOTE	Demoted.
PCMS_EVENT_DLGC	Request is delegated to a user.
PCMS_EVENT_DLGI	Item is delegated to a user.
PCMS_EVENT_EXTRACT	Object is checked out.
PCMS_EVENT_FETCH	Object is gotten/fetched (or browsed).
PCMS_EVENT_LOCK	Object is locked.
PCMS_EVENT_MAIL	A Dimensions CM mail message is being sent.
PCMS_EVENT_PROMOTE	Promoted.
PCMS_EVENT_RELATE	Object is related to another object.
PCMS_EVENT_REMOVE	Removal.
PCMS_EVENT_REMNAME	Renaming.
PCMS_EVENT_RETURN	Object is checked in.
PCMS_EVENT_UNRELATE	Object is unrelated from another object.
PCMS_EVENT_UNLOCK	Object is unlocked.
PCMS_EVENT_UPDATE	Updated.
PCMS_EVENT_UPLOAD	Project is uploaded to*.

* These events are for information purposes only (PCMS_EVENT_PRE_OP) or for aborting the operation (PCMS_EVENT_POST_OP)—you cannot manipulate the objects passed into them (PCMS_EVENT_VALIDATE_OP)

While a single Dimensions CM command, such as 'getting an item', can fire a single event type (PCMS_EVENT_FETCH) it is also possible for a command to generate multiple different events. Consider, for example, the following command:

```
UI "FS:TEST.A-SRC,1" /REV=2.1/ATTRIBUTE=(COMPLEXITY="Delta7")
```

This command performs a check out, check in, and an update of attribute values. Thus, if you ran the command as described previously, the following events would be fired.

Event Type	Action to a Dimensions CM Item
PCMS_EVENT_EXTRACT	Checked out from Dimensions CM.
PCMS_EVENT_MAIL	Notification of an event happening that sends out email.
PCMS_EVENT_RETURN	Checked in to Dimensions CM.
PCMS_EVENT_UPDATE	Attributes updated.

If you added additional parameters to this command, such as /STATUS or /RELATE_CHDOC, other events would also have been fired.

The table below describes the commands that fire events and the objects types they involve.

EventId	PART	ITEM	REQUEST	PROJECT	BASELINE	RELEASE
*PCMS_EVENT_*						
ACTION	SPV, UP	AI, SI	AC		AWS	
ADD		AIWS				
CANCEL	(N/A)	CIU				
CREATE	CP, CPV	CI	CC	CWSD, DWS, MWS	CBL, CMB, CRB	REL
DELETE	DPV	DI		DWSD, RWS	DLB	DREL
DEMOTE		DMI	DMRQ		DMBL	
DLGC	(N/A)		DLGC			
DLGI		DLGI				
EXTRACT	UP	EI, UI				
FETCH		FI	BC			
LOCK		LCK		LCK		

EventId	PART	ITEM	REQUEST	PROJECT	BASELINE	RELEASE
*PCMS_EVENT_*						
MAIL	CP, DPV, SPV, UP, UPA, CPV	AI, AIWS, CI, DI, DLGI, DPI, EI, LCK, RI, RIWS, UI, UIA, ULCK	AC,CC, DLGC, RBCD, RCCD, RICD, RWCD, UC, XBCD, XCCD, XICD, XWCD	AWS, DWS, LCK, RWS, ULCK, UWA	ABL, CBL, CMB, CBL, DBL	
PROMOTE		PMI	PMRQ		PMBL	
RELATE	RIP	RII	RCCD, RICD, RPCD, RWCD			
REMOVE		RIWS		MWSD		
RENAME				SWF		
RETURN	UP	RI, UI				
UNLOCK		ULCK		ULCK		
UNRELATE	XIP	XII	XCCD, XICD, XPCD, XWCD			
UPDATE	UPA, UP, CP, CPV	CI, EI, RI, UI, UIA	CC, UC	UWA	UBLA	

Determining the Event You Want

When the `userSuppliedFunction()` is invoked, one of the parameter passed in is a pointer to a `PcmsEventStruct` that can be interrogated for details such as:

- The event type, for example, `PCMS_EVENT_CREATE`.
- Where in the hierarchy the event is being fired, for example, validate or pre-event.
- The object type the event is being fired on, for example, `PCMS_ITEM`.

By examining the following fields of the `PcmsEventStruct` you can trap the event you specifically require.

`eventId` the event type.

whenCalled	the position in the event hierarchy.
objType	the type of object that the event was fired on.

First and Second Event Calls

Populating all the parameters for an event can take time, especially if you are connected to a remote database over a WAN. As a result of this, each time an event is to be fired you get a chance to determine whether or not you are really interested in that event. This separation is known as the 'first and second call' to an event.

The first call to an event populates only the sparse details on the Dimensions CM object and the PcmsEventStruct to allow you to determine if you are interested in this event. The specific details provided in the PcmsObjStruct depend on what command is called, on what object, and under what circumstances—as shown in the next two tables.

Object type	First call details
Baselines	objType, typeUid, typeName, productId, objId, variant, revision, userName, status, dateTime
Items	objType, typeUid, typeName, productId, objId, revision, userName, status, dateTime
Parts	objType, typeUid, typeName, productId, objId, userName, dateTime
Projects	objType, typeUid, typeName, productId, objId, variant, revision, description, userName, status, dateTime
Releases	objType, productId, objId
Requests	objType, productId, typeName, status, typeUid

Object type	Second call details
Baselines	uid, objType, typeUid, typeName, productId, objId, variant, revision, userName, status, dateTime
Items	uid, objType, typeUid, typeName, productId, objId, variant, revision, description, userName, status, dateTime
Parts	uid, objType, typeUid, typeName, productId, objId, variant, revision, description, userName, status, dateTime
Projects	uid, objType, typeUid, typeName, productId, objId, variant, revision, description, userName, dateTime
Releases	uid, objType, productId, objId, dateTime
Requests	uid, objType, typeUid, typeName, productId, objId, userName, status, dateTime

A first call event can be determined by checking if:

```
ptrUser == (PcmsObjStruct *)0 and ptrErrorMessage == (char *)0
```

If you select the first call, then only the `ptrPcmsEventStruct` will be fully populated. The `ptrObj` will only have the object-specification fields and `uid` filled in. If as a result of examining these partial details, you decide that you really want this event, then returning the status `PCMS_OK` will generate a second event call. If you are not interested in this event, then return `PCMS_FAIL` and the second call to this event will not be fired.

The second event call can be regarded as the 'real' event call. This has all the data structures filled in and allows you to access all the attribute information for the object. It is on this call that your event should perform its operation.

Event Call Summary

The following table summarizes the event callout mechanism.

Step	Command
1	Read Dimensions CM Command.
2	FIRST CALL VALIDATE EVENT and check that VALIDATE EVENT is required.
3	If VALIDATE EVENT is required, then call VALIDATE EVENT again.
4	If VALIDATE EVENT not required, then END.
5	Evaluate User Data and Execute the Dimensions CM Command.
6	FIRST CALL PRE-EVENT and check that PRE-EVENT is required.
7	If PRE-EVENT is required, then call PRE-EVENT again with fully detailed data.
8	If PRE-EVENT not required, then END.
9	COMMIT Dimensions CM Command to database.
10	FIRST CALL POST-EVENT and check that POST-EVENT is required.
11	If POST-EVENT is required, call POST-EVENT again with fully detailed data.

Steps 3, 7, and 11 occur when the real VALIDATE, PRE and POST EVENTS are done.

Writing a DTK Callout Event

This section describes how to design and write a DTK event, the pitfalls to watch out for, and in what situations an event is applicable.

Is an Event the Solution for You?

Before you start to write an event to implement your solution, you must decide if an event is what you really need. To help you decide bear in mind the following questions.

- Is the functionality that you seek to achieve already in Dimensions CM?

  If you are seeking to implement stronger rules for object relationships or attributes, this functionality is already available in Dimensions CM.

- Is the functionality that you seek to achieve initiated by a Dimensions CM command?

  The events are strictly intended to allow you to perform inline processing or checking on a specific object. They are not intended to allow you to run multiple Dimensions CM commands on the same object. When a validate-event or pre-event is fired on an object, then that object becomes locked until the transaction has been committed to the database. If you try to spawn another Dimensions CM command, then you run the following risks:

  - The Dimensions CM command that you spawned will be suspended waiting for the lock to be released, and it will never be released until a time-out has occurred.
  - The Dimensions CM command will cause the same event to be fired that will spawn yet another Dimensions CM command that calls the same event spawning yet another command, and so on until your machine locks up.

  If you are intending to use an event to spawn other Dimensions CM related commands, then you are strongly advised to use a separate DTK client application to perform this sequencing of commands.

- Does the operation you want to capture actually fire an event?

  Not all Dimensions CM commands fire events. You need to be sure that the operation you want to capture actually fires an event.

Designing Your Event

When you have decided that your solution requires an event, you need to decide which event you need to capture, and which type of event you have to use. Events are fired when you run a Dimensions CM command, so decide on the list of Dimensions CM commands that invoke the events you want and scope this list to the type of objects you wish to process. Examining this list you may discover that you may need to filter out certain events, commands or objects to obtain just the processing you want. You will need to code this filtering into your event. When you are looking at this list remember that additional parameters can call additional events. You can filter the events that you select by either examining the `ptrPcmsEventStruct` as described previously in "Public Function Call" on page 159, or you can access the actual Dimensions CM command being run through `PcmsGetCommandLine()` and filter on this.

Once you know the list of commands and type of events that you are going to trap, you need to consider where in the callout hierarchy this trapping will occur. The basic rules are to trap:

- The validate event if you wish to change any information.
- The pre-event if you wish to be able to stop the operation.
- The post-event if you wish to perform an action after the Dimensions CM operation has committed data to the database and freed all the locks on that object.

Suggestions for More Common Operations

- Changing attributes (user and system) or setting defaults

  To change, set or reformat attributes you have to capture the PCMS_EVENT_UPDATE at the PCMS_EVENT_VALIDATE_OP stage in the call hierarchy.

- Changing object Id on creation

  If you wish to wish to change the object identifier or filename used when an object is created, then you have to capture the PCMS_EVENT_CREATE and PCMS_EVENT_UPDATE events at the PCMS_EVENT_VALIDATE_OP stage in the call hierarchy.

- Allowing the action of an object only if certain criteria are matched

  If you have specific checks that you wish to perform before objects are actioned from one state to another e.g. releasing a baseline to test, then you have to capture the PCMS_EVENT_ACTION event at the PCMS_EVENT_PRE_OP stage in the call hierarchy.

- Logging to another application that a Dimensions CM operation has occurred

  If you have integrated Dimensions CM with another application and wish to signal to that application that a command has been run, then you have to capture all the events at the PCMS_EVENT_POST_OP stage in the call hierarchy.

Writing Your Event

You have to write your event code with the userSuppliedFunction() as the interface point between Dimensions CM and your event code. The return codes from this function call are the standard PCMS_OK, PCMS_FAIL, and PCMS_ERROR. These return calls have the following effects:

First Event Call	
PCMS_OK	causes a second event call to occur.
PCMS_FAIL	causes Dimensions CM to ignore this event and continue with its normal processing for the operation. No second event call will be made.

Second Event Call	
PCMS_EVENT_VALIDATE_OP	PCMS_OK will allow the Dimensions CM operation to continue. If any attributes have been changed, then the new values will be used. PCMS_FAIL will cause the Dimensions CM operation to fail and any error messages in the ptrPcmsErrorMessage pointer will be printed. PCMS_ERROR is the same as PCMS_FAIL
PCMS_EVENT_PRE_OP	PCMS_OK will allow the Dimensions CM operation to continue.

Second Event Call	
	PCMS_FAIL will cause the Dimensions CM operation to fail and any error messages in the ptrPcmsErrorMessage pointer will be printed. PCMS_ERROR is the same as PCMS_FAIL.
PCMS_EVENT_POST_OP	Because the operation has been completed, the status at this point is irrelevant, but for consistency you should return PCMS_OK.

In the first event call you need to interrogate the ptrPcmsEventStruct to determine whether or not to trap this event and, if so, return PCMS_OK, else return PCMS_FAIL.

In the second event call you need to place the code to support your event logic and return the appropriate status.

DTK Event Internals

When an event is passed to your function you are able to both manipulate the data supplied and/or view many of the internal changes that have occurred or will occur on an object as a result of the Dimensions CM command. This section discusses what information these pointers give you and how you can use them to achieve various different effects.

- The PcmsEventStruct pointer – ptrPcmsEventStruct

  This pointer is the most important structure passed down to an event. It is used to both control the event operation and also to indicate what attributes (both user and system) have been affected. How to determine which event is being called has already been discussed. How the attributes are controlled is determined through the noAttrsChanged and attrsChanged members of this pointer.

  When events are fired, any system or user defined attributes that have been modified as a result of the command are populated into the noAttrsChanged and attrsChanged members. On a validate-event you can manipulate these variables to add, reset or remove attribute values:

  - If you are resetting an attribute value, then loop through the attrsChanged pointer until you find the attribute structure that you wish to change. Once you have found this attribute, you can use the multi value attributes (MVA) or single value attributes (SVA) macros to reset the attribute value. If you are resetting values on a MVA attribute, then you must first free the memory associated with this attribute through PcmsMvaFree() before adding your new values.
  - If you are adding a new attribute value, then you will need to:
    - Resize the attrsChanged pointer to add a new PcmsObjAttrStruct.
    - Increment the noAttrsChanged index by 1.
    - Set the appropriate values on the new attribute structure.

    It is important to note that the attrDef pointer in the attribute structure must be set to NULL.

- If you are removing an attribute definition, then you will need to resize the `attrsChanged` pointer to remove this attribute and decrease the `noAttrsChanged` index by 1.

- The `PcmsObjStruct` – pointer `ptrObj`

  This pointer contains all the details on the object that the event is currently processing. On a first call to the event this object contains only minimal data. On the second call to the event this object is fully populated.

  When you create a new object, such as an item or a part, a validate-event is fired during which you can manipulate the contents of this pointer to change the object's details. You are able to change entries in the `objId`, `variant`, and `revision` fields. Using this mechanism you could write an event that changes item `Ids` to suit your own automatic object identity generator.

- The `PcmsObjStruct` – pointer `ptrUser`

  This pointer contains all the details on the user currently running the event. This pointer is populated only on the second call and is 'read only'.

- The error pointer – `ptrErrorMessage`

  This pointer allows you to set up an error message that will be printed by Dimensions CM when a validate-event or pre-event returns a status other than `PCMS_OK`. This allows you to print your own custom error messages when an event fails.

- The `noEventInfo` and `ptrEventInfo` pointers

  These pointers operate together. Their usage is described on page 172.

Changing System-Attributes on Validate Events

While there are no restrictions on changing user-defined attributes in the validate event, you are, however, limited to what system attributes you can change. While you can modify the *attrsChanged* pointer to include any system attribute definition, only the following will have any affect.

Event Type PCMS_EVENT_ :	Object Type PCMS_ :	System Attribute PCMS_ATTR_ :
CREATE	ITEM	FORMAT
		FILENAME
		LIB_FILENAME
		DIRPATH
		USER_FILENAME
	PART	PARTNO
		LOCALNO
	RELEASE	DIRPATH

Event Type PCMS_EVENT_ :	Object Type PCMS_ :	System Attribute PCMS_ATTR_ :
EXTRACT[a]	ITEM	FORMAT
		USER_FILENAME
		REVISION
RETURN[b]	ITEM	FORMAT
		USER_FILENAME

a.Check out.
b.Check in.

If you specify any other values for system-attributes, these will be ignored.

The above-mentioned attributes can be changed only when a new object is created or an item is checked out at a new revision.

Changing User-Attributes on Validate Events

There are no restrictions on what you can do with user-defined attributes on a validate event. However, Dimensions CM will apply the usual validation to any attributes that you setup in the `attrsChanged` structure. If the attribute is not defined, has the wrong value or the user does not have the role to change it, then Dimensions CM will generate an appropriate error message.

Recommendations on How to Change Attribute Values

The following steps provide a recommended approach on how you should change the `attrsChanged` pointer.

- Examine the `attrsChanged` structure in the `ptrPcmsEventStruct`.
- If the pointer is `NULL`, you need to allocate memory to this pointer (that is the size of `PcmsObjAttrStruct`) and set `noAttrsChanged` to 1. Once the memory has been allocated, then set the members of the `attrsChanged` pointer to the appropriate values.

  For example, for an SVA

  ```
  PtrPcmsEventStruct->attrsChanged[0].attr=<ATTR_NO>
  PcmsSvaSetValue(ptrPcmsEventStruct->attrChanged[0].value,
                  "text String",0);
  ptrPcmsEventStruct->attrChanged[0].attrDef =
                  (PcmsObjAttrDefStruct *)0;
  ```

- If the pointer currently has attributes setup, then you need to check if the attribute you want to change is currently in that pointer. You can do this by looping through the `attrChanged[x].attr`(s) and looking for a match to your attribute number. Once you have found a match then use `PcmsSvaSetVal()` or `PcmsMvaReSetVal()`to reset

the value. If you are resetting the MVA values for an attribute, then remember to free the attribute value set first through `PcmsMvaFree()`.

- If the pointer currently has attributes setup, but you cannot find a match using the search method indicated above then you need to re-allocate memory to the `attrsChanged` pointer to add a new `PcmsObjAttrStruct`. Using this newly allocated structure you can set the attribute values as described above and increment the `noAttrsChanged` variable by `1`.

Calling DTK Functions Within Events

When you call DTK functions from within an event the connection identifier (`conId`) that you need to use is `0`. This is a special connection identifier that relates to the current connection that Dimensions CM has to your database. You do not need to call `PcmsConnect()` or `PcmsDisconnect()` to access the DTK functions. If you try to use these functions to initiate a connection to the database or another database, then your Dimensions CM session may become unstable.

Specialist DTK Event Functions

There are a number of DTK functions that, although available to DTK client applications, are specifically aimed at helping you to write your event. These functions are aimed at memory management and accessing the Dimensions CM command line.

DTK Function	Description
`PcmsEvntCalloc()`	Wrapper to `calloc()`.
`PcmsEvntFree()`	Wrapper to `free()`.
`PcmsEvntMalloc()`	Wrapper to `malloc()`.
`PcmsEvntRealloc()`	Wrapper to `realloc()`.
`PcmsGetCommandLine()`	Access to the Dimensions CM command that invoked this event.

Unsupported DTK Function Calls from Within an Event

You can call virtually all the DTK functions from within an event. There are, however, a number of exceptions to this rule. Some of the DTK functions, due to the nature of the command that they are running, are not allowed to be called from within an event. These functions are listed below.

DTK Function	Description
`PcmsCheckMessages()`	Check results from Dimensions CM commands.
`PcmsConnect()`	Connect to a Dimensions CM database.
`PcmsDisconnect()`	Disconnect from a Dimensions CM database.
`PcmsExecCommand()`	Execute a Dimensions CM command.
`PcmsGetConnectDesc()`	Get current connection details.

DTK Function	Description
PcmsSendCommand()	Execute a Dimensions CM command.
PcmsSetAttrs()	Set attributes on a Dimensions CM object.
PcmsSetCallback()	Set callback functions.
PcmsSetDbErrorCallback()	Set callback functions.
PcmsSetDirectory()	Change working directory.
PcmsSetIdleChecker()	Set callback functions.

Using the ptrEventInfo in Events

The special void * pointer ptrEventInfo is filled in when certain events are fired to provide you with additional information pertinent to those events. The information contained in the pointer will change depending on the event that is being fired. The following table lists the structures that this pointer needs to be typecast to depending on the event being fired.

DTK Event	DTK Structure
PCMS_EVENT_CREATE	PcmsChdAttachmentStruct
PCMS_EVENT_DELIVER	PcmsDeliverStruct
PCMS_EVENT_DLGC	PcmsUserRoleStruct
PCMS_EVENT_DLGI	PcmsUserRoleStruct
PCMS_EVENT_FETCH	PcmsChdAttachmentStruct
PCMS_EVENT_MAIL	PcmsUserRoleStruct
PCMS_EVENT_RELATE	PcmsRelStruct
PCMS_EVENT_UNRELATE	PcmsRelStruct
PCMS_EVENT_UPDATE	PcmsChdAttachmentStruct

If you need to access the information in these structures, then your event needs to do the following:

- For RELATE and UNRELATE event types, typecast the ptrEventInfo through:

```
PcmsRelStruct *ptrRel = (PcmsRelStruct*)*ptrEventInfo;
```

- For request CREATE, FETCH, and UPDATE event types, typecast the ptrEventInfo through:

```
PcmsChdAttachmentStruct *ptrAttachs =
    (PcmsChdAttachmentStruct*)*ptrEventInfo;
```

- For other events, typecast the pointer through:

```
PcmsUserRoleStruct *ptrRel = (PcmsUserRoleStruct*)*ptrEventInfo;
```

Once you have typecast the pointer, you can use the ptrEventInfo pointer to access the information. For example, in a MAIL event you might do the following:

```
{
        PcmsUserRoleStruct   *ptrUser =
             (PcmsUserRoleStruct*)*ptrEventInfo;
        int                  noUses  = *noEventInfo;
        int                  i = 0;

        for (i = 0; i < noUsers; i++)
        (void)fprintf(fd,"\nFound user : %s",ptrUser[i].user);
}
```

In Delegate Events (`DLGI` and `DLGC`) the `applyDeny` and `treeWalk` members of `PcmsUserRoleStruct` have special meanings that relate to the option specified on the command line. These meanings are listed below:

Operation	applyDeny	treeWalk
/ADD	Y	Y
/DELETE	N	N
/REPLACE	Y	N

Recompiling With New Versions of Dimensions CM

You must recompile any application that uses the Events Callout Interface after upgrading to a new version of Dimensions CM. This ensures that your applications make use of any changes to the events in new versions of Dimensions CM.

Event Examples

The release media contains a number of example events and makefiles to help you get started. These are contained in the `examples` subdirectories of the Dimensions CM installation specified below:

- `$DM_ROOT/pcms_api/examples` (UNIX)
- `%DM_ROOT%\pcs_api\examples` (WINDOWS)

IMPORTANT! To build the example events use the following compiler versions:

- Solaris: `Sun C++ 5.10 SunOS_sparc 2009/06/03`
- HPUX Itanium: `aC++/HP C Version A.06.25`
- Windows 64bit: `MSVC++2010`
- Linux: `GCC 3.4.6`
- AIX: `IBM XL C/C++ for AIX V11.1`

Events - A Final Word and a Warning

Events are a powerful and versatile way of extending the rich functionality of Dimensions CM. They allow you to implement your own specific process controls and integrations with external applications. Used well, events can enhance both your working practices and use of Dimensions CM. However, if your events have not been written carefully, you do run the risk of affecting Dimensions CM functionality, especially if your event causes memory corruption. You are strongly advised to test any complex events thoroughly before deploying them, and if possible use a memory tracking tool to verify memory use.

Part 3

Java API

Part 3: Java API contains the following chapters:

dmclient 177

Chapter 6

dmclient

Introduction	178
Using dmclient	179
Javadoc	181
Examples	181

Introduction

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

The table below highlights the important interfaces and classes that a consumer of the API is likely to make frequent use of. Examples of their use can be found in examples located in the Dimensions CM installation (see "Examples" on page 181).

Name	Interface /Class	Description
DimensionsArObject	Interface	Represents a Dimensions CM business object that has attributes and relationships to other business objects.
DimensionsConnectionManager	Class	Entry point to the API for obtaining an instance of the DimensionsConnection interface.
DimensionsConnection	Interface	Represents a connection to a Dimensions CM server. Provides access to an instance of the DimensionsObjectFactory interface.
DimensionsDatabase	Interface	Represents the base database to which the current connection pertains. Provides access to collections of key Dimensions CM business objects such as products and attribute definitions.
DimensionsLcObject	Interface	Represents a Dimensions CM business object that has a lifecycle.
DimensionsObjectFactory	Interface	Provides access to key Dimensions CM business objects such as the base database and the current user.

dmclient has the following main features:

- **Multiple Connections**

  Dimensions CM clients can maintain multiple, concurrent connections to disparate Dimensions databases. Serena recommends that Dimensions CM clients do not establish multiple, concurrent connections to a single Dimensions CM database.

- **Multithreaded Access**

  Dimensions CM clients can access the Java API from multiple concurrent threads. No upper limit is imposed on the number of threads that may access the API concurrently.

- **Caching**

  Dimensions CM objects are represented as Java objects; references to these objects may be cached by a client. However, objects are not automatically updated when the

underlying data changes. Therefore, clients should expect a runtime exception to be thrown if they attempt to call a business method on a cached object that no longer exists in the database.

- **Querying the Database**

  Ad-hoc SQL querying is not supported. The Java objects that represent Dimensions objects make use of a key-value system of attributes for storing their properties. These properties are cached in the client's memory. The `getAttribute()` method is used to read a property from the cache, and the `queryAttribute()` method is used to read a property from the database.

  In addition to properties, a Dimensions CM object may have related Dimensions CM objects, which are also cached in the client's memory. The `getParentXXX()` and `getChildXXX()` methods on an object are used to read related objects from the cache. The `queryParentXXX()` and `queryChildXXX()` methods are used to read related objects from the database.

- **Networking**

  No restrictions are placed on Dimensions CM clients when they access a Dimensions CM database locally, over a LAN, or over a WAN. Clients should expect API performance over a LAN to be better than over a WAN.

- **Exceptions**

  The majority of API methods do not throw checked exceptions. If a method can throw an unchecked exception, it is noted in the method's Javadoc.

- **Connection Durability**

  Persistent connection loss is only likely to occur if either the server is down or if the physical connection between client and server has been broken. In those circumstances, either a `DimensionsNetworkException` will be thrown or you will get unexpected results from API methods. In this case, to be absolutely sure the server is unavailable, consumers of the API should call `DimensionsConnection.getConnection(true)`, which will cause dmclient to ping the server. If it returns anything other than `DimensionsConnection.STATE_CONNECTED`, either the server is unavailable or the session has expired, so your code should behave accordingly.

- **Java Compatibility**

  The `dmclient` API is compatible with Java 6 or later, but Java 7 or later is recommended on all supported platforms.

Using dmclient

jar files

To use `dmclient`, include the following jar files in your runtime classpath:

- `dmclient.jar`
- `darius.jar`
- `dmnet.jar`
- `dmfile.jar`
- `commons-logging.jar`

Windows example

```
java -cp %DM_ROOT%\java_api\lib\dmclient.jar
%DM_ROOT%\java_api\lib\darius.jar
%DM_ROOT%\java_api\lib\dmnet.jar
%DM_ROOT%\java_api\lib\dmfile.jar
%DM_ROOT%\java_api\lib\commons-logging-api.jar mypackage.MyClient
```

UNIX example

```
java -cp ${DM_ROOT}/java_api/lib/dmclient.jar
${DM_ROOT}/java_api/lib/darius.jar
${DM_ROOT}/java_api/lib/dmnet.jar
${DM_ROOT}/java_api/lib/dmfile.jar
${DM_ROOT}/java_api/lib/commons-logging-api.jar mypackage.MyClient
```

Entry Point to dmclient

The entry point to dmclient is

```
com.serena.dmclient.api.DimensionsConnectionManager
```

This point provides access to a

```
com.serena.dmclient.api.DimensionsConnection
```

object and from there to a

```
com.serena.dmclient.api.DimensionsObjectFactory
```

interface, which provides access to the top level Dimensions CM objects.

Getting Started

To get started, include the following lines at the start of a Java program:

- `import com.serena.dmclient.api.`
- `import com.serena.dmclient.collections.`
- `import com.serena.dmclient.objects.`

TIP To begin you only need access to the first package listed above.

Time Format Strings in Java-API

If you use the JAVA API to search for Dimensions CM objects with an Action date greater than a specified date, a time format string value needs to be passed for the filter.

For example, to search for requests with an action date greater than a specified date, you would have a code like:

```
lastCheck = Calendar.getInstance().getTime();
filter.criteria().add(new
    Filter.Criterion(SystemAttributes.LAST_ACTIONED_DATE, lastCheck,
    Filter.Criterion.GREATER_EQUAL));
List<Request> releaseRequests =
    conn.getObjectFactory().getBaseDatabase().getAllRequests(filter);
```

The time format string you need to supply needs to be the same as returned by the query:

- For non-Oracle databases this is `"DD-MON-YYYY HH24:MI:SS"`, where the three letter month name is one of `"JAN"`, `"FEB"`, `"MAR"`, `"APR"`, `"MAY"`, `"JUN"`, `"JUL"`, `"AUG"`, `"SEP"`, `"OCT"`, `"NOV"`, `"DEC"`—for example, `"31-AUG-2008 23:59:59"`. For most system date attributes, this should be in the timezone of the machine where the `dmappsrv` is; excepting the revised date, which is in UTC.
- For Oracle databases with English or American NLS settings, the situation is much the same (except the Oracle server machine's timezone is used instead of the `dmappsrv` machine's—this timezone is usually the same, but is a peculiarity of the fact that date translation is done by the database for Oracle, and by the server for other RDBMS).
- For Oracle databases with non-English/American NLS settings, the situation should also be the same, but you may find that for system date attributes the date string needs to be in locale-specific `"DD-MON-YYYY HH24:MI:SS"` format (that is, with the abbreviated month names not being English).

Javadoc

Javadoc is the industry standard for documenting Java classes. The Javadoc for `dmclient` provides full documentation about all the packages, interfaces, and classes, and can be found at the following locations:

Windows `%DM_ROOT%\java_api\docs\api\index.html`

UNIX `${DM_ROOT}/java_api/docs/api/index.html`

Examples

Examples of the important interfaces and classes that a consumer of the API is likely to make frequent use of are located in the Dimensions CM installation as follows:

Windows `%DM_ROOT%\java_api\examples`

UNIX `${DM_ROOT}/java_api/examples`

NOTE You can also use the Dimensions CM CruiseControl plug-in with the Dimensions CM Ant task to implement a continuous integration build based on changes in a Dimensions CM project. For more details, see Part 5 "Integrating Dimensions CM with Build Management Tools" in the *Dimensions CM Build Tools User's Guide*.

Part 4

The Dimensions CM Templating Language

Part 4: Scripts and Templates contains the following chapters:

The Templating Language and Processor	185
Structured Information Return	245
Introduction to Build Templates	249
Build Templates for Distributed Platforms	261
Build Templates for MVS Platforms	275
Openmake Templates	293
Remote Job Execution Templates	299
Deployment Area Scripts	309

Chapter 7

The Templating Language and Processor

Introduction	186
About Templates	186
Directives	188
Special Directives	201
General Predefined Symbols	202
Control Symbols	206
Remote Job Execution Predefined Symbols	207
Dimensions Build Standard Symbols	208
Dimensions Build User-Defined Optional Symbols	212
Dimensions for z/OS Predefined Symbols	217
Template Expansion and Scripts	218
Complex Symbol References	221
Template Inline Functions	221
Testing Templates	226
Adding Template Variables to the Dimensions Configuration File	228
Extending the Template Processor with User Written Functions	230
The Template Processor Interface	243

Introduction

A template is a customizable text file containing variables and control words. In Serena® Dimensions® CM, templates are used to execute builds, execute general commands through remote job execution, control deployments, and to construct the body of e-mail messages used for notifications.

Dimensions CM 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 templates on all Dimensions CM nodes.

A stand-alone utility for testing templates enables you to check the behavior of your templates, for details see page 226.

This chapter describes the templating language and processor. For information about writing different types of templates see the following chapters:

- Build templates on page 249.
- Openmake build templates on page 293.
- Remote execution templates on page 299.
- Deployment area scripts on page 309.
- Email templates, see the *Process Configuration Guide*.

About Templates

Templates are organized into records. Records starting with a `')'` in column one are directives. Symbols are used to name values that are either a single value or an array of one-dimensional tables. Symbol references are expressed by `'%variable.'` where the variable is looked up and substituted. A template is inspected record by record and symbols are replaced and written to the output stream. Dimensions CM constructs a symbol table using the information passed from the commands and environment variables. For each single value a scalar is created, and for each multiple value an array is created. The symbol table represents the context in which template substitution occurs.

Variables can come from the following sources:

- The `)SET` directive in templates.
- Build configurations.
- The `PARAMETERS` qualifier in the `REXEC` command.

Default Template Characters

The template uses the following default characters:

Character	Usage
)	Directive
%	Start substitution
.	End substitution
( )	Used in complex references to control nesting. For example: `%(symt(10)).` references element 11 of the symbol table `symt`. For more details about complex symbol references, see page 221.
:	Used to reference symbol tables nested inside other symbol tables. For example: `%(symt(10):fred)%` references the symbol `fred` nested in element 11 of the symbol table `symt`. For more details about complex symbol references, see page 221.

NOTE You can use the `)ATTR` directive to change the default characters used by the templater, for details see page 188.

Example Input:

```
)SET name=World
Hello %name.
```

Output:

```
Hello World
```

Example Input:

```
)SET A=B
This line uses %A.
)IF %A.=B
The value of A is B.
)ELSE
The value of A is not B.
)ENDIF
```

Output:

```
This line uses B.
The value of A is B.
```

Directives

Directives are commands that are processed by the templater. The templating language includes the directives listed below.

)ADJUST

Description Adjusts the value of the integer `<variable>` by adding the integer `<value>`. You can use this directive to control loops.

Syntax

```
)ADJUST <variable> <value>
```

where:

- `<variable>` specifies a variable you define with the directive `)SET` (see page 198).
- `<value>` specifies an integer value that you use to adjust `<variable>`.

Example Input:

```
)SET TEST=205
)ADJUST TEST -10
The value of TEST is %TEST.
```

Output:

```
The value of TEST is 195
```

)ATTR

Description Changes the default characters used by the template. You must specify a list of exactly six characters in the following order:

- Value #1: Directive character
- Value #2: Start substitution character
- Value #3: End substitution character
- Values #4, #5, and #6: Complex references characters

Syntax

```
)ATTR <value1,value2,value3,value4,value5,value6>
```

Example To change the directive character to hash '#':

```
)ATTR #%.():
```

To change the variable references to look like %var%:

```
)ATTR )%%():
```

To change back to the defaults:

```
)ATTR )%.():
```

NOTE If you use the `)IM` directive to embed a file inside a template you can use `)ATTR` to change that file's defaults. This does not change the default settings of the parent template.

)CALL

See "Additional Template Directives" on page 231.

)CALLBACK

Description Enables you to inject data directly into a template using a programming API.

Syntax `)CALLBACK <ref> <NT>`

where:

- `<ref>` specifies parameters that are delivered to the user's)CALLBACK function.
- `<NT>` specifies that the data provided by the)CALLBACK function is treated as literal. If you do not specify <NT> the data provided by the)CALLBACK function is subject to further expansion by the templater.

Example See the example for `DMEXECENV` on page 212.

)CM

Description Enables you to insert comments into a template.

Syntax `)CM <text-string>`

where:

`<text-string>` is a textual comment.

Example

```
)CM
)CM The following is a single line command!
)CM
```

)CERTLOGON

Description Enables the template processor to log onto Dimensions CM at template processing time without identifiable passwords.

Syntax `)CERTLOGON <certificate>`

where:

`<certificate>` specifies a certificate.

Example `)CERTLOGON EF555456D4ABF9F0A23EB4B309FCE1AF7BD8CC235E443CF81D6B....`

)COMMAND

Description Enables a template to execute any *dmcli* command from the template processor at processing time using a pre-existing session established via one of the following:

-)LOGON
-)CERTLOGON
- `-v` on the `template_test` command.

Syntax `)COMMAND <command>`

where:

`<command>` specifies a Dimensions CM command.

Example `)COMMAND LWSD`

)COPYSYM

Description Transfers SIR variables, via Dimensions CM commands, to the template processor's symbol table. For example, if you specify `s1:s2` the variable `s2` from SIR is inserted into the template symbol table as `s1`.

Syntax `)COPYSYM <varlist>`

where:

`<varlist>` specifies a list of variables or a pair of variables in the format `varto:varfrom`.

Example `)COMMAND CERT1:DMCERTIFICATE CERT2:DMCERTIFICATE2 RC`

)DELETE

For details see page 235.

)DUMP

Description A dump utility to assist in debugging and writing templates. it is available is various guises as explained below.

Syntax `)DUMP [comment [colno]]`

- The form

  ```
  )DUMP
  ```

  dumps the symbol table as a series of I level messages.

- The form

  ```
  )DUMP comment
  ```

  puts the symbol table into the output file from templating using '`comment`' as a start of line comment operator.

- The form

  ```
  )DUMP comment colno
  ```

  interprets '`colno`' as a number (which must be positive) and truncates all output in this form to the specified column number.

- The forms

  ```
  )DUMP )SET [ colno]
  )DUMP SET [ colno]
  ```

  put out the data as a series of `)SET` and `)VECTOR` commands suitable for pasting into a document.

- The form

  ```
  )DUMP "str" [ colno]
  ```

  where "str" contains the string `)SET`, takes the portion of the string up to the `)SET` and prefixes each `)SET` with this comment. This means you can have both `)SET` and comment them out in one operation.

Example

```
)DUMP "//* )SET" 70
```

outputs `//*` in front of each `)SET` or `)VECTOR` and also truncates the line at 70.

)ELSE

Description Switches inclusion on or off in a conditional block.

Syntax `)ELSE`

Example See the examples for `)ENDIF` on page 192.

)ENDIF

Description Ends an)IF conditional block.

Syntax)ENDIF

Operators:

- =
- EQ (equals)
- NE (not equal)
- !=
- STRCONTAINS

Example Input:

```
)SET n=1
%n.
)IF %n. = 1
echo Is one
)ELSE
echo Is not one
)ENDIF
)ADJUST n 1
%n.
)IF %n. = 1
echo Is one
)ELSE
echo Is not one
)ENDIF
```

Output:

```
1
Is one
2
Is not one
```

Example The following example controls when compile options are used:

```
)SET debug=Y
%debug.
)IF %debug.=Y
   <compile options>
)ENDIF
```

Example The following is an example of echoing strings:

Input:

```
)IF Nothing STRCONTAINS oth
echo this line is echoed
)ENDIF
```

Output:

```
echo this line is echoed
```

Example The following is another example of echoing strings:

Input:

```
)SET A=
)IF %A. != ""
This line is actioned
)ENDIF
)IF %A. !=
This line IS actioned
)ENDIF
)IF "%A." != " "
This line is also actioned
)ENDIF
```

Output:

```
This line is actioned
This line is also actioned
```

)ENDEXPAND

Description Ends template expansion started with)EXPAND.

Syntax `)ENDEXPAND`

For more details see page 193.

)ENDR

Description Ends a repeat block

Syntax `)ENDR`

Example See the examples for `)REP` on page 196.

)ENDSCRIPT

See "Additional Template Directives" on page 231.

)EXPAND

Description Expands a template. You can define `)EXPAND` with or without a script:

- **)EXPAND without a script**

  In a simple `)EXPAND` where you do not specify a script, the text block between `)EXPAND` and `)ENDEXPAND` is expanded by the template processor:

```
)EXPAND
<text>
<text>   Expanded by the template processor
<text>
)ENDEXPAND
```

For example, you can use)EXPAND to specify an action that might be confused by the special template characters, or to restrict the sections of a template that are expanded.

- **)EXPAND with a script**

  When you specify a script, the text block between)EXPAND and)ENDEXPAND is processed by the named script processor, and the result is embedded into the template at the point where you insert this directive.

  Syntax:

  ```
  )EXPAND <script> <parameters>
  ```

  where:

  - <script> specifies a processor that will be used to process the script.
  - <parameters> specifies parameters for the script that you specified above.

NOTE This directive is implemented by invoking the system command on the specified file. On z/OS this will cause an address space to be created, and this will exist only for the duration of the script invocation. That operation is expensive and should be avoided if possible.

For more details about using)EXPAND to expand templates, see page 218.

)IF

Description Starts an)IF conditional block.

Syntax

```
)IF <expression>
```

where:

<expression> is the value whose existence you are checking.

Example See the examples for)ENDIF on page 192.

)IFDEF

Description Starts a conditional block based on a symbol's existence.

Syntax

```
)IFDEF <symbol>
```

where <symbol> specifies the symbol whose existence you are checking.

Example

```
)IFDEF debug
The symbol debug is defined
)ENDIF
```

)IFGROUPDEF

For details see page 235.

)IFGROUPNDEF

For details see page 235.

)IFNDEF

Description Starts a conditional block based on a symbol's non-existence.

Syntax

```
)IFNDEF <symbol>
```

where `<symbol>` specifies the symbol whose non-existence you are checking.

Example

```
)IFNDEF debug
The symbol debug is not defined
)ENDIF
```

)IM

Description Embeds a file into the template at the point where you insert this directive. You can also nest `)IM` directives inside other `)IM` directives.

NOTE When you use `)IM` to embed a file you can use `)ATTR` to change that file's defaults. This does not change the default settings of the parent template. For details see page 188.

Syntax

```
)IM <filespec> <NT>
```

where:

- `<filespec>` specifies the file you want to embed.
- `<NT>` specifies that the named template is literal text and is not a template.

Example Input (`outer.tpl`)

```
This is section 1
)SET NAME=John Doe
)IM inner.tpl
This is section 2
This is section 3
)IM inner.tpl NT
This is section 4
```

Input (`inner.tpl`)

```
My name is %NAME.
```

Output

```
This is section 1
My name is John Doe
This is section 2
This is section 3
My name is %NAME.
This is section 4
```

)LOAD

For details see page 235.

)LOGON

Description Log on to the specified Dimensions CM server. This logon will be active until the end of the script.

Syntax)LOGON <user> <pwd> <host> <db> <dsn>

where:

- <user> specifies the user-id.
- <pwd> specifies the user-id's password.
- <host> specifies the server host name.
- <db> specifies the Dimensions CM database.
- <dsn> specifies the database connection string (data source name).

)REP

Description Starts a repeat block that you can use in conjunction with array variables. The directive repeats until it has been round all the elements in the specified array. In the context of REP the array variables named in the)REP statement are available without subscripting for insertion.

The special form %n. expands the nth variable in the immediate)REP without the need to name the variable.

Syntax)REP <var1> <var2> %n.

where:

- <var1> <var2> specifies variables to control and repeat, and for insertion.

Example The example below has a single array variable containing three elements.

Input:

```
)VECTOR dates(3)
)SET dates(0)=12th Nov 2005
)SET dates(1)=13th Nov 2005
)SET dates(2)=14th Nov 2005
)REP dates
%dates.
)ENDR
```

Output:

```
12th Nov 2005
13th Nov 2005
14th Nov 2005
```

Example The example below has two arrays each containing three elements. The second)REP directive (chdocs) is nested inside the first (dates). The output is a list of three dates, each containing a list of all the change documents.

Input:

```
)VECTOR chdocs(3)
)SET chdocs(0)=PVCS_EC_124
)SET chdocs(1)=PVCS_EC_567
)SET chdocs(2)=PVCS_EC_444
)VECTOR dates(3)
)SET dates(0)=12th Nov 2005
)SET dates(1)=13th Nov 2005
)SET dates(2)=14th Nov 2005

)REP dates
%dates.

)REP chdocs
      %chdocs.

)ENDR

)ENDR
```

Output:

```
12th Nov 2005

      PVCS_EC_124
      PVCS_EC_567
      PVCS_EC_444

13th Nov 2005

      PVCS_EC_124
      PVCS_EC_567
      PVCS_EC_444

14th Nov 2005

      PVCS_EC_124
      PVCS_EC_567
      PVCS_EC_444
```

Example The example below has two arrays each containing three elements. The array variables are embedded inside the text. The output is three text strings, each containing a different change document and date.

```
)VECTOR chdocs(3)
)SET chdocs(0)=PVCS_EC_124
)SET chdocs(1)=PVCS_EC_567
)SET chdocs(2)=PVCS_EC_444

)VECTOR dates(3)
)SET dates(0)=12th Nov 2005
)SET dates(1)=13th Nov 2005
)SET dates(2)=14th Nov 2005
```

```
)REP chdocs dates

The change doc %chdocs. is due on the %dates.

)ENDR
```

Output:

```
The change doc PVCS_EC_124 is due on the 12th Nov 2005

The change doc PVCS_EC_567 is due on the 13th Nov 2005

The change doc PVCS_EC_444 is due on the 14th Nov 2005
```

)SAVE

For details see page 235.

)SCRIPT

See "Additional Template Directives" on page 231.

)SET

Description Creates or changes a value in the symbol table.

Syntax

```
)SET var = value
```

where `var` can be a scalar variable name or a reference to an array element when in the form `var[n]`.

Example

```
)SET name=World
Hello %name.
```

)SETL

Description Creates a value in lowercase, or changes an existing value to lowercase.

Syntax

```
)SETL var = <value>
```

where `var` can be a scalar variable name or a reference to an array element when in the form `var[n]`.

Example Input:

```
)SETL name = JOHN
%name.
```

Output:

john

)SET_PATH

Description Treats <value> as a path specification in UNIX format. This value is converted to the equivalent path specification in the local operating-system format, for example:

```
/a/b/c
```

becomes

```
\a\b\c
```

if the local operating-system is Windows, or

```
A.B(C)
```

if the local operating-system is an MVS system.

Syntax `SET_PATH <variable> <value>`

where:

- <variable> specifies the variable to which you are specifying a path specification.
- <value> specifies the path specification.

)SET_PATH_MVS

Description Treats <value> as a path specification in UNIX format. This value is converted to the equivalent path specification in MVS format, for example:

```
/a/b/c
```

becomes

```
A.B(C)
```

Syntax `SET_PATH_MVS <variable> <value>`

where:

- <variable> specifies the variable to which you are specifying a path specification.
- <value> specifies the path specification.

)SET_PATH_NT

Description Treats <value> as a path specification in UNIX format. This value is converted to the equivalent path specification in Windows format, for example:

```
/a/b/c
```

becomes

```
\a\b\c
```

Syntax

```
SET_PATH_NTS <variable> <value>
```

where:

- <variable> specifies the variable to which you are specifying a path specification.
- <value> specifies the path specification.

)SET_PATH_UNIX

Description Treats <value> as a path specification in Unix format, this value is converted to the equivalent path specification in UNIX format (noop), for example:

```
/a/b/c
```

becomes

```
/a/b/c
```

Syntax

```
SET_PATH_UNIX <variable> <value>
```

where:

- <variable> specifies the variable to which you are specifying a path specification.
- <value> specifies the path specification.

)SETU

Description Creates a value in uppercase, or changes an existing value to uppercase.

Syntax

```
)SETU var = value
```

where var can be a scalar variable name or a reference to an array element when in the form var[n].

Example Input:

```
)SETU name = john
%name.
```

Output:

JOHN

)SLICE

Description Enables you to send arbitrary quantities of data to a program from a single string variable. The directive creates an array in `var2` that is long enough to hold all the information in `var1` in 32 byte chunks. `var1` must already exist.

Syntax

```
)SLICE var1 var2
```

Example Input:

```
)SET FOOTPRNT=THIS IS A VERY LONG FOOTPRINT INFORMATIONAL AREA
)SLICE FOOTPRNT F
)REP F
          CL32'%F.'
)ENDR
```

Output:

```
          CL32'THIS IS A VERY LONG FOOTPRINT IN'
          CL32'FORMATIONAL AREA'
```

)VECTOR

Description Creates an array of elements.

Syntax

```
)VECTOR var(n)
```

where `var` defines an array variable with `n` occurrences.

Example

```
)VECTOR dates(3)
)SET dates(0)=12th Nov 2005
)SET dates(1)=13th Nov 2005
)SET dates(2)=14th Nov 2005
```

Special Directives

The directives listed below enable a template processing step to pass data to a later processing step:

-)SAVE
-)LOAD
-)DELETE
-)IFGROUPDEF
-)IFGROUPNDEF

See "Template Processing: Passing Data Between Steps" on page 235 for details.

General Predefined Symbols

The templating language includes general predefined symbols.

DMCERTIFICATE

Description Generates a certificate and credentials for an area. Use with the BUILD_OPTIONS qualifier in the BLD and BLDB commands:

```
BLD/BLDB /BUILD_OPTION=(DMCERTIFICATE=GENERATE,...)
BLD/BLDB /BUILD_OPTIONS=(DMCERTIFICATE=GENERATEWITHAUTH,...)
```

where:

- `GENERATE`

  Generates a certificate and credentials for an area to allow a user to log back into Dimensions.

- `GENERATEWITHAUTH`

  Generates a certificate and credentials for a Dimensions login, and an `AUTH` command, to connect automatically back to the tertiary node hosting the area.

The values of `DMCERTIFICATE` are hex-encoded in a certificate, not the text "`GENERATE`" or "`GENERATEWITHAUTH`".

Input or output Input

DMCURRENTDIRECTORY

Description The current directory.

Input or output Input

DMCYGWINCWD

Description (Windows only) The current directory in Cygwin (Windows UNIX utilities) style:

```
'.'
```

Input or output Input

DMDAY

Description Two digit string representing the day in the month when this template process is started.

Input or output Input

Example

```
07s
```

DMFOLDERSEPARATOR

Description Folder separator:

- UNIX: `'/'` (forward slash)
- Windows: `'\'` (backward slash)

Input or output Input

DMHOUR

Description Two digit string representing the hour when this template process is started. Uses GMT in a 24-hour format.

Input or output Input

Example `12`

DMMICROSEC

Description Six digit string representing the microsecond when this template process is started.

Input or output Input

Example `01348`

DMMINUTE

Description Two digit string representing the minute when this template process is started.

Input or output Input

Example `34`

DMMONTH

Description Two digit string representing the month in the year when this template process is started.

Input or output Input

Example `12`

DMOSTDERR

Description Returns the filename containing the errors of the template execution. This file exists on the machine where the script is executing.

Input or output Output

DMOSTDOUT

Description Returns the filename containing the output of the template execution. This file exists on the machine where the script is executing.

Input or output Output

DMOSTYPE

Description The operating system being used: UNIX or Windows

Input or output Input

DMOXTMPLT

Description Returns the filename containing the expanded template. This file exists on the machine where the script is executing.

Input or output Output

DMPASSWORD

Description Set to the value of the node password specified for the work or deployment area. This password can be different to the one you used to log on with.

Input or output Input

DMPATHSEPARATOR

Description The path separator for the operating system being used.:

- UNIX: `':'` (colon)
- Windows: `';'` (semi-colon)

Input or output Input

DMSECOND

Description Two digit string representing the second when this template process is started.

Input or output Input

Example `24`

DMUNIQUE

Description An eight character encoding of the date and time to within one second in the century, formatted to conform to MVS member naming standards and lexically ordered by time.

Input or output Input

Example Input:

```
//INPUT DD DSN=BLD.%DMUNIQUE..COBOL,DISP=SHR
```

Output:

```
//INPUT DD DSN=BLD.A3664E3.COBOL,DISP=SHR
```

Input or output Input

DMUSER

Description Set to the value of the node user ID specified for the work or deployment area. This user ID can be different to the one you used to log on with.

Input or output Input

DMUSERU

Description Same as for `DMUSER`, but truncated to seven characters and converted to upper case.

Input or output Input

DMYEAR

Description Four digit string representing the year when this template process is started.

Input or output Input

Example `2005`

Control Symbols

The templating language includes optional symbols to control execution.

DMSAVESTDERR

Description Saves the errors from the template execution. The filename where the errors are saved is placed in the variable DMOSTDERR. You can set this option when:

- Running the REXEC command with the /PARAMETERS qualifier:

  ```
  /PARAMETERS=(DMSAVESTDERR=boolean,,..)
  ```

- Running the templating testing program with the -g flag:

  ```
  template_test -g e
  ```

- Setting up build configurations.
- Writing a template, for example add the line:

  ```
  )SET DMSAVESTDERR=Y
  ```

Input or output Input

DMSAVESTDOUT

Description Saves the output of the template execution. The filename where the output is saved is placed in the variable DMOSTDOUT. You can set this option when:

- Running the REXEC command with the /PARAMETERS qualifier:

  ```
  /PARAMETERS=(DMSAVESTDOUT=boolean,,..)
  ```

- Running the templating testing program with the -g flag:

  ```
  template_test -g s
  ```

- Setting up build configurations.
- Writing a template, for example add the line:

  ```
  )SET DMSAVESTDOUT=Y
  ```

Input or output Input

DMCOMBINEOUTERR

Description Similar to DMSAVESTDERR and DMSAVESTDOUT but combines the output and the errors into one file and places them in the variable DMOSTDOUT.

You can set this option when:

- Running the REXEC command with the /PARAMETERS qualifier, for example:

  ```
  /PARAMETERS=(v1=val1, v2=val2...)
  /PARAMETERS=(v1=[v1,v2...],v2=[v3,v4...])
  ```

- Running the templating testing program with the -g flag:

  ```
  template_test -g c
  ```

- Setting up build configurations.
- Writing a template, for example add the line:

  ```
  )SET DMCOMBINEOUTERR=Y
  ```

Input or output Input

Remote Job Execution Predefined Symbols

The templating language includes remote job execution predefined symbols.

DMCERTIFICATE

Specifies a one-time certificate used to reconnect to Dimensions CM to perform post-build processing such as updating a job status or collecting build outputs.

DMJOBID

Specifies the remote job key.

Dimensions Build Standard Symbols

The templating language includes the Dimensions Build standard (permanently available) symbols described below. You can use these symbols in a build template or specify them in the Build Options section of a build configuration. For more details about Dimensions Build and the Primary Build Execution Monitor (pBem) see the *Dimensions CM Build Tools User's Guide*.

NOTE There is also a set of optional Dimensions Build symbols described in "Dimensions Build User-Defined Optional Symbols" on page 212. To be utilized, these optional symbols must be explicitly added as Dimensions Build options, that is, they are not defined by default.

DMPATH

Description

DMPATH tells the build step template where the source is to be found. It always contains at least one element, namely, the disk location where the build is being performed. This disk location element will always be the first element in the array.

For deployment area builds, further elements of the DMPATH array are formed by examining all deployment areas for the build configuration that are:

- firstly, located on the same node as the area being built, and
- secondly, for which the GLC sequence is higher than or equal to that of the area being built.

On MVS systems, this is the stem of the area with // preceding each entry.

For distributed systems, it is an absolute path to the area.

If the area being built is a work area, and if the DM_SP_START_STAGE symbol is set to a GLC identifier, the first array entry will be the location of the work area and the remainder of the DMPATH variable is constructed as though the work area were a deployment area at the specified stage.

NOTE On MVS platforms the data set stem does not contain low level qualifiers, which are added by the template.

MVS format:

```
//ACCOUNT.TEST
//ACCOUNT.SYSTEM
//ACCOUNT.PRODUCTION
```

Distributed format:

NOTE There is usually only one useful row in Windows and UNIX distributed environments, namely, the first.

- Windows:

```
c:\account\test
c:\account\system
```

```
d:\account\production
```

- UNIX:

```
/account/test
/account/system
/account/production
```

z/OS Example

Input:

```
)VECTOR DMPATH(3)
)SET DMPATH(0)=//BUILD.LEVEL1
)SET DMPATH(1)=//BUILD.LEVEL2
)SET DMPATH(2)=//BUILD.LEVEL3

)SET FIRST=SYSIN
)REP DMPATH
)SET THSDIR=_MDHEXTRACT_(+d,%DMPATH)
//%FIRST.       DD DSN=%DMPATH..COBOL,DISP=SHR
)SET FIRST=
)ENDR
```

This mimics the set up done by the pBem.

Output:

```
//SYSIN          DD DSN=BUILD.LEVEL1.COBOL,DISP=SHR
//               DD DSN=BUILD.LEVEL2.COBOL,DISP=SHR
//               DD DSN=BUILD.LEVEL3.COBOL,DISP=SHR
```

Distributed Example

Input:

```
)VECTOR DMPATH(3)
)SET DMPATH(0)=c:\stu\my-area
)SET DMPATH(1)=d:\build\system-test
)SET DMPATH(2)=e:\build\production

)SET INCLUDES=

)REP DMPATH
)SET INCLUDES=%INCLUDES. -I %DMPATH.
)ENDR

compile foo.c %INCLUDES.
```

This mimics the set up done by the pBem.

Output:

```
compile foo.c -I c:\stu\my-area -I d:\build\system-test -I
    e:\build\production
```

DMINPUT

Description Specifies an array of sources expressed as relative filenames, where each row names an input file. Each file is categorized according to the filename extension (on distributed platforms) or the low level qualifier (on mainframe platforms). The file type decides where the input goes in the template. See also "MDHDSN" on page 221.

NOTE If a template only reads in one file the category is irrelevant as there is only one place where the information can go.

Distributed examples

- `prog1.c`
- `account\account1.c` (Windows)
- `account/account1.c` (UNIX)

MVS examples

- `COBOL(PROG1)`
- `ACCOUNT.COBOL(ACCT1)`

DMTARGET

Description Specifies an array of relative filenames that are the outputs from a build. Each file is categorized according to the filename extension (on distributed platforms) or the low level qualifier (on mainframe platforms).

Distributed examples

- `prog1.c`
- `account\account1.c` (Windows)
- `account/account1.c` (UNIX)

MVS examples

- `COBOL(PROG1)`
- `ACCOUNT.COBOL(ACCT1)`

DMSTEP

This is constructed by Dimensions Build. It specifies the sequence number of the current step in the build process and can be used in a build template to signal to the pBem the end of a step. For more details about Dimensions Build see the *Dimensions CM Build Tools User's Guide*.

DMRUNMODE

Description This is set by pBem. It is a read only symbol that conveys the actual runmode to the template. It can be used to trigger asynchronous specific actions.

Example

```
)IF %DMRUNMODE. = ASYNC
...
)ENDIF
```

DMUSER and DMPASS

Set these symbols from a build template to change the user ID that a build step runs as. This is different to the DMUSER and DMPASSWORD symbols on the original REXEC request that started the pBem.

DMSERVER

This is set by the Dimensions Build server. It is a string such as <MACHINE>:<PORT> that can be used as a network address to locate the build server when sending messages. It is used when messages are sent directly to the build server. To send messages back to the controlling pBem use the variables DMPBEMNODE and DMPBEMPORT (see below).

DMPBEMNODE and DMPBEMPORT

This is set by pBem. It specifies the network address and port of the controlling pBem process. Asynchronous tasks need to send messages back to this address to alert the pBem of their status, but this is done automatically for MVS by SBEM.

DMTOKEN

This is constructed by the Dimensions Build server. It is an abstract build token that is used internally by the build server to create a Bill of Materials report (BOM).

DMJOBID

This is constructed by the Dimensions listener. It is an abstract symbol generated by the REXEC mechanism to track jobs. In Dimensions 9.x, the RSTAT and RLIST commands used this symbol as the primary build tracking mechanism. This symbol has been retained for backwards compatibility though the Dimensions Build server now manages the build history.

DMXFERSOURCE

Set this symbol to 'Y' to transfer sources automatically to work areas. By default Dimensions, does not transfer sources to work areas.

DMBUILDJOB

Passes the job number to an rexec task that needs to wait until the originating in-flight job has completed.

Dimensions Build User-Defined Optional Symbols

The templating language includes the Dimensions Build optional (not defined by default) symbols described below. You can use these symbols in a build template or specify them in the Build Options section of a build configuration. For more details about Dimensions Build and the Primary Build Execution Monitor (`pBem`) see the *Dimensions CM Build Tools User's Guide*.

NOTE There is also a set of standard Dimensions Build symbols described in "Dimensions Build Standard Symbols" on page 208. These predefined symbols are permanently available to be utilized as Dimensions Build options, that is, they are defined by default.

DMEXECENV

Description Sets one of three modes that control the script execution environment after a template has been expanded. It is user-defined in the template.

Mode 1 **Process as an MVS batch job**

In this mode the template is processed as an MVS batch job and is submitted to be run. The script execution environment treats the template as JCL and submits it.

Syntax:

```
)SET DMEXECENV=JCL or )SET DMEXECENV=JES
```

Mode 2 **Process with a shell script**

In this mode a command, which references the expanded template, is constructed from the string following the SHELL keyword and is sent to the 'system' command processor. Redirection of `stdout` and `stderr` is post-pended automatically. The position of the expanded template name is indicated by the use of `%s`. If default template symbol expansion is used, `%` must be doubled in the usual way.

The redirection processing is managed by the variables `DMCOMBINESTDOUTERR`, `DMSAVESTDOUT`, `DMSAVESTDERR`, `DMSTDOUT` and `DMSTDERR`. The settings of these variables is controlled by the wrapping application:

- `REXEC /PRESERVE and /SHOW`.
- For deployment scripts there are variables you can define at an area level to control script output management.
- When building, the template processor observes the build template settings and preserves the related outputs.

Syntax:

```
)SET DMEXECENV=SHELL <command template>
```

Example 1:

```
)SET DMEXECENV=SHELL /bin/bash %%s
```

The script is processed using the bash interpreter.

Example 2:

```
)SET DMEXECENV=SHELL %%s
```

The system command looks at the first two bytes of the expanded template and if this is #! uses the processor named in the script (shebang processing).

Example 3:

```
)SET DMEXECENV=SHELL perl %%s
```

Mode3 Process with a different template

In this mode the expanded template is processed again by the templater using a different template that you specify. This allows a specific template to be wrapped by another.

Syntax:

```
DMEXECENV=TEMPLATE <template name>
```

Example:

In the example below the)CALLBACK directive indicates where to insert the text from the inner template.

Input (outer.tpl):

```
)EXPAND
echo off
echo Building started
)CALLBACK NT
echo Building continuing
```

Input (inner.tpl):

```
)SET DMEXECENV=TEMPLATE output.tpl
echo building.....
echo building.....
echo building.....
```

Output:

```
Building started ─ From outer.tpl
building.....
building.....  ─ From inner.tpl
building.....
Building continuing ─ From outer.tpl
```

DMEXPAND

This is user-defined in the template.

If DMEXPAND:

- Is set to any value:

  EXPAND mode is enabled, where only the parts of the template delimited by)EXPAND and)ENDEXPAND are expanded.

- Does not exist:

  The template behaves as in Dimensions 9.x, where the entire template is expanded.

For examples see "Template Expansion and Scripts" on page 218.

DMTASKNAME

Description This is used to create a message that is displayed in the Task column of the Build Monitor Events section of the Build Job Details dialog box in Dimensions Build. You can also use other symbols, for example, to insert program names.

Example

```
)SET DMTASKNAME=$%DMSTEP, Compiling program %SOURCE.
```

DMTIMEOUT

This is set to override the time (in seconds) that the pBem waits for a step to complete before it is marked as failed. Note that Dimensions Build also has an overall build timeout value, which might be wait forever.

DMSTEPMODE

Description This is set to override the synchronous and asynchronous setting of a build for a given step; usually used to ensure that certain activities are synchronous.

Values: ASYNC or SYNC

Example

```
)SET DMSTEPMODE=SYNC
```

DMMAXRC

This is set in a build configuration or for doing a run to specify the highest acceptable return code set by build, step, area, or config. It enables a template to pass back the actual return code for information and allow certain values to be treated only as warnings. Set this symbol to the maximum value that is considered safe for that step. If the return code is higher than this value the pBem will not schedule dependent steps as the step will have failed.

DMNOCACHE

This symbol is set to `'Y'` for a template or build configuration to disable the caching of some file information. It produces a slower but more accurate build for poorly defined build configurations.

DMREBUILDALL

Controls PBEM step level optimization. Possible values:

- `NO`: (Default) rebuilds are only performed for targets that have no dependencies or where there are dependencies that are newer than the target.
- YES or ALL: all identified targets are rebuilt unconditionally.
- CURRENT: only targets that already exist in the area being used for the build are rebuilt.

DMMUSTRUN

Define this symbol in a template or build configuration to indicate that a step is essential and must run even if a prior dependent step has failed. Note that cleanup activities should be done in post-build steps defined in the build configuration.

DMFINAL

Define this symbol for a template or build configuration to indicate that a step should run last. Forces dependencies to be created to ensure that the step does run last. Note that it is best practice to use a correctly configured build configuration to specify when steps are run.

DMTTLOG

Causes detailed trace messages from the templating process to be produced, their destination depending on the context in which they are generated. These messages can be useful in some error situations to determine why a template failed.

Set this debug flag to `'Y'` in `REXEC /PARAMETERS`, `BLD /BUILD_OPTIONS`, `BLDB /BUILD_OPTIONS`, or the Dimensions CM configuration file. at the global, area, or step level

DMALTSERVER

Enables you to change the address of the build server that is defined in the BRD file—typically the server from where the build is launched—and specify the address of an alternative build server.

Must be in the following format:

```
http://<IP address of the machine hosting the build server>:<Tomcat port number>/bws/
services/monitor
```

Example `http://<dimensions-dev>:<8080>/bws/services/monitor`

Usage scenario When you launch a build from a virtual machine, and are building on a remote node, the address of the build server in the BRD file on the remote node may be the physical machine hosting the virtual machine (not the actual virtual machine). You can use DMALTSERVER to change the address to that of the virtual machine.

DM_SP_START_STAGE

When you build in a work area you can limit a search path by specifying the stage in the Global Stage Lifecycle (GSL) from where you want to start appending deployment area paths to search paths in a BRD file. Possible values:

- (Default) Empty (no value): include all stages.
- NONE: do not use any search path except the target work area.
- <GSL stage name>: include all work areas on the node at, or above, this stage.

DM_BUILDER_PROGRESS_REPORTING

If you specify this symbol, any step that has a non-zero return code causes the SBEM to issue the following message with a severity of W:

```
MDHSB16D112W Step <step number> Program <program> returned COND CODE = <cc>
```

Default: YES

Dimensions for z/OS Predefined Symbols

User ID and Account Predefined Symbols

Use the Dimensions CM symbols listed below on z/OS machines to display the security and execution environment.

Symbol Name	Usage	Example
DMSYSUSR	RACF user ID	MERSTU1
DMSYSDIR	USS home directory	/u/home/merstu1
DMSYSSHL	User's shell	/bin/sh
DMSYSUID	UID of the UNIX process (integers)	23587
DMSYSGID	GID of the UNIX process (integers)	54968695

Job Sequence Number Predefined Symbols

Dimensions for z/OS maintains a sequence number that increments by one every time you submit a template. You can use this number in the job name, or any other part of a template, to distinguish one job from another.

To allow the sequence number to be used in restricted environment, such as an eight character job name, there are various number formats that you can use. For example, if you add DMSEQA to the end of a job name when the user ID is up to seven characters long, a maximum of 36 different job names will be created.

The sequence number is made visible to the JCL tailoring skeleton through the following predefined symbols:

Symbol Name	Usage	Example
DMSEQNNN	Plain number, left justified, variable width.	98
DMSEQ999	Plain number, fixed width, three digits.	098
DMSEQ99	Plain number, fixed width, two digits.	98
DMSEQ9	Plain number, single digit.	8
DMSEQAAAA	Alpha-numeric, variable width.	002N
DMSEQAAA	Alpha-numeric, fixed width, three digits.	02N
DMSEQAA	Alpha-numeric, fixed width, two digits.	2N
DMSEQA	Alpha-numeric, single digit.	N

Maintaining Sequence Numbers

Sequence numbers are maintained on the remote node where the template is executed and not by the Dimensions CM server. If you execute templates on multiple machines, each machine has its own sequence numbers. The sequence number is maintained in an HFS file inside a directory specified in the Dimensions CM configuration symbol DM_TPLB_SEQUENCE_PATH (see page 228).

Template Expansion and Scripts

In the version of the templating language released with Dimensions 9.x, the entire template was expanded. In the current version, when you specify the symbol DMEXPAND, only the lines between the directives)EXPAND and)ENDEXPAND are treated as template lines, the rest being treated as literal text. For more details about using these directives see page 193.

There are two ways that you can implement scripts in a template:

- **Expansion—**use a script, delimited by the directives)EXPAND and)ENDEXPAND, to expand a specific part of a template to complete it, for example, to insert a date.
- **Execution**—use a script to execute an entire expanded template, for example, to call a compiler and run the expanded template through it.

A script can be any shell scripting language, typically REXX on a mainframe, and Perl, Python or any other command-line language on a distributed platform. Script mode allows you to generate complex scripts and to interpret the body of the expanded section as a command script, the output of which becomes the expanded text.

NOTE The)EXPAND directive is implemented by invoking the system command on the specified file. On z/OS this will cause an address space to be created, and this will exist only for the duration of the script invocation. That operation is expensive and should be avoided if possible.

Example The example below illustrates one simple use of template expansion.

Input:

```
)EXPAND %%s
echo echo Hello
)ENDEXPAND
```

Output:

When the script runs, it outputs "echo Hello". So these three lines will be replaced by one line containing "echo Hello". This command will execute when the template runs, printing "hello".

Example The example below illustrates another simple use of template expansion.

Input:

```
)EXPAND %%s
echo Hello > c:\temp\foo.txt
)ENDEXPAND
```

Output:

This particular script does not output anything, the three lines simply disappearing from the template. However, during execution, the file named (foo.txt) will get created.

Example The mainframe example below demonstrates the power of an external scripting language. The template does the following:

- Creates a temporary file, referred to by `'%%s'`, containing the script delimited by `)EXPAND` and `)ENDEXPAND`.
- Executes the temporary file with the REXX interpreter. REXX outputs the string `'job submitted on <date>'`. This line then becomes part of the template.

Input:

```
//*
)EXPAND %%s
/*REXX*/
Say "//* job submitted on" date()
exit 0
)ENDEXPAND
)EXPAND
//*
```

Output:

```
job submitted on 12th December 2005
```

NOTE Two percent characters '%%' are required if `'%'` is being used as a special character. The role of `'%s'` is to allow a command line to be generated where the script name is not necessarily the last parameter.

Example The mainframe example below shows how to pass variables into and out of an executed script, where REXX is used to set a template variable called MYVAR.

Input:

```
//*
)EXPAND %%s
/*REXX*/
Say ")EXPAND"
Say ")SET MYVAR=" date()
Say "//* job submitted on" date()
)ENDEXPAND
)EXPAND
//*
//* %MYVAR.
//*
```

This)EXPAND starts the block that ends with)ENDEXPAND. As it is followed by a script, this expansion is in script mode.

This)EXPAND forces the template to treat the output from REXX as template lines (the default for an expanded script is not to expand).

REXX outputs this line, which is interpreted by the templater, and causes the variable MYVAR to be set.

This)EXPAND forces the variable MYVAR to be expanded.

Output:

```
//*
//* job submitted on 8 Dec 2005
//*
//* 8 Dec 2005
//*
```

Example The Perl distributed platform example below is a template with a nested Perl script that splits out the PATH environment variable into a template array.

Input:

```
)VECTOR MYPATH(20)

)EXPAND perl
    %%s
```

This)EXPAND starts the block that ends with)ENDEXPAND. As it is followed by a script, this expansion is in script mode.

```
        print ")EXPAND\n";
```

This)EXPAND forces the template to treat the output from Perl as template lines.

```
        @dirs = split(/:/,$ENV{'PATH'});
        $i=0;
        foreach $dir (@dirs)
                {
                print ")SET MYPATH($i) = $dir\n";
                $i=$i+1;
                }
)ENDEXPAND
)EXPAND
```

This)EXPAND forces the variable 'MYPATH' to be expanded.

```
//*
//* First element = %(MYPATH(0)).
//* Second element= %(MYPATH(1)).
//*
//* This is the complete array:
//*
)REP MYPATH
===> %MYPATH.
)ENDR
//*
//* End
//*
```

Output:

```
//*
//* First element = /cygdrive/c/stu/exec
//* Second element= /cygdrive/c/WINDOWS/system32
//*
//* This is the complete array:
//*
===> /cygdrive/c/stu/exec
===> /cygdrive/c/WINDOWS/system32
===> /cygdrive/c/WINDOWS
===> /cygdrive/c/WINDOWS/System32/Wbem
===> /cygdrive/c/Oracle/jre/1.4.2/bin/client
===> /cygdrive/c/Oracle/jre/1.4.2/bin
===> /cygdrive/c/Oracle/bin
===> /cygdrive/c/unix_utils
===> /cygdrive/c/jakarta-ant-1.5/bin
===> /cygdrive/c/j2sdk1.4.2_10/bin
===> /cygdrive/c/Serena/Dimensions/10.1/prog
===> /cygdrive/c/Program Files/Serena/License Manager
===> /usr/bin
===> /cygdrive/c/Program Files/QuickTime/QTSystem/
//*
//* End
//*
```

Complex Symbol References

A simple variable reference is `'%var.'` where `%` can be changed by `)ATTR`. A complex symbol reference has an extra set of parentheses and is of the form `'%(var).'`. For example, `'%(date).'` is a complex reference to the variable `date`. The most common use of complex references is to access arrays in build templates.

The example below accesses the third row of the array DMPATH (arrays start at zero therefore (2) is the third row).

```
%(DMPATH(2)).
```

In the example below the index to the array is the variable `index`.

```
%(DMPATH(index)).
```

NOTE When a variable appears in parentheses inside a complex reference, you do not need to prefix the variable with `%`.

Template Inline Functions

Overview

Template inline functions enable you to add functionality to your templates that is not available with the standard template directives. Inline functions are processed after symbols are expanded but before template directives are executed. The text string returned by an inline function is inserted into a template at the point where the function is named.

Syntax:

```
_function name_(parameters)_
```

MDHDSN

Description `MDHDSN` is a C function that helps you to manage search paths in templates in JCL and in distributed scripts.

Syntax

```
_MDHDSN_(default-name,lookup-array,type,path-array,error_flag)_
```

where:

- `default-name` specifies a name if you do not specify `'type'`.
- (Optional) `lookup-array` specifies an array of objects.
- (Optional) `type` specifies a file or object type.
- (Optional) `path-array` specifies a search path.
- (Optional) `error-flag` is a single character specification that controls the issuing of error messages:

The error-flag options are:

0 or S - (silent)	Do not issue a message at all.
W	Issue a warning only.
I	Issue an information message only.
T	Issue a trace message only.

This is only relevant to two messages, MDHPBO00003E and MDHPBO00004E.

Usage MDHDSN has four steps:

- **Step one**: optionally specify a 'type' and a 'lookup-array' to find a name specified by your build configuration. For example, you can search for a 'c' type file in a specific build array.
- **Step two**: if you do not specify a 'lookup-array' or 'type', or if there is no match, the function searches for the name that you specify with 'default-name'.
- **Step three**: optionally specify a 'path-array' to locate an item in a search path. Returns the full path name without the initial '//'. The template fails if a file is not found in the search path.
- **Step four**: optionally specify an 'error-flag' character to control the issuing of an error message on error detection. This allows a script to be written that will not flag a build error in the case that the file is not present. If no error-flag' character is specified, an 'E' level error message will be issued on the detection of an error.

Example The following directives create two arrays that are used by the inline functions in this example:

```
)VECTOR DMPATH(5)
)SET DMPATH(0) = /zero/
)SET DMPATH(1) = /one/
)SET DMPATH(2) = /two/
)SET DMPATH(3) = /three/
)SET DMPATH(4) = /four/

)VECTOR DMSOURCE(3)
)SET DMSOURCE(0) = foo.c
)SET DMSOURCE(1) = bar.input
)SET DMSOURCE(2) = middle/blim.txt
```

The following function returns hello.c (the default) as no other parameters are specified:

```
_MDHDSN_(hello.c,,,)_
```

The following function searches for test in the array DMSOURCE and returns the default (hello.c) as there is no match:

```
_MDHDSN_(hello.c,DMSOURCE,test,)_
```

The following function searches for a 'c' type in the array DMSOURCE and returns 'foo.c':

```
_MDHDSN_(,DMSOURCE,c,)_
```

The following function searches for 'foo.c' in the array DMPATH and returns '/two/foo.c':

```
_MDHDSN_(foo.c,,,DMPATH)_
```

The following function combines two operations by searching for a 'c' type in the arrays DMSOURCE and DMPATH, and returns '/two/foo.c':

```
_MDHDSN_(hello.c,DMSOURCE,c,DMPATH)_
```

Example The following directives find an entry in the array 'S', which has a type of 'b'. This matches f2.b. If this had not worked, it would use the default of default.txt. Then, using this filename (f2.b), it would look for it on disk in all the locations in the array 'A', until found. If this does not work, it specifies not to issue an error message.

```
)VECTOR A(4)
)SET A(0)=d:\stu\c\tpl\1\
)SET A(1)=d:\stu\c\tpl\2\
)SET A(2)=d:\stu\c\tpl\3\
)SET A(3)=d:\stu\c\tpl\4\
)CM
)CM
)CM
)VECTOR S(3)
)SET S(0)=f1.a
)SET S(1)=f2.b
)SET S(2)=f3.c
)CM
)CM
)CM
%(A(0)).
%(A(1)).
%(A(2)).
%(A(3)).
%(S(0)).
%(S(1)).
%(S(2)).
)CM
found _MDHDSN_(default.txt,S,b,A,0)_
```

Example The following example shows how the function works with an MVS JCL template:

Input:

```
//SYSIN DD DSN=_MDHDSN_(COBOL(FOO),DMSOURCE,COBOL,DMPATH)_,DISP=SHR
```

Output:

```
//SYSIN DD DSN=SMITCHE.BLD.UNIT.COBOL(FOO),DISP=SHR
```

MDHEXTRACT

Description MDHEXTRACT explodes filenames and data set names into component parts.

Syntax `_MDHEXTRACT_(format,filename,[root])_`

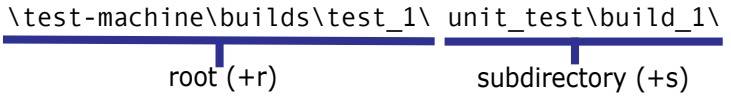
where:

- `format` is a literal string that can optionally contain one or more of the following operators:

Operator	Description
+p	Returns the filename prefix, usually `'null'` on distributed platforms and `'//'` on MVS platforms.
+r	Returns the root of the filename with a trailing directory character. Distributed example: `\test-machine\builds\test_1\` MVS example: `//test-machine.builds.test_1.` See the explanation below this table about specifying a root.
+n	Returns the name of the object, for example, `foo`.
+t	Returns the type of the object, for example, c or cobol.
+s	Returns the subdirectory with a trailing directory character. Distributed example: `unit_test\build_1\` MVS example: `unit_test.build_1.` See the explanation below this table about specifying a root.

+d	Distributed platforms: returns the entire path and filename, for example:
	`\test-machine\builds\test_1\unit_test\build_1\foo.c`
	MVS platforms: returns the full data set name without the initial '//', for example:
	`test-machine.builds.test_1.unit_test.build_1.cobol(foo)`

- `filename` specifies the filenames or data set name that you want to explode.
- `[root]` specifies the part of the filename or data set name that is the root. For example:



Example This example returns the name of the object.

Input:

```
_MDHEXTRACT_(+n,/root/test/fred.txt,)_
```

Output:

```
fred
```

Example This example returns the type of the object.

Input:

```
_MDHEXTRACT_(+n,//'MDH.DEV.UT.THING.C(FOO)',)_
```

Output:

```
FOO
```

Example This example specifies the root, and returns the subdirectory, the type, and the name of the object.

Input:

```
_MDHEXTRACT_(subdir(+s) type(+t) name=+n,
    //'MDH.DEV.UT.TEST.C(FOO)',MDH.DEV.UT)_
```

Output:

```
subdir(TEST.) type(C) name=FOO
```

Example This example uses MVS data set names.

Input:

```
_MDHEXTRACT_(t(+t) name(+n),//'MDH.DEV.UT.TEST.TEXT',)_
```

Output:

```
t(TEXT) name(TEST)
```

NOTE It is common for problems to occur with different kinds of slashes appearing in filenames, particularly when the names are computed from parts. You can use a particular form of the MDHEXTRACT function to normalize names, according to the current platform. On Windows backslashes are used ("\") and on UNIX forward slashes are used ("/") in the output. For example:

```
)SET NEWNAME=_MDHEXTRACT_(+d,%OLDNAME.)_
```

substring

Description This function reads nothing and writes back nothing, but the first variable V1 is set to the substring of V2 specified by the spos and len values. If spos is negative, the start position is the specified number of characters from the end.

Syntax

```
)CALL substring V1 V2 spos len
```

where:

- spos is optional and defaults to 0.
- len is optional and defaults to -1 (meaning the rest of the string).

Example Input:

```
)SET V2=The Quick Brown Fox
)CALL substring V1 V2 0 2
             %V1.Extensibility
```

Output: Th

Input:

```
)SET V2=The Quick Brown Fox
)CALL substring V1 V2 -3 1
%V1.
```

Output: F

pos

Description This function creates the variable V3 and sets its value to the result of the operation. The variable VMAIN is searched for by the substring VSUB. If there is no match, V3 is set to the string -1. Otherwise it is set to the string representing the decimal number equal to the starting offset of the substring.

Syntax)CALL pos V3 VMAIN VSUB

Example Input:

```
)SET VMAIN=This is mine now
)SET VSUB=mine
)CALL pos V1 VMAIN VSUB
```

Output: V1 contains the value 8.

Input:

```
)SET VMAIN=This is mine now
)SET VSUB=this
)CALL pos V1 VMAIN VSUB
```

Output: V1 contains the value 0.

Testing Templates

For a Dimensions CM server or agent installation, a template testing program called `template test` is located in the 'prog' subdirectory of the Dimensions CM root installation directory. This template testing program enables you to unit test your own templates. It is command-line driven and invokes the template processor on a single input file for processing.

Command-line

```
template_test <input file> <switches>
```

Switches

Switch	Default	Description
-?		Displays help.
-b dir	NULL	Specifies the base directory for templates.
-c command	n/a	Specifies a command string.
-d value		Specifies a value.
-e dir	NULL	Specifies an execution directory.

Switch	Default	Description
`-g redir`	None	Sets redirection options. Can be one or more of s, e, or c. ■ s: stdout is directed to a temporary file. ■ e: stderr is directed to a temporary file. ■ c: outputs are combined into one temporary file.
`-h <headerspec>`		Includes the specified header before processing the template.
`-i`		Traces storage allocates in template_test pool.
`-j`	False	Executes under JES.
`-k`		Creates a shared pool (SAVE/DELETE/LOAD).
`-l <logspec>`	NULL	Sends all messages to the log file you specify.
`-m <messagespec>`		Specifies an output for diagnostic/terminal output.
`-o <outputspec>`	NULL	Spools the output to the file that you specify instead of passing the template to the system for execution.
`-p <password>`	dmsys	Sets the password to the value that you specify before calling the routine.
`-r`	False	Reports on storage allocation and deallocation requests.
`-s`		Use service provider (_MDHDSN_ etc)
`-t`	False	If set to true, reports all messages to stdout.
`-u <user id>`	dmsys	Sets the user id to the value that you specify before calling the service.
`-v <certificate>`		Runs the template processor against a script and logs into Dimensions CM using the specified certificate. Can be used as an alternative to *dmcli*.
`-w`		Waits for execution.
`-x`	False	Passes the template to the system for execution.
`-1 <path>`	NULL	Sets the first element of the search path (after -b value).
`-2 path`	NULL	Sets the second search path element.
`-3 path`	NULL	Sets the third search path element.

Result codes

Return code	Description
0	Completed successfully
4	Completed with errors
20	Total system failure

TIP You can also use `-h headtest.template` to work with build templates. You can create `headtest.template` using the)VECTOR and)SET commands or create it in a template as part of a build:

- Using *bash* on UNIX or *cygwin* on Windows:

```
cat >headtest.template <<EOF
)DUMP )SET
EOF
```

- MVS:

```
//STEP100         EXEC PGM=IEBGENER
//SYSIN           DD    DUMMY
//SYSPRINT        DD   SYSOUT=*
//SYSUT2          DD   DISP=SHR,DSN=mylibrary(HEADTEST),
//SYSUT1          DD   *
)DUMP )SET 70
/*
```

MVS Template Testing Program

For the MVS version of the template testing program, see the following JCL example:

```
//USER1   JOB 'USER NAME',MSGCLASS=X
//*
//SETS SET B='//USER1.TEST'
//*
//TPL    EXEC PGM=MDHLTPLT,
//       PARM='-o DD:O -b . -t -w -1 "&B" TESTTPL'
//*
//STEPLIB  DD  DISP=SHR,DSN=MDH.V1010.MDHLLIB
//         DD  DISP=SHR,DSN=MDH.V1010.MDHLLPA
//SYSPRINT DD  SYSOUT=*
//CEEPRINT DD  SYSOUT=*
//*
//O        DD  SYSOUT=*
//*
```

Adding Template Variables to the Dimensions Configuration File

You can optionally add template variables to the Dimensions CM configuration file, `dm.cfg`, located in the following directories or data set.

- (Windows): `%DM_ROOT%`
- (UNIX): `$DM_ROOT`
- (MVS): `MDH.V1010.MDHPARM(MDHTDCFG)`

The table below lists the template related variables.

Variable	Description
DM_TEMPLATE_CATALOGn	Specifies a data set or directory search path where templates are located. You can specify an array of multiple search paths by adding multiple instances of DM_TEMPLATE_CATALOGn and incrementing the value of n, for example: ■ DM_TEMPLATE_CATALOG1 c:\prod\templates ■ DM_TEMPLATE_CATALOG2 d:\dist\templates Default locations: ■ Windows: %DM_ROOT%templates ■ UNIX and MVS: $DM_ROOT/templates
DM_TEMPLATE_CATALOGn (continued)	Notes: ■ On distributed platforms the search path specifies a single directory. ■ If you specify a file type, Dimensions CM looks in the search paths for matching files. On MVS, if you do not specify a file type, the default is TEMPLATE.
DM_TPLB_LOG	Switches on tracing of template processing. Specifies the name of the file that will contain detailed messages issued during template processing, and the substitutions performed. You can customize this string before using it as follows: ■ %d—inserts the date in the format *yyyymmdd.* ■ %t—inserts the time in the format *hhmmss.* ■ %p—inserts the *process id*. ■ %%—inserts one '%' character. Default: none Example: tmp/template_log_%d_%t.log
DM_TPLB_SPOOL	Specifies the name of the file or data set where the JCL will be written. The file is not passed for execution. This field may be useful when testing new templates. ■ Maximum field length: 100 bytes ■ Default: null ■ Example: tmp/spooled.jcl

(Sheet 1 of 2)

Variable	Description
DM_TPLB_TTLOG	Specifies that any trace sent to the file that you specify in DM_TPLB_LOG is also sent to the message routine (the trace comes back to the client to be displayed). ■ Default: null ■ Example: yes
DM_TPLB_SEQUENCE_PATH	Specifies the directory where the HFS file that maintains the build sequence number is stored. Default: the directory specified in the variable DM_TMP. If this variable is not set, the defaults are: ■ UNIX: /TMP ■ Windows: c:\temp

(Sheet 2 of 2)

Extending the Template Processor with User Written Functions

Template Processing Call-Out Functionality

Introduction

The functionality described here is a generalized call-out facility that allows user written code to run in-process as part of the template processing logic. User written code can live in user DLLs or on z/OS in an assembler module. The template processor can 'call out' to these routines, passing values in both directions. The user code is called with a control block (TEB) that holds the addresses of some callback routines and a context pointer, and a count and pointer array to the parameter list.

Callback Routine	Purpose
const char *TEB_ReadData(void *)	To get the next line of 'input'.
void TEB_WriteData(void *, const char *)	To write the output back to the template processor for inclusion into the input stream.
int TEB_Message(void *, const char *pszPrefix, const char *pszMessage)	To issue an error, warning, or informational message during processing. The last letter of the prefix drives the severity. 'E' causes the routine as a whole to fail.

Callback Routine	Purpose

`const char *TEB_GetString(`
`  void *,`
`  const char *pszName)`
	Looks up the corresponding symbol and retrieves a pointer to its value.

`int TEB_SetString(`
`  void *,`
`  const char *oszName,`
`  const char *pszValue)`
	Sets the value of the specified symbol to the value. Any previous data associated with the symbol is lost (the symbol is deleted first from the symbol table). FALSE is returned if there is an error.

Additional Template Directives

This call-out functionality is supported by the following additional template directives:

- `)CALL dllname.function ParametersList`
- `)SCRIPT dllname.function ParametersList`

  cards

  `)ENDSCRIPT`

These directives are implemented by the same code. The parameter `dllname` is optional:

- If you do not specify `dllname` some internal entrypoints are exposed.
- If you do specify `dllname` the DLL is loaded once and any entrypoints that are referenced in that DLL are resolved against the single handle.
- On z/OS, if you use the special Dllname Assembler, the entry point is fetched and called (no dll load is attempted).

The parameter `ParameterList` is optional and has similar syntax to that of a C program. For example, you can use pairs of double quotation marks (`" "`) to surround a single parameter with embedded spaces. The number of parameters, and the array of pointers to these parameters, is exactly the same as for a C program with `argc` and `**argv` held in the TEB block.

The 'cards' are records that are processed by the template processor (with replacement) that the function 'reads' using a specialized read processor. The function writes output back using a specialized 'Write' function that is passed through the template processor again, so any '`)`' directives in this stream are honoured.

Predefined Functions

- `durules`

  This function gets details from Dimensions CM useful for uploading that item into Dimensions CM. The syntax is as follows:

  ```
  durules <product_id> <project_id> <project root> <context_root>
      <file>
  ```

  where:

  `<product_id>` specifies the product into which the item will be placed.

  `<project_id>` specifies the project into which the item will be placed.

<project_root>	specifies the location of the root of the project within the project hierarchy (if using sub-projects, use "\" if not).
<context_root>	specifies the path that will be prepended to the file to get the full path.
<file>	specifies the file for which information is to be gathered.

Returned information.

This is a set of vectors, with one entry in each vector corresponding to each object; that is, all the [0] values pertain to the first object returned, all the [1] values to the second object returned, and so on.

DMDUCOMMENT	Comment for upload.
DMDUDESCRIPTION	Description of item.
DMDUDIRPATH	Directory path of item.
DMDUFILENAME	Filename of item.
DMDUFORMAT	Format of item (determined from upload rules).
DMDUITEM_ID	Suggested item-id for item.
DMDULIB_FILENAME	Suggested library filename for item.
DMDUOWNING_PART_ID	ID of part that is to own item.
DMDUOWNING_PART_PCS	PCS of part that is to own item.
DMDUOWNING_PART_VARIANT	Variant of part that is to own item.
DMDUPRODUCT_ID	ID of product that is to own item.
DMDUREVISION	Suggested revision for item.
DMDUSTATUS	Suggested status for item.
DMDUTYPE	Suggested type for item.
DMDUVARIANT	Suggested variant for item.
DMDUWSPATH	Suggested project path for item.

- pos

  The built-in function pos is called as follows:

  ```
  )CALL pos V3 VMAIN VSUB
  ```

  As the result of this directive:

  - The variable V3 is created, and its value is set to the result of the operation.
  - The variable VMAIN is searched for the substring VSUB. If there is no match, V3 is is set to the string -1; otherwise, it is set to the string representing the decimal number equal to the starting offset of the substring.

  For example:

  ```
  )SET VMAIN=This is mine now
  )SET VSUB=mine
  )CALL pos V1 VMAIN VSUB
  ```

  results in V1 containing the value 8;

  whereas

```
)SET VMAIN=This is mine now
)SET VSUB=this
)CALL pos V1 VMAIN VSUB
```

results in V1 containing the value 0.

- scanarea

  This function—which requires a connection to the Dimensions CM server—recursively scans the area specified and returns information on all files within the area. The syntax is as follows:

  ```
  scanarea <directory path> <area root> <filter>
  ```

  where:

<directory path>	specifies the location of the directory that will be the root of the search. It may be in NODE::DIR format.
<area root>	specifies the root of the area that contains the target directory. To scan the whole area <directory path> and <area root> will be the same.
<filter>	if specified, then only files matching this filter will be returned. An example filter is "*.txt".

Returned information.

This is a set of vectors, with one entry in each vector corresponding to each object; that is, all the [0] values pertain to the first object returned, all the [1] values to the second object returned, and so on.

DMCSFILENAME	The filename.
DMCSFILEPATH	The directory path containing the filename.
DMCSMODTIME	Last modified time.
DMCSSIZE	Byte size of file.
DMCSPERMISSIONS	Permissions of file (unix format)
DMCSISDIR	Y if this is a directory.
DMCS_LF_CHECKSUM	Checksum assuming line-endings are in LF (UNIX) format.
DMCS_CRLF_CHECKSUM	Checksum assuming line-endings are in CRLF (Windows) format.
DMCS_LF_SIZE	Size of file assuming line-endings are in LF (UNIX) format.
DMCS_CRLF_SIZE	Size of file assuming line-endings are in CRLF (Windows) format.
DMCS_FULLPATH	Fully specified path of file.
DMCSACCESSTIME	Time file was last accessed.
DMCSCREATTIME	Time file was created.
DMCSCAN_READ	Y if file can be read.
DMCSCAN_WRITE	Y if file can be written.
DMCSCAN_EXECUTE	Y if file can be executed.

DMCSDIR_HAS_SUBDIRS	Y if this is a directory and it has subdirectories.
DMCSDIR_HAS_FILES	Y if this is a directory and it has files in it.
DMCS_IS_DRIVE	Y if this is a drive.
DMCS_DRIVE_TYPE	If this is a drive, type of the drive.
DMCS_EXTENSION	The extension of the file, or empty if none.
DMCS_BASENAME	The basename of the file (part of name before extension).
DMCS_LEAFDIR	The lowest level enclosing directory, or empty if none.
DMCS_LEAFBASE	The directory path containing this file, minus the lowest level directory.

- `substring`

  There are some predefined functions that are implemented using this logic. The built-in `substring` entry point is provided, and is called as follows:

  ```
  )CALL substring V1 V2 spos len
  ```

  where:

spos	is optional and defaults to `0`. If `spos` has a negative value, then the start position is that many characters from the end.
len	is optional and defaults to `-1` (meaning the rest of the string).

  This directive reads nothing and writes back nothing, but the first variable `V1` is set to the substring of `V2` specified by the spos and len values.

  For example:

  ```
  )SET V2=The Quick Brown Fox
  )CALL substring V1 V2 0 2
  %V1
  ```

  puts 'Th' to output;

  whereas

  ```
  )SET V2=The Quick Brown Fox
  )CALL substring V1 V2 0 2
  %V1
  ```

  puts 'F' to the output.

Writing z/OS Code for this Interface

On z/OS, you could write such an exit in 'C' or possibly in assembler.

Extensibility

Depending on the context, additional pre-defined functions may be available.

Template Processing: Passing Data Between Steps

Overview

A template processing step can pass data to a later template processing step. To do this, five additional directives are provided:

```
)SAVE "groupname" variable/variablemask
)LOAD "groupname" [option]
)DELETE "groupname"
)IFGROUPDEF "groupname"
)IFGROUPNDEF "groupname"
```

where:

groupname	identifies a collection of data that can be manipulated by any template wishing to use the facility. The `groupname` can contain spaces, and can be any length. The quotation marks (`" "`) are optional, but if they are not specified then white space will delimit the group name.
variable	specifies both the name the variable will be saved as and, by looking that up, the value to save. For example: ■ If 'FROG' *does not* already exist as a symbol, '`)SAVE NewDomain FROG`' will fail. ■ If 'FROG' *does* already exist as a symbol, then: • If additionally the group '`NewDomain`' `does not` exist, it will be created; • otherwise, if '`NewDomain`' does exist, it will be updated by the addition of the symbol '`FROG`' set to its current value in the template processor symbol table.
variable mask	a variable mask is handled if the variable name contains either asterisk '*' or question-mark '?'. '*' matches any number of characters; whereas, '?' matches a single character at that point.
option	On a load, any variable in the group will be added to the current symbol table. By default, any variable of the same name in the current symbol table will be lost; this default behavior can be modified by specifying one of the following optional values: ■ `OVERWRITE` ■ `WARN` ■ `ERROR` ■ `DISCARDDUPLICATES`

Forcing Template Expansion in the pBem

You can force a template to be expanded by setting a symbol—which can be in the step or for the whole `pBem` job. This means that you can be sure that a specific template for a certain type of step will always be expanded even if optimization discards the step anyway. This processing is the default—the variable turns this off.

Semantics

)SAVE	saves the specified symbol or all matching symbols. If the group specified by group name does not exist, it will be created.
)LOAD	merges the specified group with the current symbol table. If no such group exists, an error will result. Remember, the incoming `)LOAD` symbol gets priority.
)DELETE	destroys the entire group, freeing the associated storage.
)IFGROUPDEF	behaves as `)IFDEF` and allows conditional logic based on the existence of the named group.
)IFGROUPNDEF	behave as `)IFNDEF` and allows conditional logic based on the existence of the named group.

Example Template

```
First line
)SET V2=This is a string of data
)CALL substring V1 V2 -6 8
'%V1.'
)CALL libsampleexit101uD.so.GetIdeas
)SET V4=My own parm 4
)SCRIPT libsampleexit101uD.so.DumpParameters P1 "prm 2" "prm 3"
    "%V4."
This is the first line of input to Dump - %V1.
And this the second
)ENDSCRIPT
***END***
```

The command:

```
template_test -b . -o spool.output foo2.template
```

will produce the following output:

```
First line
'f data'
Next parameter[0] = 'libsampleexit101uD.so.DumpParameters'
Next parameter[1] = 'P1'
Next parameter[2] = 'prm 2'
Next parameter[3] = 'prm 3'
Next parameter[4] = 'My own parm 4'
This is the first line of input to Dump - f data
And this the second
***END***
```

To create this output, the routine `libsampleexit101uD.so` has been created, and this looks like this:

```
/* Copyright Serena 2008
 * ---------------------
 * This is sample code and may be used as a guide for customers
 * wishing to create their own callouts.
 */
#include <stdio.h>

/* The following lines handle compilation on several platforms:
 * windows, unix and z/OS
 */

#ifdef _WIN32
#       define VBCALL __stdcall
#       define IMPFUN __declspec(dllimport)
#       define EXPFUN __declspec(dllexport)
#else
# ifdef PLATFORM_OS390UNIX
#   pragma linkage(GetIdeas, OS)
#   pragma linkage(DumpParameters, OS)
# endif
# define VBCALL
```

```
# define IMPFUN
# ifdef __GNUC__
# if __GNUC__ >= 4
#   define EXPFUN __attribute__ ((visibility("default")))
# else
#   define EXPFUN
# endif
#else
# define EXPFUN
# endif /* __GNUC__ */
#endif /* WIN32 */

/*
 * This is the common copybook for using this facility
 */

#include "template_callout_public.h"

/*
 * This first example just shows a call back to issue a message.
 * It does nothing else.
 * The last character of the prefix (JMH....W) is the severity of
    the message.
 * You should use 'E' for errors, 'W' for warnings, 'I' for
    information.
 */

EXPFUN void GetIdeas(pTeb pData)
{
    (pData->TEB_Message)
        (pData->TEB_pHandle,
         "JMHTPE1600001W",
         "Running the sample template exit");
}

/*
 * This function demonstrates a way of seeing what parameters are
 * passed to this function - both on the )SCRIPT line and in
 * the lines between the )SCRIPT and the )ENDSCRIPT.

 * So, it shows how to use TEB_PAramterCount and TEB_Parameters.

 * Then it shows how to call the TEB_WriteData routine to
 * send lines back to the templater for reprocessing.

 * The TEB_ReadData routine will return each line of input up to the
    )ENDSCRIPT
 * and these can be processed as you see fit.

 * All newlines have been stripped on input and should be left off
    on
 * output.
 */

EXPFUN void DumpParameters(pTeb pData)
{
```

```
    const char *pszInput = NULL;
    char szBuffer[10000];
    int i;

    for (i=0;i<pData->TEB_ParameterCount;i++)
    {
         sprintf(szBuffer,
             "Next parameter[%d] = '%s'",
             i,
             pData->TEB_Parameters[i]);
         (pData->TEB_WriteData) (pData->TEB_pHandle, szBuffer);
    }

    while (NULL != (pszInput = (pData->TEB_ReadData)
                                    (pData->TEB_pHandle)))
         (pData->TEB_WriteData) (pData->TEB_pHandle, pszInput);

}
```

You will need this copy book as well:

```
/*-----------------------------------------------------------------
 * Copyright (C) 2008, Serena Software Europe, Ltd. All rights
    reserved.
 *
 * No part of this software may be reproduced, stored, or
    transmitted, in any
 * form or by any means, without the prior permission in writing of
    Serena
 * Software Europe, Ltd and Serena Software, Inc.
 *-----------------------------------------------------------------
 * MODULE SPECIFICATION
 *  %PID%
 * Description:
 *  %PD%
 *%PCMS_HEADER_SUBSTITUTION_END%
 */

#ifndef TPCI_PUBLIC

#ifdef __cpluspls
 #ifdef PLATFORM_OS390UNIX
   extern "OS" {
   #pragma pack(1)
 #else
   extern "C" {
 #endif
#else
 #ifdef PLATFORM_OS390UNIX
```

```
   #pragma linkage(fReadData,OS)
   #pragma linkage(fWriteData, OS)
   #pragma linkage(fGetString, OS)
   #pragma linkage(fSetString, OS)
   #pragma linkage(fMessage, OS)
   #pragma pack(1)
 #endif
#endif

typedef int (fMessage)(void *pHandle, const char *pszPrefix, const
    char *pszMessage);
typedef int (fWriteData) (void *pHandle, const char *pszData);
typedef const char *(fReadData) (void *pHandle);
typedef const char *(fGetString) (void *pHandle, const char
    *pszSymbol);
typedef int (fSetString) (void *pHandle, const char *pszSymbol,
    const char *pszValue);

struct TplExtensionBlock {
   void         *TEB_pHandle;            /* handle to use with read/
    write/message */

   /* READ routine
    * - call this to get each line of the data up to the
    *   )ENDSCRIPT.
    * - NULL marks the end of the script.
    * - lines are terminated by a null with no newline.
    * Templater symbols will have been replaced in here.
    */
   fReadData *TEB_ReadData;

   /* WRITE routine
    * - call this to create new template lines for inclusion into
    the output stream.
    * - templater commands issued here will be honoured.
    * - The lines should terminate with a \0 character (and no \n).
    * - returns -1 on error.
    */
   fWriteData *TEB_WriteData;

   /* MESSAGE routine
    * - Call this interface to issue messages.
    * - The Message Prefix is a code which should end in a severity
    (E - error, W - warning, I - information)
    * - Processing will be deemed to have failed iff the callout has
    written an E message back.
    */
   fMessage *TEB_Message;

   /* The following routines provide a way of accessing the template
    processor symbol table.
    * The first routine will get a value from the symbol table.
    * NULL is returned if the variable doesn't exist or is vectored.
    */
   fGetString *TEB_GetString;
```

```
   /* This will put a symbol into the symbol table or if it is there
    * will alter it
    */
   fSetString *TEB_SetString;

   /* The following are the count of parameters on the line invoking
     the callback and
    * pointers to each in the traditional 'C' format.
    *
    * TEB_ParameterCount will never be less than 1.
    * *TEB_Parameters will point at the name of the called entry
    point.
    * Do not modify any value in these.
    */
int           TEB_ParameterCount;
char        **TEB_Parameters;
int   conId; /* connection id from a )LOGIN */

};

typedef struct TplExtensionBlock *pTeb;

#ifdef __cplusplus
 #ifdef PLATFORM_OS390UNIX
  #pragma pack(reset)
   }
 #else
  }
 #endif
#else
 #ifdef PLATFORM_OS390UNIX
  #pragma pack(reset)
 #endif
#endif

#define TPCI_PUBLIC
#endif
```

This sample exit has two routines implemented, and uses an include file `template_callout_public.h` which will be available to customers. The first routine just puts out a message. The second routine dumps its parameters (both the input line parameters and the script) back into the template processor.

Example Templates Utilizing The Rexx Interface

Rexx interface

Here is a quick summary of the REXX interface:

```
)SCRIPT MDHDREXX.REXX
..
..
..
```

```
)ENDSCRIPT
```

Rexx rules

1. Instead of "say", use "call mdhsay".
2. Instead of "exit", use "return".
3. Do not define any `rexx` signal handlers.
4. If you define a procedure, you need to add "expose (mdhexpose)" to the definition.
5. If you need to send a `MsgX` message, use "call mdherr 'MDHREX4700001E ...'".
6. You can specify one, two or three numbers after this space delimited. These are:
   - the size of the w/s the REXX interpreter needs to hold local data (in bytes)
   - the size of the area for storing `mdhsay` and `mdherr` messages (with some overheads for control) in bytes.
   - the number of lines the REXX occupies.

   These numbers default to 32000, 32000, and 2000. These values are the minimum; if you specify less it will not get used.

Rexx example 1 This example turns `http://www.something.com/remaining` into just "`something`"

```
)SET URL=http://www.serena.com/splat/bim/bam

)SCRIPT MDHDREXX.REXX
url="%URL."
parse var url "http://" . "." company "." .
call mdhsetvar "COMPANY",company
)ENDSCRIPT

COMPANY = %COMPANY.
```

Rexx example 2 This example reverses the slashes, "`/`" to "`\`" in an array

```
)VECTOR DMPATH(5)
)SET DMPATH(0)=/a/1/x/fo.c
)SET DMPATH(1)=/b/2/x/fo.c
)SET DMPATH(2)=/c/3/x/fo.c
)SET DMPATH(3)=/d/4/x/fo.c
)SET DMPATH(4)=/e/5/x/fo.c

)SCRIPT MDHDREXX.REXX
i=0
)REP DMPATH
tmp = "%DMPATH."
tmp=translate(tmp,"\","/")
call mdhszy ")SET DMPATH("i")="tmp
i=i+1
)ENDR
)ENDSCRIPT

)REP DMPATH
%DMPATH.
)ENDR
```

The Template Processor Interface

The callback facility has a new built-in 'catsearch' on MVS only. This invokes the C/C++ catalogue search interface, which allows flexible processing of dataset names and dataset member names within the template processor. The template processor creates a collection of parallel arrays for each matched object.

The template command for this has the form:

```
)SCRIPT catsearch dsnmask membermask opt1 opt2 …
)ENDSCRIPT
```

Possible options include:

- +/-mems
- +/-ds
- +/- aliases
- +/-mig
- pfx <prefix string>

For example

```
)SCRIPT catsearch MDHDEV.QUIXOTE.PRIMES.TXT.ASM * -ds +mems -mig
    -aliases pfx FF
)ENDSCRIPT
```

will create a series of variables ppxxxxx each of which is an array with one row describing each member of the specified dataset in turn. The pp denotes a user-settable prefix.

Once this command has been executed, additional variables are available in the template processor symbol table. These are prefixed DMCS by default, but there as an option to set a user specified value (pfx).

These variables can then be used within a)REP—)ENDR block to provide one-match-at-a-time logic.

The variables currently supported are:

Variable	Values	Description
ppALIAS	Y/N	Y => this is an alias.
ppBLKSIZE	Number	Blocksize of dataset.
ppCONTAINER	Y/N	Y => this describes a dataset that is capable of containing members
ppDSORG	String	MVS dataset organization—PO, PS, POU, PSU, DA.
ppENTRY	String	Name of data set.
ppFULLNAME	String	build/libfilesys/mdhhcats.h. Name of data set plus (member) in this row is talking about a member.
ppILQ	String	Last qualifier of dataset.

Variable	Values	Description
ppLLQ	String	Second to last qualifier of dataset.
ppLRECL	Number	Record length for FB files, largest legitimate record in Variable, `0` in RECFM U files.
ppMEMBER	String	NULL for a container or sequential dataset. Alias name for an alias (could be mixed case then). Member name for a full library member.
ppMIGRATED	Y/N	Y => this data set is migrated.
ppPDSE	Y/N	Y => this is an MVS LIBRARY.
ppPNAME	String	Primary name when the object is an alias.
ppPROGRAM	Y/N	Y => this is executable.
ppPROGRAMOBJECTTYPE	0	This is currently always `0`.
ppRECFM	String	Record format for this dataset. FB—fixed blocked, VB variable blocked, U undefined etc.
ppVOLSER	String	Volume serial of the first extent for this dataset.

Chapter 8

Structured Information Return

What is SIR?	246
Key features	246
SIR Commands	247

What is SIR?

Structured Information Return (SIR) enables you to perform scripting in any Dimensions CM command line interface. It is useful when you want to save, or inspect, additional or specific values by name. You can use SIR to return information in a format that you can:

- Insert into commands.
- Save to disk to be used in later sessions.

Key features

- SIR is an optional feature that you turn on using the SSPM command (see page 247). It remains on until the session completes or you turn it off.
- On the server end, the symbol table is added prior to the start of command processing and deleted after command processing is finished.
- On the client end, by default initially no symbol table is added until the SSPM command enables structured information return. When structured information return is turned on, there are always two symbol tables:
  - A persistent table that is explicitly updated as execution progresses.
  - A temporary symbol table resulting from the last remote command RPC that was issued.
- Use the SAVE command (see page 248) to copy values from the temporary symbol table to the persistent symbol table, usually with a different name.
- The structured object contains at least a return code that indicates the success or otherwise of the operation. 0 indicates success.
- A structured object has values added to it depending on the processing of a particular command, for example, the command used to create a change document.
- Symbolic replacement is performed on the remote command lines to be passed to the Dimensions CM server for execution.
- All Dimensions CM interfaces can use SIR, including:
  - The command console in the desktop client.
  - The command line interface (dmcli).
  - The z/OS batch command client.
- Client side processing is performed by the command line logic. Once SIR processing has been enabled, command lines are searched for `&var.` references and then resolved against the persistent and the temporary symbol tables.

SIR Commands

SSPM

The SSPM command controls SIR and has the following optional parameters:

```
ON [/USER_FILENAME="file"]
DBOTH
DPERSISTENT
DTEMP
NONE
OFF
LOAD
DUMP /USER_FILENAME="file"
DVAR variable
```

where:

- `ON [/USER_FILENAME="file"]`

  Turns SIR on. Use the /USER_FILENAME qualifier to specify where the session information will be saved.

- `DBOTH`

  Dumps the persistent and temporary symbol tables.

- `DPERSISTENT`

  Dumps the persistent symbol table.

- `DTEMP`

  Dumps the temporary symbol table.

- `NONE=OFF`

- `OFF`

  Turns SIR off.

- `DUMP /USER_FILENAME="file"`

  Restores the current persistent symbol table. Use the /USER_FILENAME qualifier to specify where the information will be restored from.

- `DVAR variablename`

  Displays a single variable’s value.

NOTE The parameters DBOTH, DPERSISTENT, and DTEMP display the values in each symbol table through the standard Dimensions CM messaging services. If SSPM processing is enabled in any command line, the special character '&' marks the start of a parameter name and '.' character marks the end. The text between the '&' and the '.' characters is used as a symbol for lookup in the temporary and persistent symbol tables and is replaced. The form '&&' is replaced by '&' and passed to the server.

SAVE

The SAVE command copies values from the temporary symbol table to the persistent symbol table. It has the following optional qualifier:

- `/LIT[ERAL]=value`

  Creates a symbol in the persistent table that has the specified value.

Examples

```
SAVE sym1 sym2
```

Causes the value of sym2 to be added to the persistent symbol table as sym1.

```
SAVE V1 /LIT="foobar"
```

Replaces `&V1.` with the value "foobar".

```
SSPM ON
   REXEC /NET=FOOBAR
         /TEMPLATE=MDHBARE1
         /PARAM=(DMHLQ=FOO, DMILQ=BAR, DMSAGE=WORK)
         /CAPTURE
         /BATCH
   SAVE JOB-NO DM_JOB_ID
```

Saves DM_JOB_ID as JOB-NO in the symbol table.

```
   RLIST &JOB-NO. /WAIT
```

Uses JOB-NO to issue a wait in the job started by the REXEC command above.

Chapter 9

Introduction to Build Templates

Understanding Build	250
What is a Build Template?	250
Build Template Types	254
Build Template Options	254
Expanding Wildcards	256
Using Build Configuration Inputs and Outputs in Templates	257
Handling Outputs and Error Logs	258
Generating a Bill of Materials	258
Search Paths	259
Initial Execution Path	259
Footprinting	259
Communicating between Templates	259
Callouts	260
Managing Build and Deployment Job Logs	260

Understanding Build

To understand the concepts discussed in this chapter you should be familiar with:

- Build configurations, which you define in Dimensions Build. For details see the *Dimensions CM Build Tools User's Guide.*
- Dimensions CM deployment including projects, streams, and areas. For details see the *Getting Started Guide*, the *User's Guide*, and the *Deployment Guide*.
- The template syntax used in build templates. For details see "The Templating Language and Processor" on page 185.
- The Primary Batch Execution Monitor (PBEM). The PBEM executes a build on the remote build machine and is invoked through its own PBEM template. Understanding the PBEM template enables you to follow what happens when a build is performed, and to run builds using the PBEM directly without using Dimensions CM. The PBEM template is called:
  - MVS: `TEMPLATE(MDHBPBM0)`
  - Windows: `pbem_start.cmd`
  - UNIX: `pbem_start.sh`

  For details about the `PBEM` architecture see the *Dimensions CM Build Tools User's Guide*.
- Secondary Batch Execution Monitor (SBEM): a standalone MVS only program, for details see page 279. For details about the `SBEM` architecture see the *Dimensions CM Build Tools User's Guide*.
- Bill of Materials (BOM): an XML report file detailing all the components included a build, for details see page 258.

What is a Build Template?

A build template is a customizable text file containing variables and control words that is processed and expanded when you run a build step. A build is a structured collection of build steps. You use a build template to control the execution of each build step. A build template has these main features:

- Contains code that defines how to build the target components of the system you are working with, typically a batch command file or shell script.
- Is written in a script language that is pre-processed by the template processor and creates an executable script, usually for the platform command processor.
- May produce a specification for a specific build tool for the platform the component is to be built on.
- Uses options from a build configuration to control the processing. Build configurations are defined in Dimensions Build, a build management, execution, and monitoring tool that is part of Dimensions CM.
- Implements build processing for a single build step. The definition of a build step is controlled by the build configuration parameters associated with the target and source combination.

- Utilizes symbols, provided in the symbol table used by the template processing step, to determine the names of the inputs and outputs of the step.
- Uses output symbols in the template processing language to return information to the build system about how the step is to be performed, and how the outputs of the step are to be processed. These choices include whether the script is to be run synchronously, how it is to be post-processed, what tools are to be invoked to control the execution.
- Uses a return code to encapsulate the success or failure of a build step.
- Is only executed when the build configuration that refers to it is built.

A build template may be in one of these formats:

- A script written directly in a build configuration.
- A script associated with a build configuration.
- A text file delivered as part of the controlled components of your system, which is located in a build area.
- A text file stored in the standard Dimensions CM location for templates.

After template processing has completed the resulting text file can be:

- A system command script:
  - Windows: a batch file
  - UNIX: a shell script
  - MVS: a JCL stream
- An input file intended for a specific program to process, for example, Make.
- Wrapped into another script or file to provide a standard environment for the step to run in.

Where are Build Templates Located?

Build templates can be located in:

- A build configuration in Dimensions Build.
- The main script associated with a build configuration.
- The build area as a checked-in and managed Dimensions CM object.
- The following default locations:
  - Windows and UNIX: the templates directory in the Dimensions Agent root area, typically: `%DM_ROOT%\templates`
  - MVS:
    - The instance definition as a library called `MDH.instance.TEMPLATE`

    or

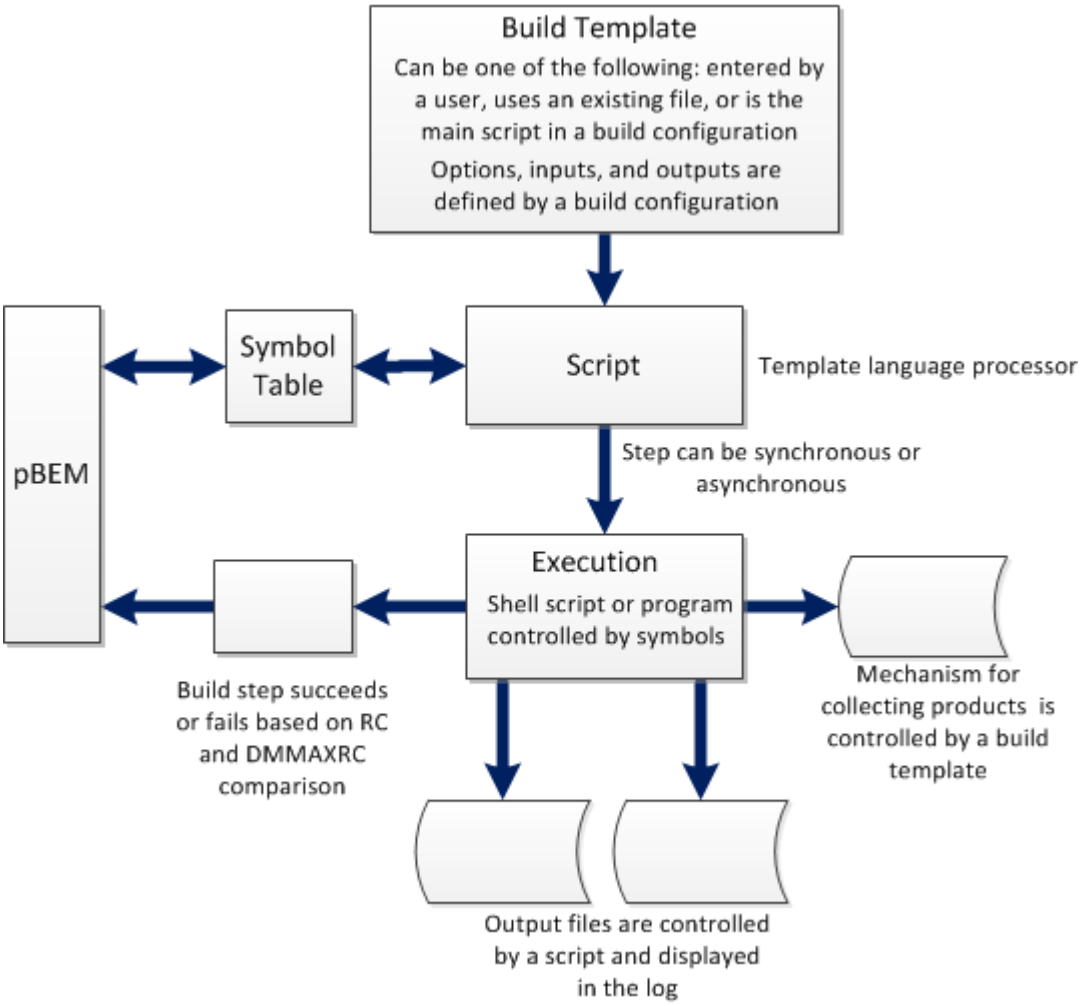
    - The templates library stored on USS in the $DM_ROOT area for the instance.

You can modify your Dimensions CM configuration file to point at additional locations.

Using Build Templates with Build Configurations

Build configurations define the inputs and outputs to each generalized build step, the options that are to be applied, and the order of the work to be performed. Each rule defined in a build configuration requires specifications of: inputs, outputs, build templates, options, the build order, and the expand wild card flags. These specifications describes how to carry out the detailed build processing. This text is defined using the build template and scripting language.

Executing a Single Build Step



Simple Build Template Example

UNIX	Windows
)SET DMSAVESTDOUT=YES)SET DMSAVESTDERR=YES echo Starting ANT build ant export rc=$? echo Ant completed rc = $rc exit $rc	)SET DMSAVESTDOUT=YES)SET DMSAVESTDERR=YES echo Starting ANT build call ant set rc=%%errorlevel%% echo Ant completed rc = %%rc%% exit %%rc%%

In the build template example above, assume that the source has been placed in a build area prior to the main build processing, either by a deployment or by populating a work area. The template has these main features:

- There is a single build step that builds everything.
- The environment for the build is minimal. It will fail if Ant or Java are not on the default search path on this system. The best practice is to construct the build environment using symbols defined in the build configuration.
- The build can construct many objects of different types in the single instance of the step. To make this work, the build configuration does not specify 'expand wild cards'.
- The step runs synchronously as this is the default on distributed platforms.
- By default the script is processed by the command shell for each platform. Communication of the log is controlled by the first two SET commands. These tell the build agent that there are two logs to pass back to the user: to the STDOUT and STDERR streams.
- Collection of outputs is performed by the server, which inspects the build area after the step has compiled, and calculates which items match the target specifications.
- The return of a result at the end is crucial; it defines to the build system whether the step has 'worked' or 'failed'.

NOTE The template processor processes all the text before it is written for a shell to execute. The default settings for the template processor means it interferes with the Windows syntax making the need to double the '%' symbol. The use of the template processor in this example is minimal.

Build Template Types

There are different types of build templates, each can be applied to distributed and MVS platforms.

High Level Templates

A high level template invokes a procedure or tool on a build machine such as a complete build of a sub-system. For example, it may invoke tools such as Make, Ant, or Openmake. You should use wild card patterns to set up build configurations to include all the source files needed for the whole build. For more details see page 263.

Compilation Templates

A compilation template configures a finely tuned compile operation including inputs, outputs, and options. Typical examples are compiling a single source file or pre-processing a single input. A compilation template can include multiple inputs and outputs but is not intended to iterate over multiple steps. For more details see page 263.

Collection Templates

Collection templates capture the common LINK paradigm where a set of objects of the same type are combined. This build configuration can contain multiple inputs and outputs, but the core of the operation is a loop, where each object in turn is combined to create the output. These operations are typical towards the end of a build where they collect together intermediate targets created earlier. For example, collecting a set of obj. files into an exe. file using a linker. For more details see page 264.

Coordination Templates

Coordination templates are used to sequence and link several different builds, including on different platforms. The build steps refer to invocations of Dimensions Build via the command line client.

Build Template Options

Build provides very flexible methods for you to control how your build steps are generalized. Your templates can be very specific or general. If they are specific, they are easy to code and get going, but this technique increases the amount of maintenance you need to perform. General templates are initially more complex to follow and require trial and error. Control of how your template works is driven by the symbols passed to the build step from the build system. Some symbols are always there; others you can define and add to meet specific needs.

The)DUMP command, part of the template language, shows you what symbols and their values are available for your use:

- UNIX: `)DUMP #`
- Windows: `)DUMP rem`

In the example above the entire symbol table will be put in the output script prefixed by the supplied comment operator.

Build options are named values that are replaced by the template processor if referenced in your build template. The name of the option is the name you give it when defining the option in either the build configuration, a)SET statement, or an option group. Its values are used to replace a reference to the symbol by the templater. For example, you might define an option called DEBUG with a value of Y or N. The template can interrogate this value and generate appropriate code to implement its meaning. Build options can be either a single value or an array of zero or more values.

Build options set template symbols. You can define them when you create a rule in a build configuration to build at one of these levels: target (transition), area, or global. If a definition for a symbol exists at the rule level it overrides the definition at the area level, which overrides a definition that is global. You can also code standard templates that you include in the expansion of your build template to provide basic defaults for system-wide or common values.

This example refines the example above so that the location of JAVA_HOME can be set for each area that you want to build in:

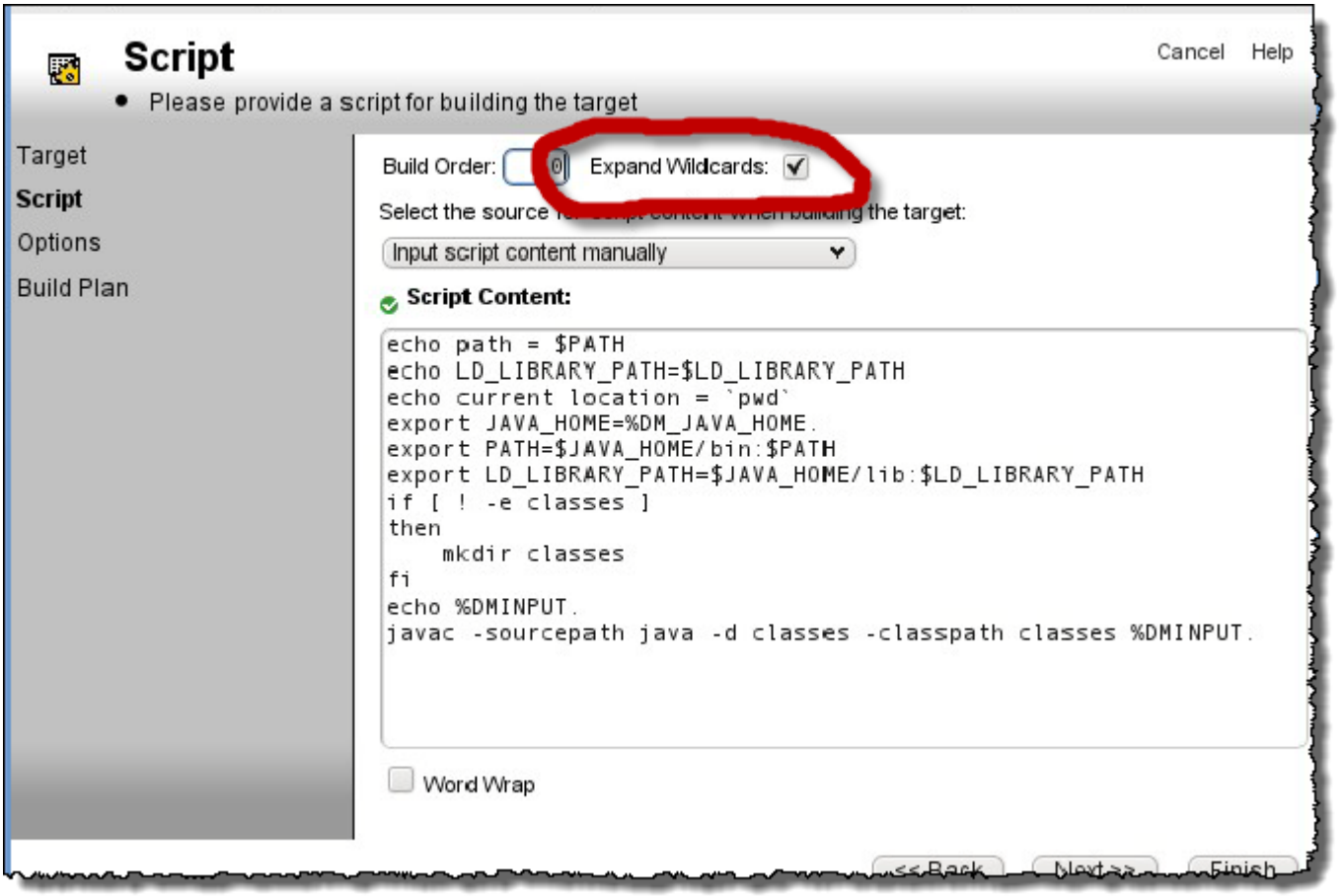
UNIX	**Windows**
)SET DMSAVESTDOUT=YES)SET DMSAVESTDERR=YES)SET DMEXECENV=CMD %%s)SET DMTASKNAME=ANT build)IFNDEF JAVA_HOME)SET JAVA_HOME=/opt/java16.sdk)ENDIF export PATH=%JAVA_HOME./bin:$PATH export JAVA_HOME=%JAVA_HOME. echo Starting ANT build ant export rc=$? echo Ant completed rc = $rc exit $rc	)SET DMSAVESTDOUT=YES)SET DMSAVESTDERR=YES)SET DMEXECENV=CMD %%s)SET DMTASKNAME=ANT build)IFNDEF JAVA_HOME)SET JAVA_HOME=c:\java16.sdk)ENDIF path %JAVA_HOME.\bin:%%PATH%% set JAVA_HOME=%JAVA_HOME. echo Starting ANT build ant set rc=%%errorlevel%% echo Ant completed rc = %%rc%% exit %e%rc%%

In this example, the template processor checks to see if there is any JAVA_HOME variable defined, and if not provides a default. The execution search path is set up to include the Java 'bin' directory, and the environment variable JAVA_HOME points to the location of your JDK. If you define JAVA_HOME on each area you intend to build in, the definition appropriate for that area is used. The variables DMTASKNAME and DMEXECENV are set to default values.

Expanding Wildcards

When you define a new target you implicitly set up a new rule. The rule can work on a collection of files or Build can generate a separate step for each matching source it finds in the input collection. You can specify the latter behavior by selecting the option Expand Wildcards. In this case, there is an implicit assumption that the targets will be named in a one to one relationship with the discovered sources.

This form of build template is particularly useful for mainframe (MVS) builds but you can use it on all platforms to process an entire class of sources to produce outputs with related names. For example, compiling *.lang programs to the *.obj format.



In the example above the option Expand Wildcards is selected and the template in the script is executed once for every Java program in the source project. DMINPUT expands to `java/<program>.java` because, in this example, all the Java programs being processed are held in that relative path in Dimensions CM.

Using a Build Type with Wildcards in Build Templates

Choosing sources with a wildcard may not be enough and you might need multiple file formats all stored as a single extension type. In Dimensions Build the Build Type is a synonym for the Dimensions concept of file format and you can use it as an additional filter to select sources. For example, in the source type .ASM you can support several sorts of assembler programs, some which require preprocessing and others that do not.

You can only use this feature with source files in conjunction with the Expand Wildcards option for a build rule.

To see the Dimensions CM format of an item, you can add it to the desktop client using customise views, or right click and choose Properties. To amend the item's format you can use the UIA command:

```
UIA "itemspec" /FORMAT=format
```

This command affects a single revision only.

NOTE Your installation may make it difficult to issue this command. When you work with build rules that use build types (Dimensions CM formats) you may need assistance from your Dimensions CM system administrator.

Using Build Configuration Inputs and Outputs in Templates

Build configurations allow you to name inputs and outputs from build steps. The following mechanism allows you to insert multiple input and output values into templates.

- DMPATH, DMPPATH

  Each is an array of area specifications for where the source is located. The first element is always the build area where the build is occurring. DMPPATH is similar to DMPATH except that the names have been modified to a form that is correct for the execution platform. For example, for Windows '/' becomes '\'.

- DMINPUT

  An array of values associated with the inputs to this build rule in the configuration. If Expand Wildcards is on this value replaces * with actual names from the project.

- DMTARGET

  An array of values associated with this build rule's target file names. If Expand Wildcards is on * is replaced by an actual value.

- _MDHDSN_

  A tool for searching an array of file names and retrieving specific values. It can also search for a file on disk and return its location.

- _MDHEXTRACT_

  A function for deconstructing and constructing path and file names for both distributed and MVS environments.

-)SET_PATH_NT var=value

  (Windows only) Converts '/' to '\'.

Handling Outputs and Error Logs

You can use the build configuration or the Dimensions CM configuration file to specify how outputs and errors from a running build step are treated. See the example on page 253 and descriptions for the following predefined template symbols:

- DMSAVESTDOUT on page 206.
- DMSAVESTDERR on page 206.
- DMCOMBINEOUTERR on page 207.

You can also use scripts to directly set the same variables:

```
)SET DMSAVESTDERR=YES
)SET DMSAVESTDOUT=YES
```

For UNIX and Windows templates these options refer to the *stdout* and *stderr* streams.

Generating a Bill of Materials

If a procedure that performs a build is capable of collecting the mapping between inputs and outputs, including secondary inputs such as header files, you can use this information to create a Bill Of Materials (BOM). A BOM is an XML file that describes what was built and what it was built from. You can generate BOMs with:

- The SBEM on MVS with Openmake on all platforms.
- dm_make on UNIX and Windows.

To generate a BOM you need to indicate the presence of the BOM to the PBEM by setting the variable DMOBOMRPT to the XML file name.

The supplied PBEM and SBEM templates on MVS create container libraries that store the BOMs so that they are unique across parallel steps and parallel builds. In distributed environments use the templating symbols DMUNIQUE and DMMICROSEC to do the same.

Using BOMs triggers a different mode of output collection where the process is driven by the contents of the BOM, and creates relationships between all the items listed in the BOM.

Without a BOM, only the items specifically mentioned in the build configuration are collected into Dimensions CM. For information about upload rules see the *Process Configuration Guide*. For information about virtual items see the *User's Guide*.

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, or the DM_SP_START_STAGE level.

Some environments use source elements from another area in the search path. Products that support this functionality include Openmake. It is also supported by the _MDHDSN_ function.

Initial Execution Path

The initial execution path is set to the area root in which the build is taking place and the build project offset if you specified one in the build configuration. You can check by including `cd` (Windows) or `pwd` (UNIX) as a command near the top of your template. On Windows, if your working directory is a UNC path, a temporary drive is mapped and unmapped to allow this to happen.

Footprinting

Footprinting occurs when you embed the name of a report produced at the end of the build into built objects. This process allows the recall of the precise make-up of a module from an executable. On MVS this is implemented in the MDHBLNKx template using a program that inserts an additional CSECT into your program containing the footprint information. It also contains the link deck used, which makes it possible to recover alternate entry points. The report is stored in Dimensions CM as a file, so that even if the Dimensions database is removed or decommissioned, the report is still available.

For more information, see the appendix *Creating and Embedding Build Footprinting* in the *Dimensions CM Build Tools User's Guide*.

Communicating between Templates

Different templates can communicate data between each other. This is useful for complex templates where you may need to send parameters from one step to another, later step. Use the following operators:)SAVE,)LOAD,)DELETE, IFGROUPDEF and)IFGROUPNDEF. For more details see page 234.

Callouts

The template processor is extensible. You can call REXX to process complex scripts and write callouts, in C or C++, which extend the templater function. This code can use parameters and read data from the template, process it as designed, and return results to the template for re-interpretation. It can also create variables in the main symbol table for use by the caller.

Callouts can use Dimensions APIs to access a Dimensions CM database via SQL or the other standard Dimensions APIs. A special directive, `)LOGON`, allows the template to log into a Dimensions database and use the connection parameters. For more information about this directive, see page 196.

NOTE You cannot use a one time certificate for this login.

Managing Build and Deployment Job Logs

Every build or deployment job creates an REXEC job that you can inspect from the command line or the administration console. You can delete jobs using the RDEL command or the administration console. When you delete a build job, all its related build logs are also deleted, which may clean up a lot of disc space.

IMPORTANT! If you do not clean up your jobs, over 50% of your database may be used by build logs.

Chapter 10

Build Templates for Distributed Platforms

Introduction	262
High Level Templates	263
Compilation Templates	263
Collection Templates	264
Using Search Paths	265
Asynchronous Templates	266
Configuring Messaging	266
Templating Techniques	267
Using Make and Creating a Bill Of Materials	271

Introduction

When you are constructing a build template a good place to start is an existing, working script. This script becomes the basis of the template. You add templating commands into the script to propagate the values that you need to perform the operation from the Dimensions build step symbol table into the scripting language context.

If you use the defaults on Windows there are conflicts between the Windows batch language syntax and the templater syntax. There are methods to mitigate this issue, for detail see page 267.

Scripting on Windows scripting is complex, however you may be able to avoid much of the complexity by careful use of the templater. Alternatively, use a different execution environment such as Perl, cygwin, or Powershell.

The Dimensions build system requires directions that are fed back from the script to influence the execution of the build steps. These directions are implemented by setting specific variables to appropriate values in the template processor symbol table. For details see page 249.

There are different approaches to using build, each with its own advantages and disadvantages. It is common to perform large step builds, using for example Make, Ant, or the Microsoft compiler in a batch mode. These are known as high level build templates, for details see page 263. You can also perform small step builds where the individual program compiles, and subsequent links are each a separate build step. These build types are discussed in Compilation templates (page 263) and Collection templates (page 264).

You can mix and match these approaches by using build ordering and expanding wildcards in build rules. For information on what the build template has available to it, and what information needs to be passed back, see page 249.

To track detailed made-of relationships you need to construct a Bill of Materials (BOM) document for each build step with one of these methods:

- Use the parameter`--xml-bom` with the `dm_make` command.
- Gather dependency information and using a tool to write the BOM.
- Write your own BOM, providing you adhere to the syntax for BOM documents.

If you do not use a BOM, the made-of relationships will be synthesized using all the identified inputs and outputs. Every input will be related to every output, which may result in very large quantities of data being generated.

High Level Templates

Typically you use high level templates for highly scripted managed builds such as Ant, dm_make, Make, or the Microsoft IDE running in batch mode.

You may need to pass the build tool switches to control its precise processing, for example: platform, language variants, and the features to be enabled. If you are building across platforms you may want to place the related outputs into directories named for the architecture. Therefore when the collection takes place, your artifacts are organised by, for example, platform, architecture, and release or test version.

It is convenient to attach variables related to the area, such as platform and architecture, to the area definition in the build configuration.

The template syntax includes pre-defined symbols, for details see "The Templating Language and Processor" on page 185.

Compilation Templates

The main part of a compilation template is choosing the correct filenames for the inputs and outputs according to the build configuration. There can be multiple inputs and outputs of different types.

When you create a build configuration in Dimensions Build you can specify any names for inputs and outputs. Templates that you write must honor those names. For each input of a particular type find the name specified in the build configuration for that object. Then search for that item in the search path, which is possibly a list of locations. This gives the actual on-disk name of the object that you want to use as your input file.

Do the following:

1 Locate the build area

Access the first element of `DMPATH`. Use the `+d` option of `MDHEXTRACT` for compatibility with the executing platform:

```
)SET CUR=_MDHEXTRACT_(+d,%(DMPATH(0)).)_
```

2 Locate the source code

There can be more than one type of input. For each input required you need to get the configuration name for that type. You may then need to search for the resulting name in the search path:

```
)SET SRC=_MDHDSN_(,DMINPUT,C,DMPATH)_
```

3 Get the target names for the outputs

There can be more than one output. For each output locate the configuration name for that type. There is no need to search in the search path as the output is always written to the build area:

```
)SET DST=_MDHDSN_(,DMTARGET,OBJ,)_
```

4 Pull out the primary name

When compiling a file you may want to create objects based on the filename. For example, if you are compiling `foo.c` you may want to create `foo.pdb`. You can extract the name component of the filename using the following method:

```
)SET PRM=_MDHEXTRACT_(+n,%XYZ.)_
```

5 Perform the build

In this example, XYZ is the result of earlier manipulations that contain, for example, `foo.c`. You now have all the variables required to do a build. The example below is a distributed template that does a simple copy. All aspects of the build configuration are honored. For example, it is not assumed that `foo.c` creates `foo.obj`. Both input and output are named according to the configuration:

```
copy /y "%SRC." "%CUR.\%DST."
```

This example is a complete compilation template:

UNIX	Windows
)SET CUR=_MDHEXTRACT_(+d,%(DMPATH(0)).)_)SET SRC=_MDHDSN_(,DMINPUT,c,DMPATH))SET DST=_MDHDSN_(,DMTARGET.o,)_)SET PRM=_MDHEXTRACT_(+n,%DST.)_ cp "%SRC.%" "%CUR.\DST." export rc=$?)SET DMEXECENV=%%s	)SET CUR=_MDHEXTRACT_(+d,%(DMPATH(0)).)_)SET SRC=_MDHDSN_(,DMINPUT,C,DMPATH)_)SET DST=_MDHDSN_(,DMTARGET,OBJ,)_)SET PRM=_MDHEXTRACT_(+n,%DST.)_ copy /y "%SRC." "%CUR.\%DST.")SET DMEXECENV=%%s

Collection Templates

A collection template is similar to a compilation template, the main difference is an iteration over all the inputs of the same type. For example, a normal `OBJ` link receives a variable sized array of input `OBJs` to be linked together. The template loops over this array and generates the correct link control cards. This is called a collection step.

The example below shows the key part of a Windows link template that constructs a list of inputs that can be given to a link or copy command. Note the use of the MDHDSN function to locate a given filename. This time only the first and fourth parameters of `MDHDSN` are set, which look for a given default name in a search path.

```
)SET LIST=
)REP DMINPUT
)SET SRC=_MDHDSN_(%DMINPUT.,,,DMPATH)_
)SET LIST=%LIST. "%SRC."
)ENDR
```

Using Mixed Inputs in Build Configurations

In the above example it is assumed that all the inputs in the `DMINPUT` array are the OBJs that you require. If there are other inputs in the configuration of a different type, such as a configuration named side-file, `DBRM` or similar, the loops around `DMINPUT` have to be enhanced to only select entries of the correct type. The example below shows you how to do this:

```
)SET LIST=
)REP DMINPUT
)SET TYPE=_MDHEXTRACT_(+t,%DMINPUT.)_
)IF %TYPE.=OBJ
)SET SRC=_MDHDSN_(%DMINPUT.,,,DMPATH)_
)SET LIST=%LIST. "%SRC."
)ENDIF
)ENDR
```

Using 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. For more details see page 259.

High level build templates are not intended for use with search paths. However, you can use search paths if they are supported by your build tool and the build tool can search for source code in the locations listed in the DMPATH array.

It you write a procedure to support search paths you can use the Dimensions CM deployment areas model to efficiently manage the hierarchy of build areas in a typical setup where most files are at the RELEASE level and a few active versions are at TEST.

Openmake supports search paths but in an indirect way. You can only define the named search path in Openmake itself, and only the name of this object is passed in the build configuration. The Openmake search path is not automatically created from the contents of the DMPATH array.

The example below shows how to use a *make* script template with a search path. The script itself is the makefile and DMEXECENV ensures that it is run through *make* and not the normal shell. The *make* vpath statement is generated from the DMPATH array that causes *make* to search for objects in the search path.

UNIX

```
)ATTR )~.():
%.tgt : %.src
        copy $^ $@
)SET search=
)REP DMPATH
)SET search=~search. "~DMPATH."
)ENDR
vpath %.src ~search.
all: a.tgt b.tgt c.tgt
)SET DMEXECENV=~DM_ROOT./dm_make.exe -f
    %s --no-cm
```

Windows

```
)ATTR )~.():
%.tgt : %.src
        copy /Y $^ $@
)SET search=
)REP DMPATH
)SET search=~search. "~DMPATH."
)ENDR
vpath %.src ~search.
all: a.tgt b.tgt c.tgt
)SET DMEXECENV=~DM_ROOT.\dm_make.exe -f
    %s --no-cm
```

Asynchronous Templates

UNIX and Windows build templates can run synchronously or asynchronously. A small piece of template syntax is required to allow the asynchronous task to communicate back to the controlling build monitor (PBEM) when it has completed.

The standalone utility `bldcomms`, which is supplied with Dimensions CM, can communicate with the `PBEM` using the parameters that you specify. The example below shows how to use `bldcomms` on distributed systems to communicate with the `PBEM`:

UNIX

```
…
…
EXPORT RC = 20
…
)IF %DMRUNMODE. = ASYNC
)CM
)CM the following is a single line
    command!
)CM
bldcomms monitormessage %DMPBEMNODE.
    %DMPBEMPORT. "R=%%RC%%. I=%DMSTEP.
    T=%DMYEAR.%DMMONTH.%DMDAY.%DMHOUR.%D
    MMINUTE.%DMSECOND.%DMMICROSEC."
)ELSE
exit %RC.
)END
```

Windows

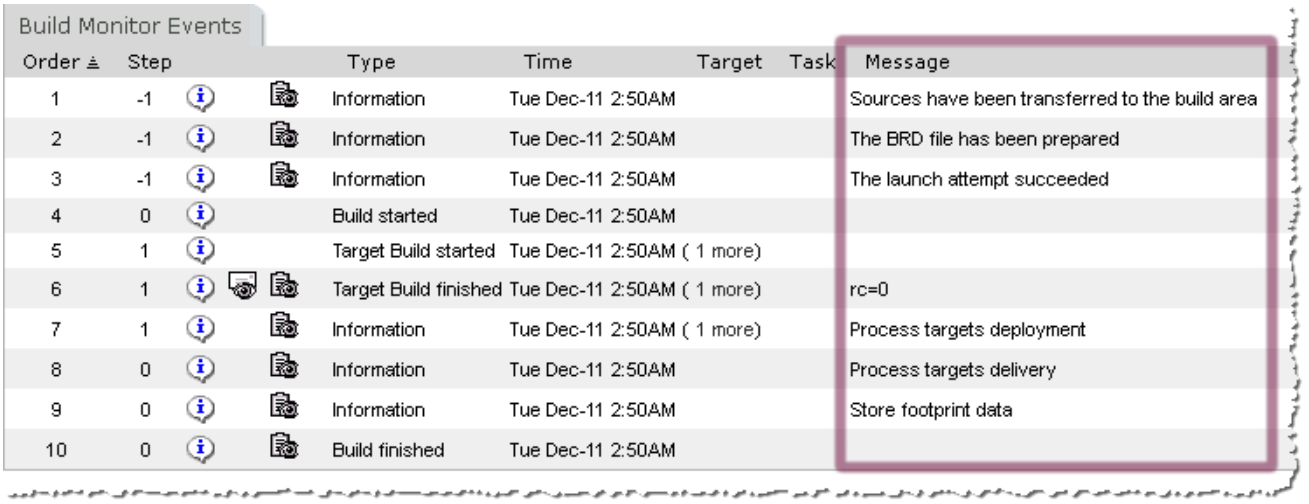
```
…
…
SET RC = 20
…
)IF %DMRUNMODE. = ASYNC
)CM
)CM the following is a single line
    command!
)CM
bldcomms monitormessage %DMPBEMNODE.
    %DMPBEMPORT. "R=%%RC%%. I=%DMSTEP.
    T=%DMYEAR.%DMMONTH.%DMDAY.%DMHOUR.%D
    MMINUTE.%DMSECOND.%DMMICROSEC."
)ELSE
exit %RC.
)END
```

In this example the result is `RC`. When run synchronously this is the return from the template. When run asynchronously this return is part of a message passed back to the `PBEM`. The other variables used in this example are pre-set by Dimensions CM and described in "The Templating Language and Processor" on page 185.

For additional details about `bldcomms` see the *Dimensions CM Build Tools User's Guide*.

Configuring Messaging

When you run in asynchronous mode the steps communicate back to the `PBEM` using the `bldcomms` utility. In both asynchronous and synchronous modes the `PBEM` then communicates with Dimensions Build server to update it about the progress of the build. Dimensions Build displays build messages in the Message column of the Build Monitor Events section of the Build Job Details dialog box (which appears after you start a build).



Build Monitor Events

Order	Step			Type	Time	Target	Task	Message
1	-1			Information	Tue Dec-11 2:50AM			Sources have been transferred to the build area
2	-1			Information	Tue Dec-11 2:50AM			The BRD file has been prepared
3	-1			Information	Tue Dec-11 2:50AM			The launch attempt succeeded
4	0			Build started	Tue Dec-11 2:50AM			
5	1			Target Build started	Tue Dec-11 2:50AM (1 more)			
6	1			Target Build finished	Tue Dec-11 2:50AM (1 more)			rc=0
7	1			Information	Tue Dec-11 2:50AM (1 more)			Process targets deployment
8	0			Information	Tue Dec-11 2:50AM			Process targets delivery
9	0			Information	Tue Dec-11 2:50AM			Store footprint data
10	0			Build finished	Tue Dec-11 2:50AM			

You can specify the task name for a particular step by providing the PBEM with a meaningful description string that is used instead of the generic message. To supply a task name, set the variable DMTASKNAME to the required string and optionally add other variables:

```
)SET DMTASKNAME = Step %DMSTEP. - Compiling %SOURCE.
```

You can also send messages directly from the template that is executing to Dimensions Build. The purpose of these messages is to report on the progress of long steps. For example, a large *make* script could send a message for every sub-directory that it starts to build or for every target it creates. To send these types of messages use the bldcomms utility but with different parameters:

```
bldcomms buildmessage %DMSERVER.  0 %DMTOKEN.  3  0  0 "END"
    %DMSTEP. "Phase2" "Target1" "just built foo.exe"  0 "" "" "" ""
```

There are many parameters for bldcomms, however the key ones are the last three string fields that you can use for any message. The messages also appear in the Build Monitor Events section of the Build Job Details dialog box. For more details about bldcomms see the *Dimensions CM Build Tools User's Guide*.

Templating Techniques

This section describes techniques for solving specific templating issues.

Using Special Characters in Windows

When you use templates there can be problems with special characters when there is a conflict between the script language and the templater. For example, by default template variables are introduced with %, which is also used by DOS.

Using the)ATTR command

To mix template variables with other syntax you can use the)ATTR command to change the special characters that the template uses. In the example below, PATH is a DOS variable and DMPATH is a template symbol:

```
)ATTR )~.():
ECHO PATH is %PATH%
ECHO DMPATH is ~DMPATH.
```

In this example, the % character is no longer being used to signal to the template processor the start of a variable. The ~ character is used instead. For full details of the)ATTR command see page 188.

TIP If you use this technique choose a conventional set of alternate characters and use it for all your templates.

Escaping Characters that Cause Problems

You can use special characters twice though this produces results that are more difficult to read. In the example below the first line is the version in the template, and the second is the version in the script after templating:

```
ECHO PATH is %%PATH%%
ECHO DMPATH is %DMPATH.
```

Another class of problem arises if the character data you have in the variables you receive from Dimensions CM contains special characters that have meaning to the shell. In general, you need to avoid this situation as much as possible. Placing strings in double quotes may help. However putting quotes into Dimensions metadata is not advised.

)EXPAND and)ENDEXPAND

One technique is to set all the variables you will require for the rest of the script, and then turn off template expansion as follows:

```
set MYPATH = %FOOBAR.
…
)ENDEXPAND
ECHO Path is %PATH%, my path is %MYPATH%
```

It is also possible, using)EXPAND, to turn expansion back on. For more details see page 193.

Handling Array Variables

Variables can be arrays. Some pre-defined symbols, and those created manually with the)VECTOR command, require special handling. If you want to process several array variables at once, each having the same number of elements, then list the variables on a

)REP. This passes all the arrays in parallel. Normally some kind of loop around each element of the array is needed as shown in the following example:

UNIX	Windows
)SET COUNT=0)REP DMINPUT)ADJUST COUNT 1 echo In %COUNT. = file %DMINPUT.)ENDR	)SET COUNT=0)REP DMINPUT)ADJUST COUNT 1 ECHO Input %COUNT. = file %DMINPUT.)ENDR

Occasionally you might want to test the filename inside a loop to find items that match a criteria. A common criterion is to loop around all inputs and find all those that match a particular type. This is the purpose of the MDHEXTRACT function, which can break a filename into pieces so that they can be compared:

UNIX	Windows
)REP DMINPUT)SET TYPE=_MDHEXTRACT_(+t,%DMINPUT.)_)IF %TYPE.=C Echo %DMINPUT. Is a C file)ENDIF)ENDR	)REP DMINPUT)SET TYPE=_MDHEXTRACT_(+t,%DMINPUT.)_)IF %TYPE.=C Echo %DMINPUT. Is a C file)ENDIF)ENDR

For details about MDHEXTRACT see page 223

A common technique is to build up a larger structure from a list of files or directories. The example below makes a single list from all the elements of DMINPUT.

```
)SET LIST=
)REP DMINPUT
)SET LIST=%LIST. %DMINPUT.
)ENDR
```

Picking a Specific Array Value

The symbols DMPATH and DMPPATH return arrays of specifications for the various areas of a search path in the order they should be searched. DMPATH returns this value in unidir format, and DMPPATH returns it in a format correct for your platform. The first element of this area is always the current build area. To access this directly use:

```
%(DMPATH(0)).
```

or

```
%(DMPPATH(0)).
```

Decomposing a Path or File Specification

You can use the function MDHEXTRACT to pull part file specifications. The function operates on a file path and name, or a file name, and extracts portions to be reassembled into a form that you request. The following example works for both Windows and UNIX.

```
)REP DMINPUT
)SET TYPE=_MDHEXTRACT_(+t,%DMINPUT.)_
)IF %TYPE. = C
echo %DMINPUT. Is a C file
)ENDIF
)ENDR
```

where:

- +t indicates the string following the comma (a symbol which is expanded prior to this operation running) is to be analyzed as a file specification, and the extension of the file returned as a value for the variable TYPE. You can pull out other information with this construct.
- +n returns the unadorned file name. If you add a third parameter (the root) the supplied name can be split into root and subdirectory.

For more details about MDHEXTRACT see page 223.

NOTE To find a given unique file in a list and search for it in the search path, use the `MDHDSN` function. For details see page 220.

Combining STDERR and STDOUT Streams

To combine your stderr and stdout streams use the following template syntax:

```
)SET DMCOMBINEOUTERR=YES
```

The result is a single stream returned as a log to the build server.

Submitting Formatted Text after Preprocessing

You can tell the template processor how to submit the formatted text after preprocessing using the variable DMEXECENV. By default, the text is passed synchronously to a command processor for the platform. There are however other options. For example, you might want to place the generated script into a wrapper script (also preprocessed, but after the first script has been handled):

```
)SET DMEXECENV=TEMPLATE perl.template
```

The example above looks for a template called perl and processes it in turn. The writer of this template can choose where the text from the main build template is to be placed, and establish an execution environment suitable for executing this text.

The perl template writer sets DMEXECENV to a new value so that this process does not continue recursively

UNIX	Windows

)SET DMEXECENV=TEMPLATE perl.template
my $a=10;
my $b=20;
my $c = $a + $b;
print "C = ", $c, "\n";
)SET DMEXECENV=BASH %%s
perl <<EOF
)CALLBACK NT
EOF
export rc=$?
Echo Final return code is $rc
exit $rc
	This facility is not available on Windows.

For more details about DMEXECENV see page 212.

Using Make and Creating a Bill Of Materials

The dm_make utility (supplied separately from Dimensions CM) works in conjunction with a program called mcxslave to provide compilations of complex systems and create BOMs. This facility works on all distributed platforms, including USS for z/OS.

The following simple example for UNIX and Windows illustrates the use of Make.

```
ifeq ($(PLATFORM),linux)
   EXE := x
   CCOMP := gcc
endif
ifeq ($(PLATFORM), windows)
   EXE := exe
   CCOMP := gcc-4
endif
all : alpha.$(EXE) beta.$(EXE)
alpha.$(EXE) : alpha.c
   $(CCOMP) -o alpha.$(EXE) -g alpha.c

beta.$(EXE) : beta.c gamma.c
   $(CCOMP) -o beta.$(EXE) beta.c gamma.c
```

The example below assumes alpha.c, beta.c and gamma.c are to be combined to make alpha.exe and beta.exe. For Windows, it also assumes that cygwin has been installed. It also takes no account of header files, although if you use adg style processing these will also figure in the BOM.

UNIX

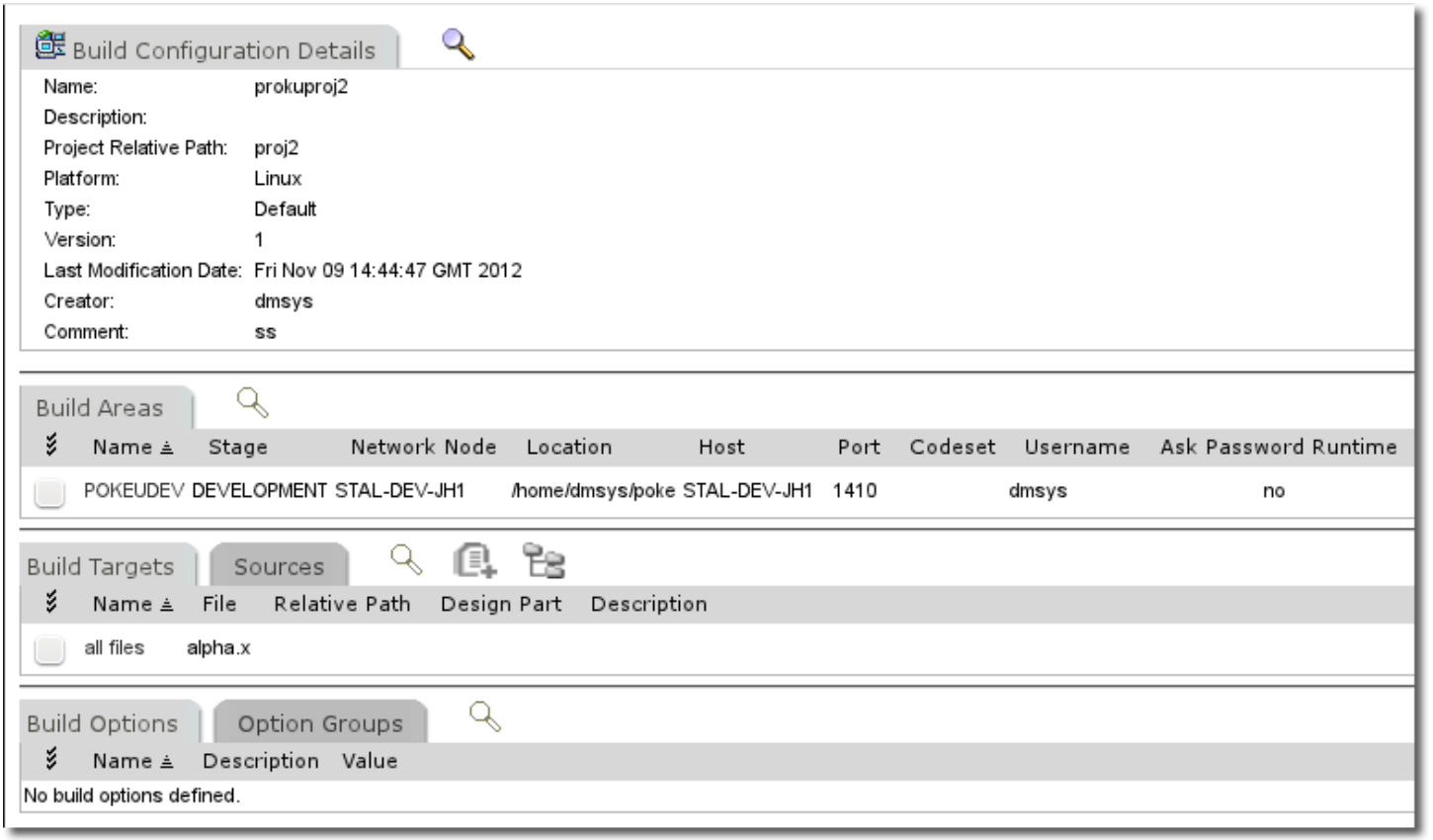
```
)CM
)CM we want intermingled stderr and
stdout
)CM
)SET DMCOMBINEOUTERR=YES
)CM
)CM we want a BOM with a non-reusable
name
)CM
)SET DMOBOMRPT=./
£DMUNIQUE..£DMMICROSEC..£DMSTEP..bom.
xml
)CM
)CM Dump symbol table if requested
)CM
)IFDEF DM_TPLB_TTLOG
)CM symbol has to be defined..
)IF £DM_TPLB_TTLOG. = YES:
)CM and has to be YES
)DUMP # 72
)ENDIF
)ENDIF
#
# set up dimensions environment for
some
# release of dimensions
#
. /opt/serena/dimensions 12.2.2/
dmprofile
#
# document where this is running
#
pwd
#
# document environment
#
echo $PATH
#
# delete all execs first
#
rm *.x
#
# use make to build the product
#
dm_make -f proj2.mk \
   --xml-bom=%DMOBOMRPT.\
   PLATFORM=linux
export rc=$?
#
# document the make return code
#
echo make returned error code = $rc
#
# document the bom we are using
#
cat %DMOBOMRPT.
#
# tell the pBem how it all went
#
exit $rc
```

Windows

```
)CM
)CM make this friendlier for windows
)CM
)ATTR )£.()
)CM
)CM we want intermingled stderr and stdout
)CM
)SET DMCOMBINEOUTERR=YES
)CM
)CM we want a BOM with a non-reusable name
)CM
)SET
DMOBOMRPT=.\£DMUNIQUE..£DMMICROSEC..£DMSTEP
..bom.xml
)CM
)CM this would default
)CM
)SET DMEXECENV = cmd /c %s
)CM
)CM Dump symbol table if requested
)CM
)IFDEF DM_TPLB_TTLOG
)CM symbol has to be defined..
)IF £DM_TPLB_TTLOG. = YES:
)CM and has to be YES
)DUMP rem 72
)ENDIF
)ENDIF
rem
rem document where this is running
rem
cd
rem
rem document environment
rem
echo %PATH%
rem
rem make cygwin compilers available
rem
path c:\cygwin\bin;%PATH%
rem
rem delete all exes first
rem
rm *.exe
rem
rem use make to build the product
rem
dm_make -f proj2.mk ^
   --xml-bom=£DMOBOMRPT. ^
   PLATFORM=windows
set rc=%ERRORLEVEL%
rem
rem document the make return code
rem
echo make returned error code = %rc%
rem
rem document the bom we are using
rem
type £DMOBOMRPT.
rem
rem tell the pBem how it all went
rem
exit %rc%
```

The examples above contain code that you may wish to drop in your production template. All the targets are deleted first, which you do not have to do. You can use a *clean* script, or use dm_make and specify a *clean* target. '`*.x`' that matches targets on UNIX for this build.

This is the UNIX version of the build configuration in Dimensions Build:



Chapter 11

Build Templates for MVS Platforms

Introduction	276
Writing Plain JCL Templates	279
Wrapping Generated JCL into the SBEM	280
Using Outputs in a Build Template	281
Build Step Warning Messages	282
Computing the Final Return Code	282
Creating a Bill of Materials from the SBEM	283
Source Types	284
Mapping between MVS and Dimensions CM	284
Using REXX in a Build Template	286
Controlling Simultaneous Conflicting ENQs in Data Sets	290

Introduction

The processing for MVS build templates is similar to that for distributed platforms (see Chapter 10, "Build Templates for Distributed Platforms" on page 261) however there are differences, particularly in the mandatory asynchronous execution of build steps, and in the mechanisms used to capture relationships between the various built items.

MVS build templates are typically larger than distributed ones, and handle smaller units of compilation and link. Templates are supplied with Dimensions CM, which you can use unaltered out of the box or modify to suite your environment. The templates are located in MDH.instance.TEMPLATE and are a copy of the code base versions. You can compare your instance versions with the code base version. If you check your templates into Dimensions CM you can use version control to compare revisions.

Secondary Build Execution Monitor

MVS builds normally use the Secondary Build Execution Monitor (SBEM), a specialised JCL execution engine that is closely integrated with the rest of build. The SBEM occupies a similar place to the command shell in distributed platforms. It executes build steps in an environment configured so that relationships between source, object, and executable modules can be observed and recorded. It handles the asynchronous communication required to report on progress, and constructs a Bill of Materials (BOM) for each build step, which defines the various objects created and their relationships.

The SBEM executes a subset of JCL that allows all of the processing you need to perform MVS builds. Some standard JCL syntax is missing and there are ways of working around these shortcomings.

The SBEM has the following limitations:

- Does not implement PROC-PEND or JOB cards.
- Does not provide symbolic replacement, although the template processor processes the templates first, and this can provide a lot of equivalent functionality.
- By default, does not wait for a data set resource if none is available (you will get an error from dynamic allocation). See page 290.

There are some situations where you need to communicate with the SBEM to force particular actions. You can do this using specialized comments in the input stream to the SBEM. These are called SBEM directives and all start with `//*SBEM`.

For more information on the SBEM see the appendix *Dimensions Build Utility Programs* in the *Build Tool User's Guide*. For details about using the SBEM in build templates, page 280.

IMPORTANT! The word STEP is ambiguous in the context of MVS builds. A build step is a single task for the PBEM to schedule an SBEM to execute. That task may in turn require multiple JCL steps to be executed.

MVS Templates

The MVS templates supplied with Dimensions CM produce both object format files .OBJ (LRECL 80, RECFM FB) and linked NCAL modules .LNKLIB (LRECL0, RECFM U). You can choose either format to standardize on. The better format is .LNKLIB as it is faster to link with.

The LINK template (MDHBLNK0) expects as input a SYSLIN deck to define the rules for constructing the output module. This is the input to the linkage editor setting the parameters for the link edit, and specifies at a minimum a starting module to be loaded (INCLUDE).

Member name	Main inputs	Expected main outputs	Purpose
MDHBASC0	*.ASM	*.OBJ *.LNKLIB	Compile a single Assembler program
MDHBCBC0	*.COBOL	*.OBJ *.LNKLIB	Compile a single COBOL program
MDHBCCB0	*.CCOBOL	*.OBJ *.LNKLIB	Compile a CICS COBOL program
MDHBMAP0	*.MAPSET	*.LOAD *.H *.PLI *.COPY *.ASM	Compile a CICS BMS mapset.
MDHBCCC0	*.C	*.OBJ *.LNKLIB	Compile a single C program
MDHBPLC0	*.PLI	*.OBJ *.LNKLIB	Compile a single PL/1 program
MDHBLNK0	*.SYSLIN	*.LOAD	Link edit a collection of compiled programs into a single load module or DLL
MDHBDUMY			Dummy template

There is currently no support for DB2 but it is not difficult to setup.

Basic MVS Example Template

Assume that you have a system consisting of many Assembler and COBOL programs that are combined into a small collection of programs. In this environment there are many copy books, both for the Assembler components and for the COBOL programs. In this example there is a very deep nested set of called modules, where the name of the module matches the name of the entry point.

Using the administration console the following rules are defined:

Rule	Output	Input	Expand Wildcard	Build Order	Template	Description
1	*.LNKLIB	*.ASM	Y	1	MDHBASC0	Compiles all the Assembler programs. Macros and copy books are used in the assemblies and dependencies are tracked.
2	*.LNKLIB	*.COBOL	Y	1	MDHBCBC0	Compiles all COBOL programs. The copy statements are all resolved and tracked.
3	*.LOAD	*.SYSLIN	Y	2	MDHBLNK0	Links all mainlines. For every mainline there is a single SYSLIN link deck that defines the addressing mode, residency mode, and main program for each mainline. AUTOCALL can be used to resolve most dependencies. All dependencies are tracked automatically.

This set of rules is a good first start for an ordinary batch system. Particular rules for specific source modules can use either Dimensions attributes or *PROCESS cards to manage variations in the compiles between modules.

The wildcards means that single steps build single modules and the unit of dependency is the smallest possible. Changing a source module will result in rebuilding fewer components.

The build order means that all LNKLIB modules will exist as soon as linking starts. When a final module is being linked, all the modules that might be required will have been built.

The SBEM provides I/O monitoring both for compile and link processing. You can track the dependencies between inputs and outputs (targets), and apply this information when you need to rebuild part of the system.

These templates support search paths and the construction of the load modules includes footprinting.

TIP The templates MDHBLNK0, MDHBASC0, and MDHBCBC0 contain descriptions of the syntax.

Terminating a Build Step

MVS templates must run asynchronously as they become batch jobs using JCL. A small piece of template syntax is required to allow the asynchronous task to communicate back to the controlling build monitor (PBEM) when it has completed. The required processing is different when you use an SBEM step to control template execution and when you write plain JCL.

Using the SBEM

The SBEM handles message passing to the PBEM. The `//*SBEM END` statement is a directive to the SBEM to send the step completion message. The message has to contain specific information to identify the type of message being sent, such as the step number, and a date and time stamp which acts as a check that the PBEM performs to make sure this message is from the expected client. For example:

```
)SET TIMESTAMP=%DMYEAR.%DMMONTH.%DMDAY.%DMHOUR.%DMMINUTE.
)SET TIMESTAMP=%TIMESTAMP.%DMSECOND.%DMMICROSEC.
//*SBEM END I=%DMSTEP. T=%TIMESTAMP.
)SET DMEXECENV=TEMPLATE TEMPLATE(MDHBSBM0)
```

where:

- TIMESTAMP is built by the PBEM at template processing time.
- DMSTEP is a number that defines which step this is and allows the return code (inserted by the SBEM) to be associated with the correct sub task.

Writing Plain JCL Templates

You can write plain JCL templates to run programs that fail in the SBEM context, or because you do not want to run the SBEM authorised but do want to run an IKJEFT01 (batch TSO) step. However, automatic monitoring cannot be performed.

The standalone utility MDHLBLCM, which is supplied with Dimensions CM, can communicate with the PBEM to relay build step return codes.

To run from plain JCL without the `SBEM` do the following:

```
)SET TIMESTAMP=%DMYEAR.%DMMONTH.%DMDAY.%DMHOUR.%DMMINUTE.
)SET TIMESTAMP=%TIMESTAMP.%DMSECOND.%DMMICROSEC.
//*
//STEP200 EXEC PGM=MDHLBLCM,
)CM
)CM next line should end col 73.
)CM
//          PARM='POSIX(ON),ENVAR("_CEE_ENVFILE=DD:ENV")/monitormessage
)CM
)CM next line should start col 17. (we need the space in col 16)
)CM
//        %DMPBEMNODE. %DMPBEMPORT. "R=0 I=%DMSTEP. T=%TIMESTAMP."'
//*
//STEPLIB  DD  DISP=SHR,DSN=%DMCDHLQ..%DMCDILQ..MDHLLIB
)IF %DMCDLPA. = NO
//         DD  DISP=SHR,DSN=%DMCDHLQ..%DMCDILQ..MDHLLPA
//ENDIF
//ENV      DD  DISP=SHR,DSN=%DMCDHLQ..%DMINST..PARM(MDHTDCFG)
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
```

To create an MVS high level style build step, add the above code to an existing JCL procedure. The JCL runs normally and tells the PBEM when it finishes that it has succeeded. To send back a failure, more complex JCL including a fragment of code with `R=n` is required, where `n` must exceed DMMAXRC for this step.

For more details about `MDHLBLCM` see the appendix *Dimensions Build Utility Programs* in the *Dimensions CM Build Tools User's Guide*.

Wrapping Generated JCL into the SBEM

Invoking the SBEM is complex as the JCL must be correct to invoke it. Your build template expands to inline JCL, which the SBEM reads as data. So that the build template can control which SBEM template is used for execution, and to provide parameters to that template, a facility is provided in the template processor for post-processing a second template as part of the submission process. The first set of output records is then 'wrapped' into the second template, prior to submission to the Job Execution System (JES). You have one place where the SBEM invocation is defined and all your build templates can share the single job card and SBEM logic.

This example is used in all the supplied templates:

```
)SET DMEXECENV=TEMPLATE TEMPLATE(MDHBSBM0)
```

In the general form this is:

```
)SET DMEXECENV=TEMPLATE templatename
```

NOTE The second template must redefine `DMEXECENV` otherwise you will get a loop. In the template MDHBSBM0 there is the following line:

```
)SET DMEXECENV=JCL
```

The location where the original template output needs to be placed is indicated by the following special template processor directive:

```
)CALLBACK NT
```

where NT means 'no tailoring' and the whole line is replaced by all the lines of the processed template. For more details see page 189.

NOTE A build template can control aspects of the SBEM JCL via variables.

Using Outputs in a Build Template

Using Outputs from Build Steps

You can use a build configuration or a Dimensions build template to send the text that you want to be visible from the build log to the build server. Create some JCL that writes small text files and point the DMOSTDOUT and DMOSTDERR variables at the files, for example:

```
)SET DMSAVESTDOUT=YES
)SET DMSAVESTDERR=NO
)SET DMOSTDOUT=//%CURDIR.ASMTERM(%SRCMOD.)
```

The compiler(s) have options that can create suitable mini-summaries (SYSTERM, TERM).

IMPORTANT! This data is stored on the database as a CLOB or CLOBs. You should keep the STDOUT and STDERR files small. Do not store whole listings.

Using Listings and Reports

Listing files are important but are typically large and are handled by Dimensions CM. To speed up transfers, check that they are item types that can and will be compressed.

```
//COMPASM EXEC PGM=IEV90,PARM='...'
//*
//*SBEM TGT OBJECT
//*SBEM TGT SYSPRINT Assembly_Listing
//…
//SYSPRINT DD DISP=SHR,DSN=%CURDIR..ASMLIST(%SRCMOD.)
```

In the example above, the first `//*SBEM TGT` identifies a primary target for the whole build step. The second line, with the comment `Assembly_Listing`, specifies a secondary target, which is a listing type. The `Assembly_Listing` tag is not syntax checked, its existence signals to the SBEM that it is a listing target.

Using Mixed Inputs

In the examples in the previous section it is assumed that all the inputs in the `DMINPUT` array are the source numbers that you require. If there are other inputs in the configuration of a different type, such as a configuration named side-file or `DBRM`, the loops around `DMINPUT` have to be enhanced to only select entries of the correct type.

Even though this is a link style collection template you can use patterns from compilation templates. For example, if the input array contains a single side file of the type `SYSDEFSD` that is needed in the link, its name can be found from the `MDHDSN` function, as used in the compilation examples:

```
)SET SIDE=_MDHDSN_(,DMINPUT,SYSDEFSD,DMPATH)_
```

Build Step Warning Messages

When an MVS build job executes, each build step causes a template to be expanded and passed to the SBEM for execution. The SBEM then executes each step of the task. If you specify the build symbol DM_BUILDER_PROGRESS_REPORTING, any step that has a non-zero return code causes the SBEM to issue the following message with a severity of W:

```
MDHSB16D112W Step <step number> Program <program> returned COND CODE = <cc>
```

'W' type messages are sent to the log and are signalled as an alert.

TIP If you name the steps carefully in your templates you can give users a clear understanding of the meaning of the warnings.

Computing the Final Return Code

A build step is the execution of a single build rule and a JCL step is executed in a build step. There can be multiple JCL steps in a single build step. For example, a compilation template might:

1 Compile the source.

2 Write the compilation listing to the spool for local inspection.

3 Link edit it NCAL[1].

4 Write the link edit listing to the SPOOL for inspection.

MDHNASC0, the Assembler sample template that ships with Dimensions CM, has four JCL steps required for the execution of a single build step. Only the first and third JCL steps are significant for the final return code.

You can combine the individual JCL steps of a single build step in various ways to calculate the final return code. The SBEM has a final return code for a stream, which starts at 0. As each step is executed, the final return code is modified in accordance with the setting on an SBEM directive for the current step. The return code can have one of the following forms:

```
//*SBEM CDE MAX
//*SBEM CDE IGNORE
//*SBEM CDE FAIL
```

where:

- MAX: Chooses the maximum of the return code from this JCL step, considered with the stored return code for the whole build step.
- IGNORE: Ignores the return code from this JCL step.
- FAIL: Cancels the execution of the whole build step if the return code from this JCL step is non-zero.

NOTE The aggregated return code is used for R= computation at the time of processing the //*SBEM END statement.

1. There are two possible intermediate forms for the output from a compiler. The most basic is object format, however using the binder you can create equivalent NCAL objects that link faster.

Creating a Bill of Materials from the SBEM

To build the Bill of Materials you must provide the SBEM with information during the build process to indicate the objects of interest, and to control the success or failure of the build steps. There are several classes of information which a BOM can communicate: primary source modules, secondary dependencies, and primary and secondary targets. You use SBEM directives on each relevant JCL step the SBEM executes to describe the type of information to monitor. The BOM treats the collection of all recorded information from all the JCL steps for which monitoring is used as the product of this build step. Duplicates are removed. Input objects detected during monitoring that are not in the search path are also removed.

The SBEM detects input and output files for the BOM under the control of directives that start with //*SBEM. Most SBEM directives are DDNAME based and appear after the EXEC statement to which they relate. The directives SRC, DEP, TGT, TFP and OTH should appear before any DDNAMES for that step.

DEP

```
//*SBEM DEP DDNAME
//*SBEM DEP DDNAME (*)
```

Creates a dependency. This is a file that the current process needs so that it can work, such as a copy book used during a Cobol compile. You can manually add explicit dependencies, in which case the DDNAME must refer to a readable file (rather than a container) and multiple DEP statements are allowed.

If you use (*) the SBEM monitors disk activity on that DDNAME and looks for open actions for particular members.

You can monitor multiple ddnames.

On a link step, //*SBEM DLP SYSLIB(*) monitors auto called libraries.

SRC

```
//*SBEM SRC DDNAME
//*SBEM SRC DDNAME (*)
```

Identifies the primary input source for this build step. The first form involves no monitoring but looks at the JCL used for this step, picks up the ddname corresponding to the one you have specified, and sets the primary input to the data set name used. If you are compiling a library member, this should be a data set name with a member specified. By this stage the search path resolution has been performed so that this becomes the name where the input is located even if it is higher up the search path.

The second form is not usually used for SRC, but will monitor the specified concatenation (named by the ddname) for any input modules and will save all modules referenced by this path as SRC.

TGT

```
//*SBEM TGT DDNAME
//*SBEM TGT DDNAME listing-type-name
//*SBEM TGT DDNAME (*)
//*SBEM TGT DDNAME (*) listing-type-name
```

Use one or more of these forms. The first two are normal. The first and third forms are used for target of the build step. The second and fourth forms are used to track listings, such as targets, which usually are not the focus of the build step. The

listing_type_name must be non-empty for a listing target but otherwise has no meaning.

You can identify two different outputs from a build step as the primary targets. For example, a compile might produce a target of type .OBJ and a target of type .LNKLIB, using statements such as:

//*SBEM TGT SYSLIN

//*SBEM TGT SYSLMOD

from different JCL steps in the same build template (which the shipped templates do). In the above example, SYSLIN is the output of the compiler/assembler processing, and SYSLMOD is the output from the NCAL Link edit step. The two modules are equivalent, but in different formats.

TFP

```
//*SBEM TFP DDNAME [(*)] [listing-type-name]
```

Use TFP with fully footprinted targets to trigger full footprint reporting.

OTH

```
//*SBEM OTH DDNAME [listing-type-name]
//*SBEM OTH DDNAME(*) [listing-type-name]
```

OTH is similar to TGT except that it does not create made-of relationships. This is useful for co-targets where the additional relationships can easily be understood, for example DBRMs, or are not required by your processes.

Source Types

Many MVS environments use a single low level qualifier for all sources. This is not recommended practice but it may work. However, you are constrained to using build types (the Dimensions FORMAT system attribute) to distinguish between the variety of sources.

Mapping between MVS and Dimensions CM

The relationship between MVS source names and UNIX style path naming used by Dimensions CM is not obvious. Dimensions Build makes the following assumptions about this mapping:

- All the sources are stored in MVS libraries.
- '/' separators in workset paths are translated to '.' between MVS qualifiers.
- The low level qualifier of a data set name is also a folder in Dimensions and has the same value as the file's extension. The file name is the member name.
- Valid source element names are those that can be represented in JCL without the use of ' in the DSN= statement. All folder names, ignoring case, must conform to the rules for qualifiers. All member names, ignoring case, must conform to the rules for member names in JCL. No member names can be longer than eight characters and you cannot use embedded nulls or spaces.

Example:

On the MVS side the working location is SYSDEV.PROJSRC.

Dimensions Name	MVS Name
Cobol/A1.cobol	SYSDEV.PROJSRC.COBOL(A1)
A1.COBOL	Results are unpredictable and may vary: ■ No container object ■ Arrives as a sequential data set
LOAD/FRED	Results are unpredictable and may vary: ■ No extension ■ Stored at SYSDEV.PROJSRC.LOAD(FRED) and then back-loaded as LOAD/FRED.LOAD
LOAD/FRED.LOAD	SYSDEV.PROJSRC.LOAD(FRED)
Subproj/ASM/mycode.ASM	SYSDEV.PROJSRC.SUBPROJ.ASM(MYCODE)
LOAD.ANYTHING.ASM/E.ASM	Not valid, results are unpredictable and may vary and the qualifier is too long.
LOAD.ANY.ASM/E.ASM	Results are unpredictable and may vary: ■ Stored at SYSDEV.PROJSRC.LOAD.ANY.ASM(E) ■ System will then expect to put it back as LOAD/ANY/ASM/E.ASM
SUBPROJ/LOAD/FRED.LOAD	SYSDEV.PROJSRC.SUBPROJ.LOAD(FRED)
$A/FOO.$A	SYSDEV.PROJSRC.$A(FOO) Works with US, NZ, AU code pages but not UK, FR etc.
@/JOHN.@	SYSDEV.PROJSRC.@(JOHN) Works with US and UK code pages but not in France.
MY-NAME.COBOL/A.COBOL	SYSDEV.PROJSRC.MY-NAME.COBOL(A) The hyphen character is allowed in file names on MVS.
1FOLDER/COBOL/MY.COBOL	Not valid, '1' is not allowed as a leading character for a qualifier.

Using REXX in a Build Template

Although the UNIX REXX environment gives efficient access to HFS files, the best way to get to data sets is to use an "MVS" environment that gives access to EXECIO. The following example template JCL shows how to do this (refer to the annotated version below for details):

```
//*
//* <<<<< REXX START
//* <<<<< REXX START
)SET SERENA=_MDHDSN_(,DMINPUT,COB,DMPATH)_
)SET SERENA=_MDHEXTRACT_(+d,%SERENA.)_
)EXPAND %%s
/* rexx */
say ")EXPAND"
say "//* Hi from rexx" date()
call bpxwdyn "alloc fi(stu) da('%SERENA') shr reuse"
address mvs 'execio * diskr stu (fini stem s.'
say ")VECTOR LINES("s.0")"
say ")SET LINECOUNT = "s.0
    do i=1 to s.0
    say ")SET LINES("i-1") = "strip(s.i)
    say "//*" i-1") = "left(s.i,20)
    end
exit 0
)ENDEXPAND
)EXPAND
//* <<<<< REXX END
//* <<<<< REXX END
//*
```

Annotated version

1
```
//*
//* <<<<< REXX START
//* <<<<< REXX START
```
The top of the template.

2
```
)SET SERENA=_MDHDSN_(,DMINPUT,COB,DMPATH)_
```
Set the variable SERENA to the filename of the Cobol input file. Rather than hard code the name, use the inbuilt facilities for finding it. The above statement searches for an entry of type COB in the array DMINPUT. (If your type is different, you'll have to change this.)

Assuming the statement finds that one of your inputs is indeed of type COB, it will then search for this in the search path array DMPATH. This is important when reading in files like this, as you don't know in advance where the build is going to find the member; for example, it may be in RELEASE.

NOTE

The above statement is also available in UNIX and Windows templates, and works in a similar way.

3 `)SET SERENA=_MDHEXTRACT_(+d,%SERENA.)_`

The return from this function is a name suitable for use in UNIX allocates, which means it will have `'//'` at the front, which we don't want. The above statement "normalizes" the name into regular JCL syntax.

If you do not have a file of type `COB` as an input to this step, or it cannot be found on disk, then the result is the string `<NULL>`, which you could test for if you wanted. This string will give an JCL error (which is good, as you want the build to fail in this case).

NOTE

The above statement is also available in UNIX and Windows templates, and works in a similar way.

4 `)EXPAND %%s`

Start an `EXPAND` block that specifies a script (in this case it is just the name of the file itself, `%%S`, as this will be executable in its own right). The double `'%%'` is to protect the `'%'`from template expansion. (%% becomes %).
This must not appear inside of a `)REP`, as that doesn't work.

5 `/* rexx */`

Start of the Rexx itself.

6
```
say ")EXPAND"
say "//* Hi from rexx" date()
```

Get Rexx to emit the `')EXPAND'` instruction. This means that when the templater processes the output from Rexx, it will expand it as if it were further template lines, rather than just literal text.

7 `call bpxwdyn "alloc fi(stu) da('%SERENA.') shr reuse"`

Use `BPXWDYN` as a stand in for the TSO allocate command (remember that we are not in a TSO environment at this point.). This is basically a wrapper around the IBM SVC99 `dynalloc` function. The template variable SERENA is set above to be the full data set name of the input file.

NOTE

Documentation for `BPXWDYN` (the IBM Dynamic Allocation Text interface) can be found at:
ftp://ftp.software.ibm.com/s390/zos/tools/bpxwdyn/bpxwdyn.html

8 `address mvs 'execio * diskr stu (fini stem s.'`

The DDNAME (stu) is now pointing at the COB source file. This is read into a rexx stem variable using a facility that emulates the TSO EXECIO command.

The rexx stem called 's' now contains lines from the file. You can do whatever we like with this data. In the remainder of this example, it is demonstrated how you would put this data into some templater variables, so the lines would be available direct from JCL using a)REP command. At this point, you could do any rexx based logic you liked, and set flag variables in a similar fashion.

9 `say ")VECTOR LINES("s.0")"`

Creates a template array called LINES of the right size.

10 `say ")SET LINECOUNT = "s.0`

Creates a template variable containing the size, in case that is needed as well.

11 `do i=1 to s.0`

Loop round each line of the file.

12 `say ")SET LINES("i-1") = "strip(s.i)`

Assign an individual line. Template arrays start at 0; rexx arrays start at 1.

13 `say "//*" i-1") = "left(s.i,20)`

Optional debug line. This provides some visual confirmation in the output JCL that rexx actually did something.

NOTE

It would be advisable to omit this line if the input is a large Cobol program.

14 `end`

End of loop.

15 `exit 0`

End of rexx.

16 `)ENDEXPAND`

End of expand block.

The procedure is now back in templater mode, having finished with rexx. It is also now in non-expand mode, as a by-product of the above statement.

17 `)EXPAND`

Turn expand mode back on to enable further template syntax to be used.

18
```
//* <<<<< REXX END
```

```
//* <<<<< REXX END
```

The JCL template can now be continued, but now there are more variables available for use.

20
```
//*
```

```
//* Cobol source is %LINECOUNT. lines long
```

```
//*
```

An example of using the new knowledge about the build input file.

Invoking the REXX Interface

For performance reasons, Serena recommends using the interface to invoke REXX. However, you can still use REXX as described above.

```
)SCRIPT MDHDREXX.REXX
..
..
..
)ENDSCRIPT
```

Usage rules:

- Instead of "say" use "call mdhsay".
- Instead of "exit", use "return”.
- Do not define any REXX signal handlers.
- If you define a procedure, add "expose (mdhexpose)" to the definition.
- If you need to send a MsgX message, use "call mdherr 'MDHREX4700001E ...'"

Example This example turns http://www.something.com/remaining into just "something"

```
)SET URL=http://www.serena.com/splat/bim/bam
)SCRIPT MDHDREXX.REXX
url="%URL."
parse var url "http://" . "." company "." .
call mdhsetvar "COMPANY",company
)ENDSCRIPT
COMPANY = %COMPANY.
```

Example This example reverses the slashes, "/" to "\" in an array:

```
)VECTOR DMPATH(5)
)SET DMPATH(0)=/a/1/x/fo.c
)SET DMPATH(1)=/b/2/x/fo.c
)SET DMPATH(2)=/c/3/x/fo.c
)SET DMPATH(3)=/d/4/x/fo.c
)SET DMPATH(4)=/e/5/x/fo.c
)SCRIPT MDHDREXX.REXX
i=0
)REP DMPATH
tmp = "%DMPATH."
tmp=translate(tmp,"\","/")
call mdhszy ")SET DMPATH("i")="tmp
i=i+1
)ENDR
)ENDSCRIPT
)REP DMPATH
%DMPATH.
)ENDR
```

Controlling Simultaneous Conflicting ENQs in Data Sets

In Dimensions CM 12.2.2 and earlier, if a data set allocation was unable to proceed because it was allocated exclusively to another task, the SBEM allocation would always fail. The facility described below allows an installation to emulate JES2 'Waiting for data set' processing.

This facility is available if your SBEM is authorized. Users also require the rights to access the authorized features. You can control what features are available using MDHTDCFG variables. Refer to the *IBM MVS Authorized Assembler Guide* for details of the FLGS2 word in SVC99 (DYNALLOC) processing. Note that the use of this facility can lead to a deadlock on your system if used improperly, and that operator prompts issued by MVS enable recovery.

There are three separate bits in FLGS2 that users can partially control:

- (Normal use) Wait for data set (W4D)
- Wait for unit (W4U)
- Wait for volume (W4V)

You can define these variables in MDHTDCFG:

Variable	Default Value (HEX)
DM_MVS_SBEM_FLGS2_INITIAL_MASK	0
DM_MVS_SBEM_FLGS2_FINAL_AND	0xDFB00000

There are three //*SBEM directives:

- W4D
- W4U
- W4V

The initial FLGS2 word is set from DM_MVS_SBEM_FLGS2_INITIAL_MASK. So, to always wait for data set ENQs, set this variable to 4000000.

If you code //*SBEM W4D on any step, then for that step only the 40000000 bit is set in the FLGS2 word. In the same way, for W4U and W4V the related bits are set.

DM_MVS_SBEM_FLGS2_FINAL_AND is ANDed with the working FLGS2 mask. This is set by default to stop any setting of unsuitable bits in this word. An installation could use this word to turn off W4D, W4U, or W4V processing altogether or for any combination of these bits.

Chapter 12

Openmake Templates

Introduction 294

Variables 295

Introduction

NOTE

- For information about the templating language and syntax see page 185.
- For information about Dimensions Build see the *Dimensions CM Build Tools User's Guide.*
- For information about using Openmake see the documentation that was installed with the product.

To support the integration between Dimensions Build and Openmake, Serena® Dimensions® CM ships with the following Openmake build templates:

- MVS: `TEMPLATE(MDHBOMT0)`
- Windows and UNIX: `om_build.template`

The default template directory is `<install root>/templates`.

The integration flow between Dimensions Build and Openmake is as follows:

1 The build configuration, and other parts of Dimensions CM, make available variables that describe the build that is to be executed.

2 These variables are used by the Openmake templates to construct commands to invoke the two Openmake utilities, `bldmake` and `om` (note that the Openmake knowledgebase server must be set up correctly).

3 When the Openmake utilities have completed (the template has finished executing), the build steps created by Openmake are run. You can monitor these steps from the Openmake knowledgebase server.

4 The final `om` build step communicates to Dimensions Build that the build has completed.

5 By this stage Openmake build steps have created a BOM (bill of materials) containing a description of all the inputs and outputs in the build. The name of this file is passed back to Dimensions Build so that the BOM to be picked up, parsed, and the relevant target items collected into Dimensions CM.

In the flow above the BOM is created after the entire build submitted by `om` has completed. These jobs are run asynchronously with respect to Openmake.

Variables

The input to the Openmake template is a set of variables. The table below lists the variables and their source. Some of the variables are unique to Openmake and are provided by the build configuration as area or configuration symbols. Most of the variables are described in one of the following documents:

- Chapter 7, "The Templating Language and Processor" on page 185.
- The *Serena Dimensions CM System Administration Guide.*

Key to the abbreviations used in the table

- Platform:
  - *—all
  - M—MVS
  - U—UNIX
  - W—Windows
- Optional: optional variable or symbol
- Source:
  - config—the build configuration
  - build—created by the build server
  - cfg—dm.cfg (the Dimensions configuration file)
  - sys—provided by the system
  - out—output (a variable created for use by the caller).

Variable or Symbol	Platform	Optional	Source	Description
ANT_HOME	U	Yes	config	The PATH and ANT_HOME environment variables
DM_CURRENT_USERNAME	M		build	The current Dimensions Build user (used for the Openmake knowledgebase server log in).
DM_LICENSE	WU	Yes	cfg	Environment variable for SERENA_LICENSE_FILE
DM_META_DATASET	M		cfg	MVS listener metadata
DM_META_LOCK_MAJOR	M		cfg	MVS listener metadata
DM_META_LOCK_MINOR	M		cfg	MVS listener metadata
DM_OM_OSPLATFORM	W		config	Control file (-f option) for Openmake
DM_OPENMAKE_INSTALL	*	Yes	cfg	Default fallback for DMOMINSTALL
DM_OPENMAKE_SERVER	*	Yes	cfg	Default fallback for DMOMSERVER
DM_ROOT	WU		cfg	Dimensions root installation
DM_TPLB_MACHINE	M	Yes	cfg	Default fallback for DMMACHINE
DM_TPLB_PUBLIC	M	Yes	cfg	Default fallback for DMPUBLIC

Variable or Symbol	Platform	Optional	Source	Description
DMBLDLOGURL	U		out	Openmake URL
DMBLDMAKE_OPTIONS	*	Yes	config	bldmake options
DMBLDPROJ	*		config	Openmake project name
DMCDHLQ	M		cfg	High level qualifier from the installation
DMCDILQ	M		cfg	Intermediate level qualifier from the installation
DMCDLPA	M		cfg	LPA mode
DMCLEANBUILD	M		cfg	Request cleanup
DMDAY	*		sys	Date (day)
DMDIRECTORY	*		sys	Build location
DMHOUR	*		sys	Date (hour)
DMINHLQ	M		cfg	Instance HLQ
DMINST	M		cfg	Instance mid level qualifier
DMMACHINE	*		build	Machine name (DNS) of node
DMMICROSEC	*		sys	Date (microseconds)
DMMINUTE	*		sys	Date (minute)
DMMONTH	*		sys	Date (month)
DMMVSUSR	M		sys	MVS system username RACF
DMOBOMRPT	*		out	BOM location
DMOMDATETIME	*		build	bldmake `-ld` parameter
DMOMINSTALL	*	Yes	config	Openmake installation path
DMOMSERVER	*	Yes	config	Openmake knowledgebase server address
DMOPTIONS	*	Yes	config	om options
DMOSPLATFORM	U		config	Control file (`-f` option) for Openmake
DMPBEMNODE	*		sys	TCP/IP node address of controlling PBEM
DMPBEMPORT	*		sys	TCP/IP port the `PBEM` is listening on
DMPUBLIC	*		config	Public build job (true or false)
DMRUNMODE	UW		sys	SYNC or ASYNC
DMSEARCHPATH	*		config	Openmake search path name
DMSECOND	*		sys	Date (seconds)
DMSECONDS	*		sys	Date (seconds)
DMSEQ9	*		sys	Build sequence number
DMSTEP	*		sys	`BRD` step number in a build
DMSYSUSR	M		sys	System user
DMTARGET	*		config	Array of target objects
DMUNIQUE	*		sys	Unique identifier for a run

Variable or Symbol	Platform	Optional	Source	Description
DMUSER	*		sys	Dimensions CM user
DMYEAR	*		sys	Date (year)
JAVA_HOME	U	Yes	config	Dimensions environment variable

Chapter 13

Remote Job Execution Templates

Introduction	300
Templates	300
Simple Synchronous Templates	300
Asynchronous Remote Jobs	302
Remote Job Template Example	303

Introduction

The Serena® Dimensions® CM REXEC command creates a remote job on a Dimensions CM node. You can view the details of remote jobs in the Remote Jobs cluster in the administration console, for details see the *System Administration Guide* or the administration console help.

You can also manage remote jobs from the command line using the commands REXEC, RDEL, and RSTAT, for details see the *Command-Line Reference*.

Templates

Remote job execution templates are expanded and executed on the node specified in the REXEC command. The templates use the syntax and variables described in "The Templating Language and Processor" on page 185. However, these are not build templates and build specific variables are not available.

Remote job execution templates are located in the system directories specified by the remote node dm.cfg variable DM_TEMPLATE_CATALOGn. For details about this variable see page 227.

Parameters

The REXEC system delivers a set of variables to the template that contain information about the context of the request. The system can also deliver additional parameters that you can supply using the /PARAM option of the REXEC command. You can use this technique to define remote subroutines and supply them with varying arguments at run time.

Parameters that you supply can be single values or arrays, for example:

- Template test.tpl:

```
echo VALUE is %VALUE.
)REP INPUT
echo "array line is %INPUT." >>c:\temp\foobar.log
)ENDR
```

- Invocation using REXEC:

```
REXEC /template=test.tpl /network=node /
    param="VALUE=hello,INPUT=[one,two,three]"
```

Simple Synchronous Templates

The /NOBATCH and /NOCAPTURE options of the REXEC command create simple synchronous templates. These templates run quickly, the REXEC command waits for completion, and the return code is available and reported. A remote job entry is not created in the administration console as the templates will have already finished. Simple

synchronous templates do not connect back to the Dimensions CM server, and you cannot use MVS batch JCL.

Feedback

Synchronous templates generate two types of feedback:

- Numeric return code is reported and corresponds to the exit status of the script. `0` is considered success.
- An error string is also collected and is reported at the command prompt in the event of a failure. You can use error strings to communicate template expansion time errors. To see this error message, the return code has to be non-zero.

The example below is a synchronous template that returns an expansion time error message:

```
)SET DMERRMSG=Current user is %DMREMOTEUSER.
exit 5
```

When this template is executed from the `REXEC` command the message is displayed as feedback. You can use this feature for messages other than error messages, as shown in the example above. The template expansion process can use shell or Perl scripts therefore you can make it as complicated as required.

You can also use this feature to send back listings from the remote node. The template can allocate `DMERRMSG` as an array, and fill it with lines of output.

The example below is a synchronous `REXEC` template that returns the environment variable PATH list in the `DMERRMSG` array to be displayed at the command prompt. This is a UNIX example that uses an embedded Perl script during the template expansion time.

```
)VECTOR DMERRMSG(20)
)EXPAND perl %%s
 print ")EXPAND\n";
 @dirs = split(/:/,$ENV{'PATH'});
 $i=0;
 foreach $dir (@dirs)
   {
    print ")SET DMERRMSG($i) = $dir\n";
    $i=$i+1;
   }
)ENDEXPAND

)EXPAND
exit 5
```

The following command transcript shows an `REXEC` command from a Windows system executing the above template on a Linux node. Note that inside the message feedback from `REXEC` is the array of `PATH` locations in the UNIX host.

```
C:\temp>dmcli
Serena Dimensions CM 10.1.0 FT1 at 20:32:08 Wednesday 09 August 2006
Copyright (c) 1988-2006 Serena Software, Inc. All rights reserved.
Dimensions>auth /network=suse /user=xxx /password=xxxx
Operation completed
Dimensions>rexec /nobatch /nocapture /network=suse /template=t.txt
execute expanding script, cmd= perl /tmp/tpl11832-1.sh
Passing the template for execution
```

```
cmd = /bin/sh "/tmp/tpl21832-2.sh"  >/dev/null 2>/dev/null
/opt/serena/changeman/dimensions/10.1//prog
/usr/bin
/bin
.
/opt/oracle/product/10g/bin
Template processing complete and command submitted: user result code
    1280
Job R-4197028 FAILED - rc(1280) errno(0)
Operation completed
Dimensions>
```

The return code above (1280) is 0x500. This is the UNIX shell exit code in the high word of the result code, and importantly it is not zero.

The DMERRMSG variable is relevant during template expansion time and not during script execution time. In the example above the Perl script does run at expand time because it is an inline script and not an execution script. For details about template expansion see page 217.

Asynchronous Remote Jobs

The /BATCH option of the REXEC command creates an asynchronous process on the remote node. The REXEC call returns immediately and the process is left running. The /CAPTURE option causes the generation of the DMCERTIFICATE variable, which allows the template to connect back to Dimensions CM to report progress. The template can use the session established with the certificate to perform any action, such as execute CI commands to move files into Dimensions CM control.

The command most associated with a template connecting back to Dimensions CM is RSTAT. RSTAT allows the script to update the job table with status information allowing you to see how the asynchronous process progressed. Return code information and a message can also be reported. If a local file name is supplied it is collected by Dimensions CM and loaded as messages into the job log display in the administration console. You can issue the RSTAT manually to update a job, but normally it is issued by a template. To use the certificate, a template can use the dmcli program with the -cert option to log in to the originating server.

The MVS JCL equivalent uses the batch command program, MDFLCMD. However, the certificate on MVS is longer than 80 bytes. To use it with JCL, use the slice option to split it into an array suitable for use as instream JCL input.

The examples below show a method of connecting back to Dimensions CM from a template, performing actions, and then using RSTAT:

- This Windows example uses a command string directly on the dmcli command line:

```
)IF %DMASYNC. = YES
dmcli -cert "%DMCERTIFICATE." -cmd "rstat %DMJOBID. /
    description=\"example rstat command\" "
)ENDIF
exit 0
```

- This example uses MVS JCL. Note the use of DMNODE. An automatic AUTH to this node is performed by the certificate handling mechanism:

```
//CALLB   EXEC PGM=MDFLCMD,
// PARM=('POSIX(ON),ENVAR(""_CEE_ENVFILE=DD:DIMVARS"")',
// '/')
//STEPLIB  DD DISP=SHR,DSN=%DMLLIB.
)IF %DM_USE_LPA. = NO
//         DD DISP=SHR,DSN=%DMLLPA.
)ENDIF
//DIMVARS  DD DISP=SHR,DSN=%DMDIMVARS.
//CEEPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//LOGIN    DD DATA,DLM=ZZ
/CERTIFICATE
)REP DMCERTIFICATEPARTS
  %DMCERTIFICATEPARTS.
)ENDR
ZZ
//COMMAND  DD *
RSTAT %DMJOBID. -
  /STATUS=SUCCEEDED -
  /RC=0 -
  /DESCRIPTION="Job has run to completion" -
  /RESPONSE_FILE="DMNODE::DATASET.NAME(MEMBER)"
/*
```

Remote Job Template Example

The example below pulls a remote directory into Dimensions CM. The command is sent by REXEC to a remote machine, with parameters that describe the directory (for example, `c:\temp`) and file pattern (for example, `*.txt`) that are required. The remote script connects back to Dimensions CM to issue CI commands, and then uses RSTAT to update the remote job status.

After issuing the command, the results are displayed in the administration console and the output log from the individual Dimensions CM commands is available in the Job Details/Job log view.

This is the directory on the remote machine:

```
C:\temp>dir
 Volume in drive C has no label.
 Volume Serial Number is A8CE-8883

 Directory of C:\temp

08/10/2006  03:09 PM    <DIR>          .
08/10/2006  03:09 PM    <DIR>          ..
08/10/2006  12:30 PM               131 a.txt
08/10/2006  10:42 AM                34 list.txt
08/10/2006  02:15 PM               125 log.txt
08/10/2006  10:55 AM               417 out.txt
08/10/2006  11:36 AM               198 stupl1.pl
```

```
08/10/2006  02:15 PM               203 stupl2.pl
08/10/2006  09:53 AM            21,951 testlog.txt
08/10/2006  12:09 PM                79 zlist2.txt
08/10/2006  12:09 PM                 0 zlist3.txt
08/10/2006  12:08 PM                20 ztest1.bat
              10 File(s)         23,158 bytes
               2 Dir(s)   2,790,297,600 bytes free
C:\temp>
```

The REXEC command is issued from another machine:

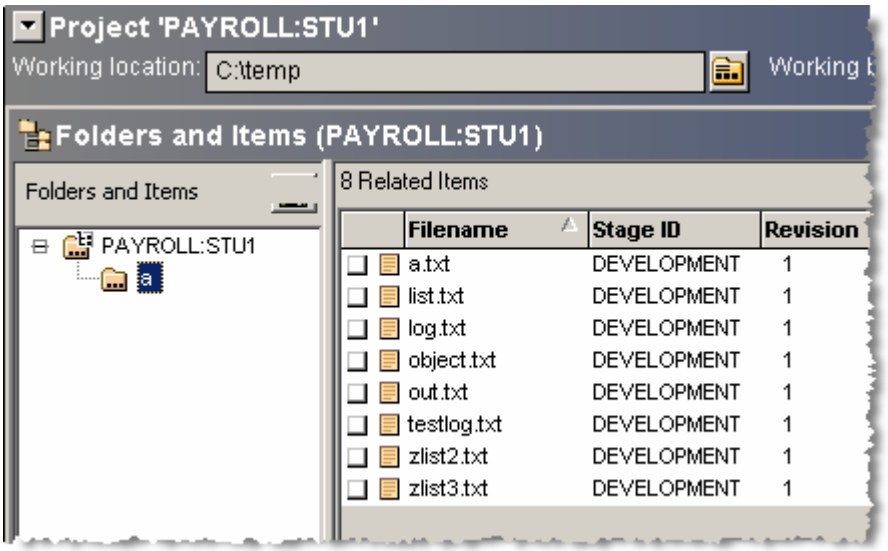
```
Auth /network=nodename /user=xxxx /password=xxxxx
rexec /network=nodename /template=grabdir.tpl /capture /
    PARAM="P_DIR=c:\temp,P_PAT=*.txt,P_PROD=PAYROLL,P_PART=PAYROLL"
```

The REXEC returns. Inspection of the remote job queue shows that the job has finished and additional log records are available:

Show Logs

Item	Log Type	Log
0		Serena Dimensions CM 10.1.0 FT1 at 15:08:37 Thursday 10 August 2006
0		Copyright (c) 1988-2006 Serena Software, Inc. All rights reserved.
0		CI "payroll:item0 txt.a-src" /keep /format=src /filename=item0 /part=payro
0		Item PAYROLL:ITEM0 TXT.A-SRC;1 has been forwarded to DMSYS
0		Operation completed
0		CI "payroll:item1 txt.a-src" /keep /format=src /filename=item1 /part=payro
0		Item PAYROLL:ITEM1 TXT.A-SRC;1 has been forwarded to DMSYS
0		Operation completed
0		CI "payroll:item2 txt.a-src" /keep /format=src /filename=item2 /part=payro
0		Item PAYROLL:ITEM2 TXT.A-SRC;1 has been forwarded to DMSYS
0		Operation completed
0		CI "payroll:item3 txt.a-src" /keep /format=src /filename=item3 /part=payro
0		Item PAYROLL:ITEM3 TXT.A-SRC;1 has been forwarded to DMSYS
0		Operation completed
0		CI "payroll:item4 txt.a-src" /keep /format=src /filename=item4 /part=payro
0		Item PAYROLL:ITEM4 TXT.A-SRC;1 has been forwarded to DMSYS

Inspection of Dimensions CM shows that the items have been created:



The `grabdir.tpl` template shown below is for Windows, though you can port it easily to UNIX. Although this is a Windows example, it uses Perl. To try this example, make sure you have a version of Perl in your path. Note that there is a `dm.cfg` symbol on the server controlling how many lines of log output can be attached to a remote job. Set DM_REXEC_RESPONSE_MAX_LINES to a small number (about 50) to ensure that rouge scripts do not fill the database with output.

```
)CM
)CM ============================================================================
)CM grabdir
)CM ============================================================================
)CM
)CM Example REXEC template
)CM
)CM Windows version
)CM
)CM 1. Lists contents of a given directory with a DOS pattern
)CM 2. For each file, creates an item in Dimensions, using supplied parameters
)CM 3. Updates remote job, and supplies the trace as a response file.
)CM
)CM Parameters:
)CM
)CM   P_DIR  - directory on remote node, ending in "\"
)CM   P_PAT  - pattern (dos format)
)CM   P_BASE - base name to generate new items
)CM   P_PROD - Product name
)CM   P_TYPE - item type
)CM   P_FORM - item format
)CM   P_PART - design part
)CM   P_WSD  - Workset directory. If present, should end in a "/"
)CM   P_CLEAN- delete working files
)CM
)CM ============================================================================
)CM

)SET DMSAVESTDOUT=Y
)SET DMSAVESTDERR=Y
)SET DMCOMBINEOUTERR=Y


)CM
)CM ============================================================================
)CM default the parameters
)CM ============================================================================
)CM

)IFNDEF DM_TMP
```

```
)SET DM_TMP=c:\temp
)ENDIF

)IFNDEF P_DIR
)SET P_DIR=c:\temp\
)ENDIF

)IFNDEF P_PAT
)SET P_PAT=*.txt
)ENDIF

)IFNDEF P_BASE
)SET P_BASE=auto
)ENDIF

)IFNDEF P_PROD
)SET P_PROD=payroll
)ENDIF

)IFNDEF P_TYPE
)SET P_TYPE=txt
)ENDIF

)IFNDEF P_FORM
)SET P_FORM=src
)ENDIF

)IFNDEF P_PART
)SET P_PART=payroll
)ENDIF

)IFNDEF P_WSD
)SET P_WSD=
)ENDIF

)CM
)CM ===========================================================================
)CM create our perl script
)CM ===========================================================================
)CM

)EXPAND type %%s > "%DM_TMP.\perl-%DMUNIQUE..pl" 2>NUL:
    $PDIR=$ARGV[0];
    $PTMP=$ARGV[1];
    $i=0;
    while (<STDIN>)
    {
        $line=$_;
        chomp $line;
        print "CI \"%P_PROD.:%P_BASE.$i %P_TYPE..a-%P_FORM.\" /keep /format=%P_FORM. /filename=%P_BASE.$i /
part=%P_PART. /ws_filename=\"%P_WSD.$line\" /user_filename=\"$PDIR$line\" \n";
        $i=$i+1;
    }
    print "RSTAT %DMJOBID. /description=\"grabdir example template\"  /status=succeeded /rc=0 /
response_file=\"$PTMP\\log-%DMUNIQUE.txt\" ";
)ENDEXPAND
)EXPAND

)CM
)CM ===========================================================================
)CM create the input file for the perl script
)CM ===========================================================================
)CM

)EXPAND "%%s" >NUL: 2>NUL:
dir %P_DIR.%P_PAT. /B > "%DM_TMP.\input-%DMUNIQUE..txt"
)ENDEXPAND
```

```
)EXPAND

)CM
)CM ===========================================================================
)CM create the Dimensions commands
)CM ===========================================================================
)CM

)EXPAND "%%s" >NUL: 2>NUL:
perl "%DM_TMP.\perl-%DMUNIQUE..pl" "%P_DIR.\" "%DM_TMP.\" < "%DM_TMP.\input-%DMUNIQUE..txt" > "%DM_TMP.\output-
%DMUNIQUE..txt"
)ENDEXPAND
)EXPAND

)CM
)CM ===========================================================================
)CM EXECUTE
)CM ===========================================================================
)CM

dmcli -cert "%DMCERTIFICATE." < "%DM_TMP.\output-%DMUNIQUE..txt" >"%DM_TMP.\log-%DMUNIQUE.txt" 2>&1

)CM
)CM ===========================================================================
)CM cleanup
)CM ===========================================================================
)CM

)IFDEF P_CLEAN
del "%DM_TMP.\perl-%DMUNIQUE..pl"
del "%DM_TMP.\input-%DMUNIQUE..txt"
del "%DM_TMP.\output-%DMUNIQUE..txt"
del "%DM_TMP.\log-%DMUNIQUE.txt"
)ENDIF

exit 0
```

Chapter 14

Deployment Area Scripts

Introduction	310
Variables	312
Differences in Relation to Build Templates	314
Debugging	314
Other Features	314
Specifying User Attributes in Deployment Area Scripts	315
Example	317

Introduction

NOTE For information about the templating language and syntax see Chapter 7, "The Templating Language and Processor" on page 185.

You can attach the following types of templates to a deployment area:

- Pre-event transfer script
- Post-event transfer script
- Fail-event transfer script

Use these scripts when you require detailed control of a deployment, for example, to insert rows into an external database, or to perform a bind operation on a deployed object. The pre and post event scripts enable you to control the timing of these operations. In the event of a failure the fail-event transfer script enables you to roll back any external changes that you made.

NOTE

- As of Serena® Dimensions® CM 10.1.1, the Dimensions CM server runs the scripts on the node hosting the area in bulk once per command, as opposed to once per item, which was the case for previous Dimensions CM releases. As before, when a `DPI` command results in a single item transfer into one area, then transfer scripts will be executed before and after the item transfer into the area. However, when a `DPR` or a `DPB` command results in multiple item transfers into one area, then:
  - The pre-event transfer script will be executed once before any items are transferred into the area.
  - The post-event transfer script will be executed once after all items are successfully transferred into the area.
  - The fail-event transfer script will be executed once if there were any errors transferring items into the area or the post script failed to execute successfully.

  Bulk script execution results in significant improved deployment performance.
- You can store area scripts in your Dimensions repository. However, the preferred location is on the node where the deployment area is located, in the system template folder specified by the Dimensions configuration file (dm.cfg) symbol `DM_TEMPLATE_CATALOGn`. For details about this variable see page 227. Note that this is the configuration file that belongs to the node and not to the Dimensions server.
- Areas scripts are implemented differently from build templates, which you can execute from a build area using search paths.

Changes Introduced in Dimensions CM 12.2.1

Changes to deployment scripts were introduced in Dimensions CM 12.2.1, some of which are significant, particularly for MVS users.

- Deployments are now REXEC jobs therefore the RLIST and RDEL commands can be used.
- For MVS, the old behavior for deployment scripts was 'fire-and-forget'. The new default behavior is to wait for the script to connect back to Dimensions and mark the script as successful or unsuccessful. There are new symbols in dm.cfg to control this processing:
  - DM_WAIT_FOR_MVS_DEPLOYMENT_SCRIPT (Y/N): sets or clears waiting for MVS deployments.
  - DM_MAX_MVS_DEPLOY_SCRIPT_WAIT n: sets the maximum number of seconds that an MVS deployment job will be waited for. The default is 240 seconds.
- It is now possible to attach script parameters to a deployment area using `/SCRIPT_PARAMETERS`. This has been implemented, in the command line only, for the UA and CA commands. The LA command has changed to indicate that script parameters exist, but they cannot be listed at present. The LA command now also has a `/NODETAIL` switch to provide a short list of areas.
- The `RDEL /FORCE` option has been added to enable the removal of REXEC jobs even if the state is not a terminal one. This can be useful when cleaning up in some circumstances.
- The exact list of variables being passed has changed (see "Variables" on page 312). There are more variables and some of their meanings have changed. In the past, all variables were either singleton, or in an array with one element for every item being deployed. Now there are arrays of baselines, requests, and baseline product names which are not in parallel with the list of items being handled.
- The new `)DUMP` syntax provided by the template processor can be used to more easily see the variables that have been passed to the deployment script.
- There are new sample templates in the z/OS installation (`MDHBPRE0`, `MDHBPST0` and `MDHBERR0`) that provide sample pre-, post-, and fail scripts. These demonstrate the use of `)DUMP` for MVS systems and will connect back to Dimensions to mark the script as having succeeded.
- The commands that cause deployments to occur return SIR variables containing the related jobs started. Each variable has a name that incorporates the Area name. These can be used with `REXEC /WAIT` to cause a script of commands to stop until the related job has completed.
- Standard deployment script variables are available that contain both the specific area name (DMAREA) and the node (machine) that this area is on (DMAREANODE). This allows a deployment script to customize itself, depending on the context.

Variables

IMPORTANT! Please note that the server generated scripts, described below, are not automatically checked for syntax. Any syntax errors present in the scripts will cause the scripts to fail to work as expected.

A set of area script template variables is created by the server and is made available to give the context of the operation being performed. These variables are listed below, for details see the Deploy Item command in the *Command-Line Reference*. The single-valued variables represent properties common to the operation and all items being transferred. The array-valued variables represent properties specific to each item being transferred.

- Arrayed over items

  ```
  DMBRANCH
  DMCTIME
  DMDIR
  DMFILENAME
  DMFORMAT
  DMID
  DMISDIR
  DMPREFIX
  DMPRODUCT
  DMREVISION
  DMSUFFIX
  DMTRANSFERTYPE
  DMTYPE
  DMVARIANT
  DMWSID
  DMWSPRODUCT
  ```

- From change set (arrayed):

  ```
  DMBLNID
  DMBLNPRODUCT
  DMREQUEST
  ```

- From original command (arrayed):

  ```
  DMCMDREQUESTS
  ```

- From original command (single value):

  ```
  DMCOMMAND
  ```

- Single value:

  ```
  DMAREA
  DMAREANODE
  DMCERTIFICATE
  DMCMDCOMMENT
  DMDJQJOBID
  DMJOBID
  DMJOBTYPE
  DMSERVER
  DMSTAGE
  ```

Testing Variables

To test the variables create a dummy area template that expands all of the variables listed above, and execute it on items and projects. The example below is the trace output from such a run. To get a trace similar to this, use the debugging symbols described on page 314.

```
16TP1021T IN  DMCERTIFICATE.
16TP1034T OUT
2DC74AB5E8F698998A898CE7F049CA777D45BBD491C5D575A8C9E52E100DF39809511D50EC89E531CABCB8035
FE52F5C86A24678F4EE581073F42EF0AEFD2928B84BED26C12DD518C273778781988111E902E762B5A797343B
77E420FE76ED0A4F4D229F09583C6E9BE433BA788D2D44CE19405FC3ED275772D502168D4D75C5
16TP1021T IN  DMSERVER.
16TP1034T OUT STAL-DEV-SJM3
16TP1021T IN  DMAREA.
16TP1034T OUT SCRIPTTEST1
16TP1021T IN  DMDIR.
16TP1034T OUT c:\temp\scripttest1\t1\
16TP1021T IN  DMCOMMENT.
16TP1034T OUT Initial Revision
16TP1021T IN  DMFILENAME.
16TP1034T OUT tm.c
16TP1021T IN  DMTRANSFERTYPE.
16TP1034T OUT c
16TP1021T IN  DMREVISION.
16TP1034T OUT 1
16TP1021T IN  DMBRANCH.
16TP1034T OUT
16TP1021T IN  DMFORMAT.
16TP1034T OUT TEXT
16TP1021T IN  DMPREFIX.
16TP1034T OUT tm
16TP1021T IN  DMSUFFIX.
16TP1034T OUT c
16TP1021T IN  DMWSPRODUCT.
16TP1034T OUT PAYROLL
16TP1021T IN  DMWSID.
16TP1034T OUT DIRTEST2
16TP1021T IN  DMPRODUCT.
16TP1034T OUT PAYROLL
16TP1021T IN  DMID.
16TP1034T OUT TM C
16TP1021T IN  DMVARIANT.
16TP1034T OUT A
16TP1021T IN  DMTYPE.
16TP1034T OUT SRC
16TP1021T IN  DMCTIME.
16TP1034T OUT Wed Aug 09 10:20:18 2006
16TP1021T IN  DMCOMMAND.
16TP1034T OUT CI "PAYROLL:TM C.A-SRC;1" /USER_FILENAME="C:\tm.c" /FILENAME="tm-01.c" /
PART="PAYROLL:PAYROLL.A;1" /WS_FILENAME="t1\tm.c" /FORMAT="TEXT" /COMMENT="Initial
Revision"
16TP1021T IN
16TP1034T OUT
```

Differences in Relation to Build Templates

The build specific variables, such as DMTARGET, are described in "Dimensions Build Standard Symbols" on page 208 and are not available for area scripts. Build specific functions, such as `_MDHDSN_()`, are also not available.

Area scripts should be synchronous as the server waits for success or failure to organize the next operation. You can have MVS batch JCL templates, but the success flag only indicates the successful submission of the job, not its completion. In a production environment it is unlikely that you would want a batch job submitted for each item that is deployed.

Debugging

An error during the execution of an area script can prevent item operations from working in the projects associated with that area. Use the following variables to help you to get the template to execute successfully:

In the `dm.cfg` file on the Dimensions CM server specify the following variables:

```
DM_KEEP_DEPLOYMENT_SCRIPTS      YES
DM_VERBOSE_DEPLOYMENT_SCRIPTS   YES
```

These variables allow easier analysis of what happens when a server tries to execute a template. The `%DM_ROOT%/temp` folder on the remote node contains evidence of any attempted script execution, and template messages are passed back to the server in the normal flow of messages.

In the `dm.cfg` file on the remote node where the deployment area is located specify the following variable:

```
DM_TPLB_TTLOG                   YES
```

This variable creates more detailed trace messages from the templating routines that expand the template. These messages appear in the console window of desktop client when, for example, an item is deployed.

Other Features

Although performance is degraded significantly, it is possible for area scripts to use the `DMCERTIFICATE` variable to gain access back to the Dimensions CM server from which the request came. For details see page 302.

If you are going to use DMCERTIFICATE, test it carefully to avoid an endless cycle of recursive area scripts if the area script itself causes another script to be fired, for example, if the area script tries to perform a CI operation in the same project.

Specifying User Attributes in Deployment Area Scripts

Introduction

This behavior enables you to specify which user attributes are passed to the deployment area script for each item, request, or baseline. To reduce the volume of data this is done per item type, request type, or baseline type. To trigger this behavior you must set up one or more variables that specify the attribute names required for each type:

- **Items**: `DMITEMATTR_xxx` where `xxx` is an item type to be an array of attribute names. You can specify different collections of attributes for different item types.
- **Requests**: `DMREQUESTATTR_xxx` where `xxx` is the request type, for example, TDR in the INTERMEDIATE process model.
- **Baselines**: `DMBLNATTR_xxx` where xxx is the baseline type, for example, BASELINE in the INTERMEDIATE process model.

This specification must be available to deployment area script processing when deployment is being performed. The easiest way to attach a specification to an area definition is to use the `/SCRIPT_PARAMETERS` qualifier in a CA or UA command.

During deployment script preparation (under control of the deployment server) new tables containing UIDs are exposed. These tables are called DMITEMUID for items, DMREQUESTUID for requests and DMBLNUID for baselines. For each object, new symbols will be created called:

```
DMxxxATTR_uid_attrname
```

where:

- `xxx` is ITEM for item objects, REQUEST for request objects, and BLN for baselines.
- `uid` identifies the object for which these are the attributes.
- `attrname` is the name used for the attribute in the Dimensions database.

NOTE

- Processing does not properly handle array-valued attributes with empty values. For example, if you have a variable that occurs three times and has an empty value at the end, you will get an array that is two long not three, as the processing is unable to determine that there is an empty value at the end. Empty values in a multi-valued attribute will work provided there is a nonempty value in the last array element.
- If objects do not have some of the requested attributes then you will get a single-valued attribute with a value of an empty string. This is instead of requiring you to use)IFNDEF processing.

Handling Attribute Behavior in the Template Processor

Use the following techniques to handle attribute behavior in the template processor.

Scalar Attributes

For scalar attributes the following code copies the attribute value for a single item uid and attribute to a variable called COMPLEXITY:

```
)CALL substring COMPLEXITY DMITEMATTR_%DMITEMUID._COMPLEXITY 0
```

Array Valued Attributes

For array valued attributes use one of the following techniques.

- Hard coded subscript number:

  ```
  )CALL substring COMPLEXITY DMITEMATTR_%DMITEMUID._COMPLEXITY(2) 0
  ```

- To work through all subscripts:

  ```
  )SET I=0
  )REP DMITEMATTR_%DMITEMUID._COMPLEXITY
  )CALL substring COMPLEXITY DMITEMATTR_%DMITEMUID._COMPLEXITY(%I) 0
  … do something with complexity
  )ADJUST I 1
  )ENDR
  ```

  This has been made to work by extending the TPCK_GetString routine to handle subscripted variables. You can use this technique in any callout that uses this routine.

- To work through all subscripts using a more elegant technique:

  To access)REP variables you can reference to a symbol of the form `%n.` where n is numeric. This form chooses the nth variable, starting at zero, specified in the invocation of the REP. For example:

  ```
  )REP DMITEMUID
  )REP DMITEMATTR_%DMITEMUID._VALIDSTAGE
      DMITEMATTR_DMITEMUID._COMPLEXITY
  …
  For item %DMITEMUID.
  Validstage = %1.
  Complexity = %2.
  ...
  )ENDR
  )ENDR
  ```

  will deal with each value in the array DMITEMATTR_%DMITEMUID._VALIDSTAGE and DMITEMATTR_%DMITEMUID._COMPLEXITY in turn (assuming these two arrays have the same cardinality).

Example

The example below is a simple pre-transfer event script that records the actions being performed into a cumulative log file.

```
)SET TM=%DMYEAR.%DMMONTH.%DMDAY.%DMHOUR.%DMMINUTE.%DMSECOND.%DMMICROSEC.
)REP DMDIR DMFILENAME
echo "%TM. %DMDIR. %DMFILENAME." >>c:\temp\deployment.log
)ENDR
exit 0
```

NOTE the script uses a repeat block (shown in bold) to correctly deal with single item transfers as well as multiple item transfers.

Part 5

Web Services Reference

Part 5: Web Services Reference contains the following chapters:

Getting Started	321
Tips for Using Dimensions CM Web Services	331
Web Services Reference	345
Application Lifecycle Framework Events Reference	499

Chapter 15

Getting Started

Introduction	322
Dimensions CM Web Services Architecture	322
Before You Begin	328
Installing the Dimensions CM Web Services API	330
Setting the Idle Connection Timeout	330

Introduction

About the Dimensions CM Web Services API

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

Dimensions CM Web Services Architecture

Introduction

Dimensions CM provides WS-I Basic Profile (BP) compliant SOAP Web services for performing common Dimensions CM operations utilizing Apache Axis2 1.4.1 and Document/Literal Style WSDL definitions.

Terminology

Term	Description
WS-I	The Web Services Interoperability industry consortium
WS-I Basic Profile	A specification from WS-I that provides interpretability guidance for core Web services such as SOAP and WSDL.
SOAP	Simple Object Access Protocol.
Apache Axis2	A core engine for Web services. It is a complete re-design and re-write of the widely used Apache Axis SOAP stack.
WSDL	Web Services Description Language. An XML format that allows service interfaces to be described along with details of their bindings to specific protocols.
Document/Literal Style WSDL	A WSDL binding style describes how a Web service is bound to to a messaging protocol, particularly the SOAP messaging protocol. A WSDL SOAP binding can be either a Remote Procedure Call (RPC) style binding or a document style binding, and can also have an encoded use or a literal use. Dimensions CM utilizes the Document/Literal WSDL binding style.

The implementation of most Dimensions CM Web services is based on Dimensions CM commands; and, in all such cases, the Web service description mentions the commands used. For example: the `actionRequest` Web service uses the `AC` command; while the `getItems` Web service uses the `download` command. For detailed information about Dimensions CM commands, please see the *Command-Line Reference*.

As mentioned earlier, Dimensions CM Web services are implemented using Axis2 1.4.1. This means that Axis2 gets control before the Dimensions CM Web services

implementations and performs a validation of each SOAP request and its parameters. In those cases where the SOAP request is malformed, Axis2 throws its own exception and returns a SOAP fault without delegating control to the Dimensions CM Web services implementation.

NOTE This can be important if you prepare SOAP requests manually with a utility like soapUI, which is flexible in forming requests.

Examples of malformed SOAP requests include those where:

- Some mandatory parameters are missing.
- The content of the parameters does not match their type, for example:
  - string in field with type `int`.
  - non-valid value in the `enum` field type.
  - incorrect date format.

In some cases, such errors may not be very informative. For example, containing output like:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>unknown</faultstring>
      <detail/>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

The best approach in such cases is to check the Tomcat server logs (see "Dimensions CM WSDL" on page 323 for details).

Dimensions CM WSDL

The WSDL used for Dimensions CM Web services is named `dmwebservices.wsdl`. It contains links to the following XSDs, which are co-located in same folder as `dmwebservices.wsdl`:

- `basic.xsd`
- `projects.xsd`
- `parts.xsd`
- `requests.xsd`

The Dimensions CM Web services WSDL can be obtained in several ways.

The default location for Web services implemented using Axis2 is:

```
http://<server>:<port>/dmwebservices2/services/dmwebservices?wsdl
```

The WSDL and XSDs obtained this way go through, and are modified, by Axis2. They contain links like `"dmwebservices?xsd=basic.xsd"`, which can be problematic in some

circumstances. In such circumstances, the original WSDL and XSDs are provided, and can be obtained at the following URL:

```
http://<server>:<port>/dmwebservices2/WSDL/dmwebservices.wsdl
```

These files can be also obtained directly through the operating system file system at the following locations:

```
<Tomcat-Root>\webapps\dmwebservices2\WSDL\
```

Dimensions CM Web Services Logging Mechanism

All Web services contain a logging mechanism that logs:

- The Web service input parameters.
- The result of performing the Dimensions CM command and exceptions if any occur.

By default, the logging level of Dimensions CM Web services is set to `DEBUG`, it can be changed in the following file in an installed Dimensions CM Windows installation:

```
<Tomcat-Root>\webapps\dmwebservices2\
    WEB-INF\classes\log4j.properties
```

or from the following file in an installed Dimensions CM UNIX installation:

```
<Tomcat-Root/webapps/dmwebservices2/
    WEB-INF/classes/log4j.properties
```

The Tomcat log file for a Dimensions CM Windows installation is:

```
<Tomcat-Root>\logs\stdout_<YYYYMMDD>.log
```

and for a Dimensions CM UNIX installation it is:

```
<Tomcat-Root>/logs/stdout_<YYYYMMDD>.log
```

The following is an example of the logging of a Web service that executed successfully:

```
[DEBUG] entering createRequest
[DEBUG] Request details:
   TypeName=CR
   ProductId=QLARIUS
   Title=WS2 Test 12/9/2008 16:20
   DetailedDescription=description string
[DEBUG] Setup Attribute:
   [0]:Name=Severity
   [0]:Type=Single_Value
   [0]:DataType=Char
   [0]:Value=1
[DEBUG] Setup Attribute:
   [1]:Name=PLAN_FINISH
   [1]:Type=Single_Value
   [1]:DataType=Date
   [1]:Value=2008-11-07T15:31:25+02:00
 [DEBUG] Result of DM operation: "SUCCESS: The request QLARIUS_CR_66 has been forwarded
    to PETA

Operation completed"
[DEBUG] leaving createRequest
```

The following is an example of the logging of a Web service that failed to execute successfully:

```
[DEBUG] entering deployRequest
[DEBUG] Request details:
   requestId=QLARIUS_CR_48
   stage=DEVELOPMENT
   comment=comment string
   traverseChildren=false
[DEBUG] leaving deployRequest with exception: Error: Request is already at stage
    DEVELOPMENT
```

In some cases the log may not contain records concerning Web service invocation. If a SOAP request was actually issued, this usually means that it is highly probable that the request did not pass the Axis2 validation.

Dimensions CM Web Service Error Handling

Error Handling for a Web Service that Performs a Single Dimensions CM Command

A Web service that invokes a single Dimensions CM command will, on detecting an error condition, return a SOAP fault with a description of the error that occurred. The error message can be received from the Dimensions CM server, from underlying Dimensions CMAPIs, or from the Web service itself.

The following is an example of an error received from a Dimensions CM server:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>COR0005651E The mandatory attribute 'Test Result'
(TEST_RESULT) has not been assigned to QLARIUS_CR_31</faultstring>
      <detail/>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Error Handling for a Web Service that Performs a Batch of Dimensions CM Commands

A Web service that processes a batch of Dimensions CM commands has a parameter—called `continueOnError`—which determines if subsequent commands in the batch should continue to be executed after the detection of an error condition. This parameter is *not* mandatory and set to false by default:

- If the value of `continueOnError` is false

  If one of the commands fails, the Web service does not continue with the execution of the subsequent commands and returns an error.

- If the value of continueOnError is true

  If one of the commands fails, the Web service continues to try to execute the subsequent commands and returns errors for each of the failing commands.

The following is an example of such error handling from `relateRequestToRequests` (two of three operations failed):

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>One or more operations have failed! See additional details in
    response.</faultstring>
      <detail>
        <ns2:DimensionsBatchFault xmlns:ns2="http://serena.com/dmwebservices2">
          <ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>The request QLARIUS_CR_30 is already related as INFO to
    QLARIUS_CR_58</ns2:description>
            </ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Request was not found!</ns2:description>
            </ns2:result>
          </ns2:result>
        </ns2:DimensionsBatchFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Interpretation of Axis2 Exceptions

The following is an example of an Axis2 exception:

```
[ERROR]
org.apache.axis2.AxisFault
        at org.apache.axis2.AxisFault.makeFault(AxisFault.java:430)
        at com.serena.dmwebservices2.DmwebservicesMessageReceiverInOut.fromOM
    (DmwebservicesMessageReceiverInOut.java:2201)
   ...
        at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.IllegalArgumentException
        at com.serena.dimensions.basic._1.AttributeDatatype$Factory.fromString
    (AttributeDatatype.java:393)
   ...
        ... 23 more
```

The exception is due to `AttributeDatatype` not being correctly specified. This type is described in `basic.xsd` and is used for describing the "datatype" of `Attribute`. In `basic.xsd`, the `AttributeDatatype` described as type is restricted to specified values (enumeration;) so this error can be interpreted as "user specified a non-valid value for the Attribute datatype"

Before You Begin

Before you install and use the Dimensions CM Web services API, there are a few things to consider.

System Requirements

The Dimensions CM Web services API are supported on various Web application containers and operating-system platforms. Please see the Dimensions CM readme file for full details including supported release levels.

Licensing

Dimensions CM consumes a concurrent or named license when a user first invokes a Web service, the license remaining in use until a configurable "idle connection" timeout period expires (a default of five minutes). This allows a user to make many Web services calls while making use of the same license. See "Setting the Idle Connection Timeout" on page 330 for details.

Security

The Dimensions CM Web services support using secure `https` to connect from the client. Use of SSL (Secure Sockets Layer) is recommended for any customers connecting to their Web services server in a non-secure environment. Using SSL prevents credentials from being extracted from the messages that are sent to the server. Web application containers should be configured to allow or require SSL to connect to the Web services.

For information on setting up SSL, see the System Administration Guide.

Authentication

Dimensions CM supports the following authentication types:

NOTE User credentials are needed to run a Web service, and are passed as inputs to each Dimensions CM Web Service call.

- Local operating-system authentication.
- LDAP (Lightweight Directory Access Protocol).
- PAM (Pluggable Authentication Modules).
- SSO (Single Sign On).

See the System Administration Guide for details.

Setting Up a Development Environment

Generating Web Services Stubs

The Dimensions CM Web services are defined in a WSDL file. You can build an application that interacts with the Dimensions CM Web services in any Integrated Development Environment (IDE) that can generate code stubs from this WSDL file. You can create the stub files using the WSDL file available at:

```
http://<serverName>:<port>/dmwebservices2/WSDL/dmwebservices.wsdl
```

Setting Up Microsoft Visual Studio To Use Dimensions CM Web Services

To create Microsoft Visual Studio applications that interact with the Dimensions CM Web services, you must (as explained below):

1. Upgrade Visual Studio.
2. Set up Web service stub files.

- **Upgrading Visual Studio**

  To use WS-Security, you must install the Microsoft Web Service Enhancements (WSE) product and then enable the WSE for your project.

  For information on WS-Security, see "Authentication Methods" on page 332.

  To upgrade Visual Studio .NET 2003 or Visual Studio 2005 with WSE:

  a Download the appropriate version of WSE from the following location:

  http://msdn.microsoft.com/webservices/building/wse/default.aspx

  b After installing the WSE, open your solution in Visual Studio.

  c Right-click your project and select WSE Settings.

  d Select the **Enable this project for Web Services Enhancements** check box.

  e Click **OK**.

  You can now start building applications using the Dimensions CM Web services with WSSecurity.

- **Setting Up Web Service Stub Files**

  To access the Web services from Visual Studio, you must add references to the WSDL file. When you do this, Visual Studio creates stub classes that your application can use to access the Web service methods. You can create these stub classes for Visual Basic or C#.

  Once you have created the stub files, you can update them to use the client protocol provided by the WSE.

  To set up the Web service stub files:

  a In Visual Studio, create or open a project.

  b Select Project | Add Web Reference.

  c In the dialog box that appears, enter the URL for the Dimensions CM Web services in the **URL** field.

  d Enter a name in the **Web reference name** field. This name will be used in your code to refer to the web reference.

  e Click **Add Reference**.

Installing the Dimensions CM Web Services API

To have access to the fullest range of currently available Dimensions CM Web services, Dimensions CM needs to be installed—see the *Installation Guide for Windows* or *Installation Guide for UNIX* if you have not already installed this version.In all cases, the Web services API is silently installed as part of the installation—no user input is required.

NOTE Web services are delivered in a standard WAR (Web Application aRchive) and are deployed to your web application container during installation.

Setting the Idle Connection Timeout

What is the Idle Connection Timeout?

The idle connection timeout value determines how long a connection can be idle before a license is released by the server. By default, this timeout is five minutes. You may want to adjust this value to meet your working environment.

Changing the Idle Connection Time-Out

To change the default time-out:

1. Navigate to

   `<Tomcat-Root>\webapps\dmwebservices2\WEB-INF\classes`

   For UNIX versions of Dimensions CM, use forward slashes instead.

2. Open the `connectioncache.properties` file with a text editor.

3. Locate the `killIdleConnectionAfter` line and modify the value. The value is in minutes. If no value is set, the default idle connection time out will be five minutes.

4. Save and close the file.

5. Stop and restart any configured Web Server (by stopping and restarting Tomcat). See the System Administration Guide for instructions on how to stop and restart Tomcat.

Chapter 16

Tips for Using Dimensions CM Web Services

Authentication Methods 332
Supported Character Encoding 332
Supported Datetime Formats 332

Authentication Methods

The Dimensions CM Web services API only supports:

- Authentication by "Argument"

  Authentication occurs each time a method is called. The `Auth` argument passes the Dimensions CM user ID and password in plain text. You can also use this argument to specify the host name for licensing purposes, instead of using the client's IP address.

- Authentication by Single Sign On

  This authentication mechanism is described in the System Administration Guide. For this type of authentication, arguments cannot be specified—neither fully nor partiality. Only `DB_CONN` and `DB_NAME` can be specified.

- WS-Security

  WS-Security (Web Services Security) authentication creates a security token in the SOAP header. The credentials are passed as a Username token and a Base64-encoded password.

  To define WS-Security in Microsoft Visual Studio, add a Username token to the SOAP header that contains the plain text user ID and the Base64-encoded password. You must also have Microsoft Web Service Enhancements (WSE) installed.

Use of SSL is recommended with authentication by argument.

Supported Character Encoding

The Dimensions CM Web services use UTF-8 encoding. Single-byte characters are automatically supported with UTF-8. To enable support for multi-byte characters, you must set the expected encoding to UTF-8 on the client side. In Visual Studio 2005 C#, you can do this by overriding the `GetReaderFromMessage` method and setting the reader's encoding to UTF-8.

Supported Datetime Formats

General Datetime Format

In general, the Dimensions CM Web services dateTime field values are in the SOAP/XML datetime format. This format is described in:

http://www.w3.org/TR/xmlschema-2/#dateTime

See 3.2.7.1 Lexical representation for details. For example, a Dimensions CM Web service can return a datetime such as:

```
2005-11-07T15:31:25+02:00
```

Dimensions CM Attributes Datetime Format

For those Web services that have attributes, the attribute can have single-value or multi-value fields. These values are stored in a text (`xsd:string`) field type or in array of text fields, which can contain in/out values of Char/Number/Date data type.

Single-value or multi-value attributes of `Date` data type are returned in a text (`xsd:dateTime`) field in soap/XML datetime format. Such attributes are used, for example, in the following Web services:

- checkInItem
- checkOutItem
- createDesignPart
- createDesignPartBaseline
- createProject
- createProjectBaseline
- createRequest
- createRevisedBaseline
- getRequests.
- updateRequest

Chapter 17

Application Lifecycle Framework Events

Introduction to ALF Events	336
Dimensions CM ALF Events Architecture	336
ALF Events Available	340
Logging of ALF Events	343
Supported Datetime Formats	343
Error Message Logs	343

Introduction to ALF Events

Serena® Dimensions® CM can generate Application Lifecycle Framework (ALF) events for a restricted set of operations. ALF is a former Eclipse project, in which Serena played a leading role, to develop an open integration and interoperability framework for application lifecycle management tools.

Each Dimensions CM operation fires a single ALF event and in accordance with ALF guidelines these events are generated post operation. This is different from the Dimensions CM events callout interface (see the *Developer's Reference*), which may generate a number of specific events during the course of a single operation.

The data provided by each event is a reflection of the Dimensions CM command that has issued the ALF event. Therefore, the data provided by the various events may differ; not all events will provide the same amount of data to the event clients.

NOTE When you add solutions to source control from Visual Studio using Dimensions CM for Visual Studio, it may appear that Dimensions CM breaks the model elaborated above by invoking three deliver operations. In fact Visual Studio itself triggers 3 deliver events when you add solutions to source control, in turn triggering multiple ALF deliver events.

Dimensions CM ALF Events Architecture

Introduction

ALF events provide a common mechanism for notification of events that occur in an application.

The "endpoint" of the ALF Events actually represents a Web service that can perform some action itself or trigger some related Services Flow (Orchestration).

For a general introduction to ALF architecture, see:

http://wiki.eclipse.org/ALF/architecture/ALF_Architecture

Dimensions CM ALF Events Declaration

CAUTION! Dimensions CM 12.2.2 ALF events are not compatible with the earlier Dimensions CM 10.1.3.x ALF events. This is because of new rules introduced in the 2009 R1.02 release of Serena Business Manager (formerly called Serena Business Mashups).

The declaration of the Dimensions CM ALF events structure and options are declared in a WSDL as a typical Web service. The declarations are located in the following directory:

- Windows

  ```
  <DM_ROOT>\Integrations\Mashups\
  ```

- UNIX

  ```
  <DM_ROOT>/Integrations/Mashups\/
  ```

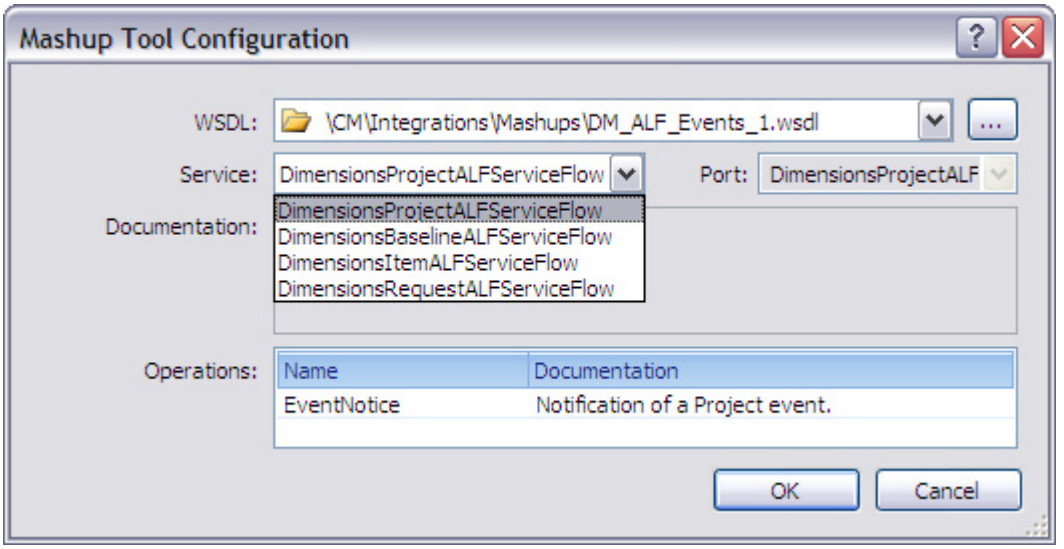
The above directory contains the following files:

- `ALFEventBase_1.xsd`
- `basic.xsd`
- `DM_ALF_Events_1.wsdl`

This WSDL defines the set of supported objects and events and the specific data provided by these events. It provides a description of following services:

Service	Description
DimensionsProjectALFServiceFlow	Events generated by Project object.
DimensionsBaselineALFServiceFlow	Events generated by Baseline object.
DimensionsItemALFServiceFlow	Events generated by Item object.
DimensionsRequestALFServiceFlow	Events generated by Request object.

The following screen illustrates how to select the appropriate service. If you need to process events from a different Dimensions CM object, you should select the same WSDL for that object, but select a different service.

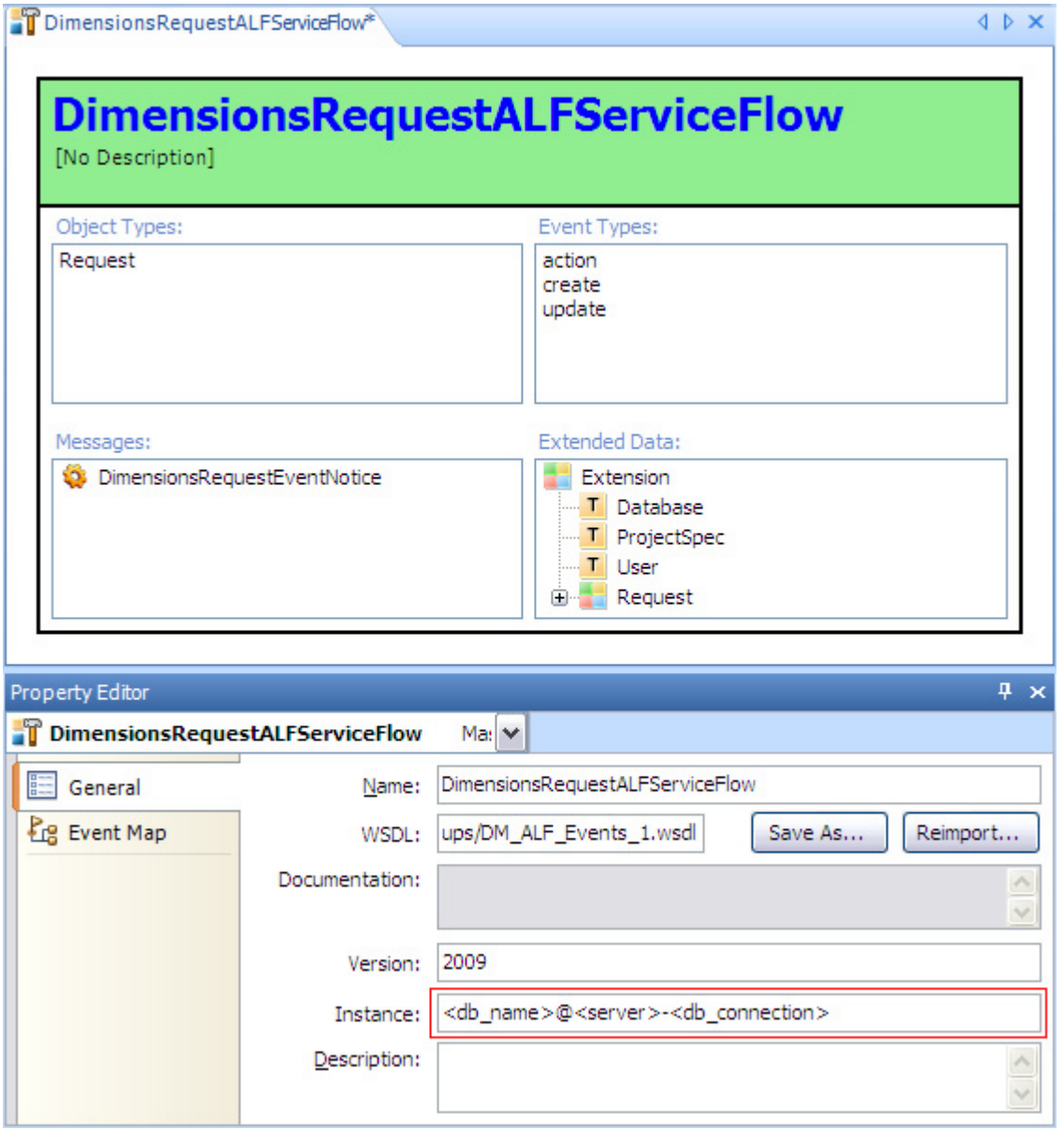


IMPORTANT! After the Process Application Tool is exported into SBM Composer, the **Instance** field (<db_name>@<server>-<db_connection>) must be correctly specified in upper case, for example, CM_TYPICAL@MY_HOST_NAM. Also, if the process application contains a Process Application Tool from Dimensions CM 10.1.3.x that has been upgraded to 12.2.2, you should ensure that the **Version** field contains 12.

If you want to replace the dynamic instance field above with a static string, update the Dimensions CM server dm.cfg to include a DM_ALF_PRODUCT_INSTANCE entry such as follows;

```
DM_ALF_PRODUCT_INSTANCE ALFDimensionsCMInstance
```

and restart the Dimensions CM listener. The string must not include XML escape characters.



Configuring ALF Events

Dimensions CMis not configured to generate ALF events by default. To configure the generation of Dimensions CM ALF events, you need to:

- Prepare an ALF events configuration file.
- Specify the location of that ALF events configuration file and an endpoint for the ALF Event Manager by adding the following to the Dimensions CM `dm.cfg` configuration file located in the Dimensions CM "root" directory:

```
DM_ALF_ENDPOINT       http://<server>:<port>/eventmanager/services/ALFEventManager
DM_ALF_EVENT_CONFIG   %DM_DFS%alf_events_config.xml
```

where `%DM_DFS%` is the `dfs` sub-directory of the Dimensions CM "root" directory, for example, `C:\Program Files\Serena\Dimensions 12.1\CM\dfs\`

The file `alf_events_config.xml` located in `%DM_DFS%` or $DM_ROOT/dfs is an example of an events configuration file. This file is liberally commented to explain its structure, but basically has the following structure:

```
<ALFEventsConfig>
  <Databases>
    <Database>
      <Name>MY_DB_ID@MY_SERVER-MY_DB_CONNECTION</Name>
      <Projects>
        <Project>
          <Name>MY_PRODUCT:MY_PROJECT</Name>
          <Objects>
            <Object>
              <Type>DMObjectType</Type>
              <Events>
                <Event>DMEventType</Event>
              </Events>
            </Object>
          </Objects>
        </Project>
      </Projects>
    </Database>
  </Databases>
</ALFEventsConfig>
```

Most sections can contain sub-elements, for example:

- Databases can contain many Database.
- Projects can contain many Projects.
- Events can contain many Event sections.

The database name and project/stream name can also be specified using wildcards, for example:

```
"QLARIUS_CM@*-DIM10","QLARIUS:UW_JAVA_*"
```

Currently, the following object types are supported:

- Baseline.
- Item
- Project
- Request
- ScheduledJob

IMPORTANT! The ALF events configuration is loaded on start up of the Dimensions CM server. If you change some parameters, you will need to restart the Dimensions CM server.

Generation of ALF events

Most Dimensions CM ALF events are generated by the Dimensions CM server `dmappsrv.exe` process. Build ALF events, however, are generated by Dimensions CM Build.

The generation of ALF events is based on the Dimensions CM events callout interface (see the *Developer's Reference*).

IMPORTANT! Deploy ALF events are not generated on creating or delivering files, because in those particular cases "Change Set" is deployed and ALF events only support Dimensions CM objects of type item, request, and baseline.

ALF Events Available

The ALF events are added to with each release of Dimensions CM. As of Dimensions CM 12.2.2 the following tables briefly describe the ALF events that are available:

NOTE All ALF events released prior to Dimensions CM 12.2.2 have been updated to be aware of the of the new or modifiedDimensions CM 12.2.2 functionality:

- Promote
- Demote
- Deploy and rollback

In most cases existing orchestrations using earlier Web services will function correctly, but old orchestrations that use deploy ALF events should still be carefully tested with the new deploy ALF events before continued use.

Baseline ALF Events

Event	Description
Build Baseline (build-submitted)	Fired by the build baseline (BLDB) operation.
Build Baseline (build-completed)	Fired when the build baseline is completed (on fail or success).
Build Baseline (build-allocated)	Fired when the build baseline is allocated.
Deploy Baseline (deploy)	Fired by the deploy baseline (DPB) operation.
Rollback Baseline (rollback)	Fired by the rollback baseline operation.

Item ALF Events

Event	Description
Check In Item (check-in)	Fired by the check in item (RI) operation.
Check Out Item (check-out)	Fired by the check out item (EI) operation.
Create Item (create)	Fired by the create item out (CI) operation.
Deploy Items (deploy)	Fired by the deploy items operation.
Deploy Items to Area	Fired by the deploy items to area operation.
Rollback Item (rollback)	Fired by the rollback item operation.
Undo Check Out (undo-check -out)	Fired by the undo check out (CIU) operation.
Promote	Promote item.
Demote	Demote item.

Project ALF Events

Event	Description
Build Project (build-submitted)	Fired by the build project (BLD) operation. **NOTE** These events are always fired if the `DM_ALF_ENDPOINT` parameter is specified in `dm.cfg`.
Build Project (build-completed)	Fired when the build project is completed (on fail or success). **NOTE** These events are always fired if the `DM_ALF_ENDPOINT` parameter is specified in `dm.cfg`.
Build Project (build-allocated)	Fired when the build is allocated.
Deliver (deliver)	Deliver to a project or stream.
Create Project (create)	Fired when a project or stream is created.

Request ALF Events

Event	Description
Action Request (action)	Fired by the action request (AC) operation.
Create Request (create)	Fired by the create request (CC) operation.
Deploy Request (deploy)	Fired by the deploy request (DPR) operation.
Deploy Request To Area	Fired by the deploy request to area operation.
RollbackRequest	Fired by the rollback request operation.
Update Request	Fired by the update request (UC) operation.

Miscellaneous ALF Events

Logging of ALF Events

Trace logging of the ALF events fired by Dimensions CM is provided as part of the SDP tracing. This may be enabled by adding the following to the Dimensions CM dm.cfg configuration file located in the Dimensions CM "root" directory:

- To enable SDP tracing:

```
DM_SDP_TRACE    <full directory path>
```

- To enable ALF event specific tracing:

```
DM_DEBUG_ALF_EVENTS    1
```

Supported Datetime Formats

The Dimensions CM ALF event datetime format is in the SOAP/XML datetime format. This format is described in:

http://www.w3.org/TR/xmlschema-2/#dateTime

See 3.2.7.1 Lexical representation for details. For example, a Dimensions CM Web service can return a datetime such as:

```
2005-11-07T15:31:25+02:00
```

Error Message Logs

Consult your ALF Event Manager documentation for details of the location of error message log files.

Chapter 18

Web Services Reference

actionRequest	347
buildBaseline	349
buildProject	352
checkInItem	356
checkOutItem	360
createDeploymentArea	364
createDesignPart	367
createDesignPartBaseline	369
createProject	373
createProjectBaseline	377
createRequest	380
createRevisedBaseline	383
createScheduleJob	386
createStream	389
createWorkArea	392
delegateRequest	394
delegateRequestForReplication	396
demoteBaselines	398
demoteItems	401
demoteRequests	404
deployBaseline	407
deployRequest	409
getItems	411
getRequests	417
getVersion	422
listDeploymentAreas	424
listProjectItems	426
listProjectRequests	429
listProjects	431
listScheduleJobs	434
listWorkAreas	439
moveItemToPart	441
promoteBaselines	443
promoteItems	447

promoteRequests	450
relateItemsToParts	453
relateRequestToRequests	456
removeDeploymentArea	459
removeWorkArea	461
runCommand	463
submitDeployBaselines	465
submitDeployItems	468
submitDeployRequests	471
submitRollbackBaselines	474
submitRollbackItems	477
submitRollbackRequests	480
unrelateItemsFromParts	483
unrelateRequestFromRequests	486
updateDeploymentArea	489
updateRequest	492
updateWorkArea	497

actionRequest

Description

The `actionRequest` Web service provides Dimensions CM actions with a Dimensions CM request using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	This is the identity of the request to be actioned or checked.
status	xsd:string	N	Specifies the new status to be given to the request.
actionCheck	xsd:boolean	Y	Indicates that Dimensions CM is to check whether the specified request can be either actioned to the state specified by the `/STATUS` qualifier or the next normal state if `/STATUS` is not specified, while conforming to the current rules. **NOTE** This qualifier must not be used with the `/CLOSURE_CHECK` qualifier.
closureCheck	xsd:boolean	Y	Indicates that Dimensions CM is to check whether the specified request can be actioned to the final state in its normal lifecycle, while conforming to the current rules. **NOTE** This qualifier must not be used with the `/STATUS` qualifier or the `/ACTION_CHECK` qualifier.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`actionRequest` provides a method to action or check the status of a Dimensions CM request.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:

- on the client (for example, because of an incorrectly defined service endpoint), or
- on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:actionRequest>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_CR_47</dmw:requestId>
      <dmw:status>TEST</dmw:status>
      <dmw:actionCheck>false</dmw:actionCheck>
      <dmw:closureCheck>false</dmw:closureCheck>
    </dmw:actionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:actionRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:actionRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

buildBaseline

Description

The `buildBaseline` Web service builds a Dimensions CM Baseline using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselineSpec	xsd:string	N	Specifies the baseline to be built.
areaName	xsd:string	N	Specifies the Dimensions CM area to be used for the build. If this is not specified, all areas associated with the build configuration or configurations are used.
batchMode	xsd:boolean	Y	Specifies that the build is to be run in batch mode (the command does not wait for the build to finish).
buildClean	xsd:boolean	Y	Specifies that the area is to be cleaned of targets before the build process begins.
buildConfiguration Name	xsd:string	N	Specifies the build configuration.L
buildOptions	basic:BuildOptionArray	Y	Specifies any build options.
capture	xsd:boolean	Y	Specifies whether built targets are to be collected. **NOTE** It is not possible to collect default targets when building at the `DEVELOPMENT` stage.
requestIds	basic:RequestIdArray	N	Identifies a request to which the new items created from the built final targets are to be related `In Response To` if required by change management rules.
targetNames	basic:BuildTargetNameArray	Y	Specifies the list of targets to be built. **NOTE** If this is omitted, all targets for the project/stream are built.
touch	xsd:boolean	Y	Apply system date/time to downloaded files.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResultArray	N	Array of results of each single operation.

Usage

`buildBaseline` provides a method to build a Dimensions CM baseline. The result for this Web service is an array, each element of which contains detailed information on each single operation of the Web service:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise, false.
objectSpec	xsd:string	Y	Contains build job Id.
url	xsd:string	Y	Path to file that contains results of build operation.
description	xsd:string	Y	Not used.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/dimensions/
    basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:buildBaseline>
      <dmw:connectionDetails>
   <ns:username>dmsys</ns:username>
   <ns:password>dmsys_test</ns:password>
   <ns:dbName>cm_typical</ns:dbName>
   <ns:dbConnection>dim10</ns:dbConnection>
   <ns:server>localhost</ns:server>
    </dmw:connectionDetails>
      <dmw:baselineSpec>QLARIUS:PROD_UW-DOTNET_1.0</dmw:baselineSpec>
      <dmw:areaName>DOTNET_BUILD_1.0</dmw:areaName>
      <dmw:batchMode>false</dmw:batchMode>
      <dmw:buildClean>true</dmw:buildClean>
      <dmw:buildConfigurationName>*UW_DOTNET_1.0;4</dmw:buildConfigurationName>
     <dmw:buildOptions>
        <ns:buildOption>
          <ns:option>Opt1</ns:option>
          <ns:value>1</ns:value>
        </ns:buildOption>
      <dmw:capture>false</dmw:capture>
```

```
      <dmw:requestIds>
        <ns:requestId></ns:requestId>
      </dmw:requestIds>
      <dmw:targetNames>
        <ns:target></ns:target>
      </dmw:targetNames>
      <dmw:touch>false</dmw:touch>
    </dmw:buildBaseline>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:buildBaselineResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>
          <ns2:result>true</ns2:result>
          <ns2:objectSpec>3</ns2:objectSpec>
          <ns2:url>http://acme:8080/bws/Monitor_job_info_modal.do?
          current=true&job_id=3</ns2:url>
        </ns2:result>
      </ns2:result>
    </ns2:buildBaselineResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

buildProject

Description

The `buildProject` Web service builds a Dimensions CM project using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
projectSpec	xsd:string	N	Specifies a project to be built.
areaName	xsd:string	Y	Specifies the Dimensions CM area to be used for the build. **NOTE** If this is not specified, all areas associated with the build configuration or configurations are used.
areaType	xsd:string	Y	Build area type (deployment or work). Specifies whether built targets are to be collected. **NOTE** If you set the '`populate`' argument, below, to '`true`' (SOAP request <dmw:populate>true </dmw:populate>) and you wish to have all items downloaded to the workarea for the build, then you must also set the '`areaType`' argument to '`work`' (SOAP request <dmw:areaType>work </dmw:areaType>).This is because the `BLD /POPULATE` parameter is applicable to work areas only, see the *Command-Line Reference*.
stage	xsd:string	Y	Applicable only to deployment areas. If the area type is `DEPLOYMENT`, this qualifier specifies the stage for which the targets are to be built.
audit	xsd:boolean	Y	Specifies that an audit is to be run before the build.
wait	xsd:boolean	Y	Specifies that the build is to be run in batch mode (the command does not wait for the build to finish).
buildConfiguration Name	xsd:string	N	Specifies the build configuration.
buildOptions	basic:BuildOptionArray	Y	Specifies any build options.

Argument	Type	Optional?	Description
targetNames	basic:BuildTargetNameArray	Y	Specifies the list of targets to be built. **NOTE** If this is omitted, all targets for the project are built.
requestIds	basic:RequestIdArray	Y	Identifies a request to which the new items created from the built final targets are to be related `In Response To` if required by change management rules.
capture	xsd:boolean	Y	Specifies whether built targets are to be collected. **NOTE** It is not possible to collect default targets when building at the `DEVELOPMENT` stage.
populate	xsd:boolean	Y	Download files to work area before each build. **NOTE** See the note accompanying the '`areaType`' argument.
touch	xsd:boolean	Y	Apply system date/time to downloaded files.
traverseRequests	xsd:boolean	Y	Traverse child source requests.
buildClean	xsd:boolean	Y	Specifies that the area is to be cleaned of targets before the build process begins.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResultArray	N	Array of results of each single operation.

Usage

`buildProject` provides a method to build a Dimensions CM project/stream. The result for this Web service is an array, each element of which contains detailed information on each single operation of the Web service:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successfully, otherwise false.
objectSpec	xsd:string	Y	Contains build job Id.
url	xsd:string	Y	Path to the file that contains the results of the build operation.
description	xsd:string	Y	Not used.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:

- on the client (for example, because of an incorrectly defined service endpoint), or
- on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:buildProject>
     <dmw:connectionDetails>
   <ns:username>dmsys</ns:username>
   <ns:password>dmsys_test</ns:password>
   <ns:dbName>cm_typical</ns:dbName>
   <ns:dbConnection>dim10</ns:dbConnection>
   <ns:server>localhost</ns:server>
     </dmw:connectionDetails>
      <dmw:projectSpec>QLARIUS:UW_DOTNET_1.0</dmw:projectSpec>
      <dmw:areaName>DOTNET_BUILD_1.0</dmw:areaName>
      <dmw:areaType></dmw:areaType>
      <dmw:audit>false</dmw:audit>
      <dmw:wait>false</dmw:wait>
      <dmw:buildConfigurationName>*UW_DOTNET_1.0;4</dmw:buildConfigurationName>
      <dmw:buildOptions>
        <ns:buildOption>
          <ns:option>Opt1</ns:option>
          <ns:value>1</ns:value>
        </ns:buildOption>
      </dmw:buildOptions>
      <dmw:targetNames>
      <ns:target></ns:target>
      </dmw:targetNames>
      <dmw:requestIds>
        <ns:requestId></ns:requestId>
      </dmw:requestIds>
      <dmw:capture>false</dmw:capture>
      <dmw:populate>false</dmw:populate>
      <dmw:touch>false</dmw:touch>
      <dmw:traverseRequests>false</dmw:traverseRequests>
      <dmw:buildClean>false</dmw:buildClean>
    </dmw:buildProject>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:buildProjectResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>
          <ns2:result>true</ns2:result>
          <ns2:objectSpec>7</ns2:objectSpec>
          <ns2:url>http://acme:8080/bws/Monitor_job_info_modal.do?
current=true&job_id=7</ns2:url>
        </ns2:result>
      </ns2:result>
    </ns2:buildProjectResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

checkInItem

Description

The `checkInItem` Web service provides a method to check in a Dimensions CM item using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
itemSpec	xsd:string	N	Specifies the item to be checked in.
projectSpec	xsd:string	N	A checked-out item can only be checked in to the specific project from which it was originally checked out (an error will be generated if you try to check it in to another project/stream). Therefore, if your current project/stream is not that project/stream, either this qualifier must be used to specify the correct check in `<project-spec>` or your current project/stream must be set to that `<project-spec>` using the Set Current Project (`SCWS`) command.
fileName	xsd:string	Y	Specifies the name of the project file name. If `/ROOT_PROJECT` is used to specify the root project, `/FILENAME` is interpreted in the scope of that project/stream. The project file name identifies the relative path (directory plus file name) from the working location of the file to be used when the item is checked in from the current project/stream.
sourceFileName	xsd:string	Y	Specifies the file in the "work-area" from which the item will be copied.
status	xsd:string	Y	Specifies a valid changed status (lifecycle state) for the checked in revision. **NOTE** The only equivalent to this parameter in GUI mode is Check In Item (RI) followed by Action Item (AI). The status, if specified, must be one which would be valid if AI had been used separately.
comment	xsd:string	Y	Comment text to explain the reason for the check in of this item revision. The comment text can be up to 1978 characters long, and can be made available within the item header.

Argument	Type	Optional?	Description
attributes	basic:AttributeArray	Y	Variable name defined for one of the user-defined attributes for items, which has also been declared usable for the `<product-id>` and `<item-type>` specified in `<item-spec>`. `<valueN>` is the substitution value to be given to this attribute.
forceUpdate	xsd:boolean	Y	If the checksum is enabled for the item type and the file checked in has not been modified, the check in will succeed only if this qualifier is used, otherwise it will fail.
checkInUnchanged	xsd:boolean	Y	Performs a CIU (Cancel Item Update) command if the user file does not differ from the base revision and Dimensions CM is configured to allow updates only if a real change is made.
codepage	xsd:string	Y	Specify one of the code page values listed in the text file `codepage.txt`, located on your Dimensions CM server in the codepage subdirectory of the Dimensions CM installation directory. This file also provides more information about translation between code pages.
contentEncoding	xsd:string	Y	This parameter disables creation and usage of metadata files in the local work area.
keep	xsd:boolean	Y	Specifies that the user area file, which is normally deleted after its data has been placed under Dimensions CM control, is to be left intact. If the command is successful and `/KEEP` is specified, local metadata is updated; the new revision number is recorded and marked as no longer checked out. `/NOKEEP` causes the local metadata to be deleted as well as the file.
fileContents	xsd:base64Binary	Y	Encoded file contents. (This parameter works only if `sourceFileName` is not specified)

Response

Argument	Type	Optional?	Description
itemSpec	xsd:string	N	Returns the item Id that was checked in.
itemURL	xsd:string	N	Returns the url of the checked-in item.

Usage

`checkInItem` provides a method to check in a Dimensions CM item.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:checkInItem>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:itemSpec>QLARIUS:4K237 ASSEMBLYINFO VB.A-SRC;13.0</dmw:itemSpec>
      <dmw:projectSpec>QLARIUS:QUOTE_WEB_1.0</dmw:projectSpec>
<dmw:fileContents>SW1wb3J0cyBTeXN0ZW0NCkltcG9ydHMgU3lzdGVtLlJlZmxlY3Rpb24NCkltcG9ydHMg
    U3lzd
GVtLlJ1bnRpbWUuSW50ZXJvcFNlcnZpY2VzDQoNCicgR2VuZXJhbCBJbmZvcm1hdGlvbiBhYm91dCBhbiBhc3N
    lbWJs
eSBpcyBjb250cm9sbGVkIHRocm91Z2ggdGhlIGZvbGxvd2luZyANCicgc2V0IG9mIGF0dHJpYnV0ZXMuIENoYW
    5nZSB
0aGVzZSBhdHRyaWJ1dGUgdmFsdWVzIHRvIG1vZGlmeSB0aGUgaW5mb3JtYXRpb24NCicgYXNzb2NpYXRlZCB3a
    XRoIG
FuIGFzc2VtYmx5Lg0KDQonIG1vcmUgY29tbWVudHMgaGVyZQ0KDQonIFJldmlldyB0aGUgdmFsdWVzIG9mIHRo
    ZSBhc
3NlbWJseSBhdHRyaWJ1dGVzDQoNCjxBc3NlbWJseTogQXNzZW1ibHlUaXRsZSgiIik+IA0KPEFzc2VtYmx5OiB
    Bc3Nl
bWJseURlc2NyaXB0aW9uKCIiKT4gDQo8QXNzZW1ibHk6IEFzc2VtYmx5Q29tcGFueSgiIik+IA0KPEFzc2VtYm
    x5OiB
Bc3NlbWJseVByb2R1Y3QoIiIpPiANCjxBc3NlbWJseTogQXNzZW1ibHlDb3B5cmlnaHQoIiIpPiANCjxBc3Nlb
    WJseT
ogQXNzZW1ibHlUcmFkZW1hcmsoIiIpPiANCjxBc3NlbWJseTogQ0xTQ29tcGxpYW50KFRydWUpPiANCg0KJ1Ro
    ZSBmb
2xsb3dpbmcgR1VJRCBpcyBmb3IgdGhlIElEIG9mIHRoZSB0eXBlbGliIGlmIHRoaXMgcHJvamVjdCBpcyBleHB
    vc2Vk
IHRvIENPTQ0KPEFzc2VtYmx5OiBHdWlkKCJCQkRCQTY3RS1CRUZDLTQyQTEtOUVFMS1FQTk0MTk0NzU4Q0YiKT
    4gDQo
NCicgVmVyc2lvbiBpbmZvcm1hdGlvbiBmb3IgYW4gYXNzZW1ibHkgY29uc2lzdHMgb2YgdGhlIGZvbGxvd2luZ
    yBmb3
```

```
VyIHZhbHVlczoNCicNCicgICAgICBNYWpvciBWZXJzaW9uDQonICAgICAgTWlub3IgVmVyc2lvbiANCicgICAg
    ICBCd
WlsZCBOdW1iZXINCicgICAgICBSZXZpc2lvbg0KJw0KJyBZb3UgY2FuIHNwZWNpZnkgYWxsIHRoZSB2YWx1ZXM
    gb3Ig
eW91IGNhbiBkZWZhdWx0IHRoZSBCdWlsZCBhbmQgUmV2aXNpb24gTnVtYmVycyANCicgYnkgdXNpbmcgdGhlIC
    cqJyB
hcyBzaG93biBiZWxvdzoNCg0KPEFzc2VtYmx5OiBBc3NlbWJseVZlcnNpb24oIjEuMC4qIik+IA0K
</dmw:fileContents>
    </dmw:checkInItem>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:checkInItemResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:itemSpec>QLARIUS:4K237 ASSEMBLYINFO VB.A-SRC;13.0</ns2:itemSpec>
     <ns2:itemURL>http://acme:8080/dimensions?jsp=api&command=openi
&object_id=QLARIUS%3A4K237+ASSEMBLYINFO+VB.A-
    SRC%3B13.0&DB_CONN=DIM10&DB_NAME=CM_TYPICAL</ns2:itemURL>
    </ns2:checkInItemResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

checkOutItem

Description

The `checkOutItem` Web service provides a method to check out a Dimensions CM item using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
itemSpec	xsd:string	N	Specifies the item to be checked out.
projectSpec	xsd:string	N	If specified, the new revision of the item will be placed in the project/stream, otherwise it will be placed in the user's current project/stream.
baselineSpec	xsd:string	Y	Specifies a release-baseline which contains the particular revision of `<item-spec>` to be selected. (As it is in a baseline, a new revision must of course be checked out as a copy of it.)
fileName	xsd:string	Y	Specifies the name of the project file name. If `/ROOT_PROJECT` is used to specify the root project/stream, `/FILENAME` is interpreted in the scope of that project/stream. The project file name identifies the relative path (directory plus file name) from the working location of the file to be used when the item is checked out from the current project/stream.
targetFileName	xsd:string	Y	Specifies the name of the file which will be created in the "work-area", and into which the item will be copied.
revision	xsd:string	Y	Specifies a new revision for the item. This new revision will be placed in the project specified by `/WORKSET`. If `/WORKSET` is omitted, then the new revision will be placed in the user's default project/stream. If omitted, Dimensions CM increments the current revision (the rightmost subfield only), unless the item revision in `<item-spec>` is at its initial lifecycle state. In this case, the revision is unchanged.

Argument	Type	Optional?	Description
attributes	basic:AttributeArray	Y	It's the Variable Name defined for one of the user-defined attributes for items, which has also been declared usable for the <product-id> and <item-type> specified in <item-spec>.<valueN> is the substitution value to be given to this attribute.
requestIds	basic:RequestIdArray	Y	Identifies a request to which the item revision is to be related In Response To.
codepage	xsd:string	Y	Specify one of the code page values listed in the text file codepage.txt, located on your Dimensions server in the codepage subdirectory of the Dimensions CM installation directory. This file also provides more information about translation between code pages.
touch	xsd:boolean	Y	Sets the modification time of the user file to the current system time. **NOTE** This parameter works only if targetFileName is specified.
overwrite (continued on next page)	xsd:boolean		When checking out an item revision, specify whether or not Dimensions CM is allowed to perform this operation depending on: ■ The existence of a local file of the same name. ■ The status (read-only or writable) of an existing local file of the same name. **NOTE** /NOOVERWRITE —which is normally the default but which can be reassigned using the SET OVERWRITE command— results in a file only being successfully checked out by Dimensions CM if the local (target) file does not already exist or is marked read-only. This is the traditional Dimensions CM behavior (with respect to the file being read-only, the assumption is that if it is writable then the file could potentially be a more recently modified revision of the item that the user does not want to lose).

Argument	Type	Optional?	Description
overwrite (continued)	xsd:boolean		**NOTE** `/OVERWRITE` results in the file being successfully checked out by Dimensions CM irrespective of the existence or writable status of any local (target) file. (This parameter works only if `targetFileName` is specified.)
noMetadata	xsd:boolean		This parameter disables creation and usage of metadata files in the local work area.

Response

Argument	Type	Optional?	Description
itemSpec	xsd:string	N	Returns the item Id that was checked out.
itemURL	xsd:string	N	Returns the url of the checked-out item.
fileContents	xsd:base64Binary	N	Returns file contents in base64 encoding.

Usage

`checkOutItem` provides a method to check out a Dimensions CM item.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:checkOutItem>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
```

```
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:itemSpec>QLARIUS:BL_QUOTE_WEB_1.0</dmw:itemSpec>
      <dmw:projectSpec>QLARIUS:QUOTE_WEB_1.0</dmw:projectSpec>
      <dmw:baselineSpec>QLARIUS:BL_QUOTE_WEB_1.0</dmw:baselineSpec>
      <dmw:fileName>Qlarius_Home\AssemblyInfo.vb</dmw:fileName>
    </dmw:checkOutItem>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:checkOutItemResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:itemSpec>QLARIUS:4K237 ASSEMBLYINFO VB.A-SRC;3.0</ns2:itemSpec>
      <ns2:itemURL>http://acme:8080/dimensions?jsp=api;command=openi;
      object_id=QLARIUS%3A4K237+ASSEMBLYINFO+VB.A-SRC%3B3.0;DB_CONN=DIM10;
      DB_NAME=CM_TYPICAL</ns2:itemURL>
      <ns2:fileContents>SW1wb3J0cyBTeXN0ZW0NCkltcG9ydHMgU3lzdGVtLlJlZmxlY3Rpb
      24NCkltcG9ydHMgU3lzdGVtLlJ1bnRpbWUuSW50ZXJvcFNlcnZpY2VzDQoNCicgR2VuZXJh
      bCBJbmZvcm1hdGlvbiBhYm91dCBhbiBhc3NlbWJseSBpcyBjb250cm9sbGVkIHRocm91Z2g
      gdGhlIGZvbGxvd2luZyANCicgc2V0IG9mIGF0dHJpYnV0ZXMuIENoYW5nZSB0aGVzZSBhdH
      RyaWJ1dGUgdmFsdWVzIHRvIG1vZGlmeSB0aGUgaW5mb3JtYXRpb24NCicgYXNzb2NpYXRlZ
      CB3aXRoIGFuIGFzc2VtYmx5Lg0KDQonIG1vcmUgY29tbWVudHMgaGVyZQ0KDQonIFJldmll
      dyB0aGUgdmFsdWVzIG9mIHRoZSBhc3NlbWJseSBhdHRyaWJ1dGVzDQoNCjxBc3NlbWJseTo
      gQXNzZW1ibHlUaXRsZSgiIik+IA0KPEFzc2VtYmx5OiBBc3NlbWJseURlc2NyaXB0aW9uKC
      IiKT4gDQo8QXNzZW1ibHk6IEFzc2VtYmx5Q29tcGFueSgiIik+IA0KPEFzc2VtYmx5OiBBc
      3NlbWJseVByb2R1Y3QoIiIpPiANCjxBc3NlbWJseTogQXNzZW1ibHlDb3B5cmlnaHQoIiIp
      PiANCjxBc3NlbWJseTogQXNzZW1ibHlUcmFkZW1hcmsoIiIpPiANCjxBc3NlbWJseTogQ0x
      TQ29tcGxpYW50KFRydWUpPiANCg0KJ1RoZSBmb2xsb3dpbmcgR1VJRCBpcyBmb3IgdGhlIE
      lEIG9mIHRoZSB0eXBlbGliIGlmIHRoaXMgcHJvamVjdCBpcyBleHBvc2VkIHRvIENPTQ0KP
      EFzc2VtYmx5OiBHdWlkKCJCQkRCQTY3RS1CRUZDLTQyQTEtOUVFMS1FQTk0MTk0NzU4Q0Yi
      KT4gDQoNCicgVmVyc2lvbiBpbmZvcm1hdGlvbiBmb3IgYW4gYXNzZW1ibHkgY29uc2lzdHM
      gb2YgdGhlIGZvbGxvd2luZyBmb3VyIHZhbHVlczoNCicNCicgICAgICBNYWpvciBWZXJzaW
      9uDQonICAgICAgTWlub3IgVmVyc2lvbiANCicgICAgICBCdWlsZCBOdW1iZXINCicgICAgI
      CBSZXZpc2lvbg0KJw0KJyBZb3UgY2FuIHNwZWNpZnkgYWxsIHRoZSB2YWx1ZXMgb3IgeW91
      IGNhbiBkZWZhdWx0IHRoZSBCdWlsZCBhbmQgUmV2aXNpb24gTnVtVmVycyANCicgYnkgdXN
      pbmcgdGhlICcqJyBhcyBzaG93biBiZWxvdzoNCg0KPEFzc2VtYmx5OiBBc3NlbWJseVZlcn
      Npb24oIjEuMC4qIik+IA0K</ns2:fileContents>
    </ns2:checkOutItemResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createDeploymentArea

Description

The `createDeploymentArea` web service creates a new deployment area.

Arguments

Argument	Type	Optional?	Description
`connectionDetails`	`basic:ConnectionDetails`	Y	Connection details for the Dimensions server.
`name`	`xsd:string`	N	Name of the new deployment area.
`networkNode`	`xsd:string`	N	Machine hosting the new area.
`directory`	`xsd:string`	N	Directory or partitioned data set where the area is located.
`detailedDescription`	`xsd:string`	Y	Description of the area.
`user`	`areas:User`	Y	Login information for the operating system user account or credential set that will own files transferred into the area. This may be a userid and password combination, or the ID of a credential set held by the server.
`fetchExpanded`	`xsd:boolean`	Y	Whether item header substitution variables are expanded when items are fetched to the area. If this is not specified, it defaults to true.
`owner`	`xsd:string`	Y	User or group that will own the new area. If this is not specified, the user that created the area is set as owner. The owner has the right to manage the definition of the area.
`stage`	`xsd:string`	N	Stage with which the area is associated.
`transferScripts`	`basic:TransferScripts`	Y	Transfer script set of pre / post / fail scripts.
`scriptParameters`	`basic:ScriptParameters`	Y	Set of named script parameters to be passed to scripts. Each parameter may be single or multi-valued.

Argument	Type	Optional?	Description
libraryCacheArea	xsd:string	Y	Library cache area that is defined by the CLCA (Create Library Cache Area) command. When executing a command that gets files, Dimensions checks whether the library cache area associated with the project or stream already contains a coy of the files. If so, Dimensions copies the files from the cache to the area rather than from the item library, improving performance in certain cases.
status	areas:Status	Y	The status of the area. If the status is ONLINE, the area may participate in file transfer operations. If the status is OFFLINE, the area is excluded from any file transfers. If this qualifier is not specified, an area with status ONLINE is created.
filter	xsd:string	Y	Name of the area filter to be used when deploying.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

createDeploymentArea provides a method to create a new deployment area. If a failure occurs, a SOAP fault is returned that contains the faultCode and faultString. The faultCode identifies whether the fault occurred on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or incorrectly named Dimensions product). The faultString will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:createDeploymentArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:deploymentArea>
            <ns1:name>sample_deploymentarea</ns1:name>
            <ns1:networkNode>localhost</ns1:networkNode>
            <ns1:directory>C:\sample_deploymentarea</ns1:directory>
            <ns1:detailedDescription>This is a sample deployment
area</ns1:detailedDescription>
            <ns1:fetchExpanded>true</ns1:fetchExpanded>
            <ns1:stage>DEV</ns1:stage>
            <ns1:transferScripts>
               <ns:preScript>pre_script</ns:preScript>
            </ns1:transferScripts>
            <ns1:scriptParameters>
               <ns:parameter>
                  <ns:name>param1</ns:name>
                  <ns:type>Single_Value</ns:type>
                  <ns:value>value1</ns:value>
               </ns:parameter>
            </ns1:scriptParameters>
            <ns1:status>OFFLINE</ns1:status>
         </dmw:deploymentArea>
      </dmw:createDeploymentArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:createDeploymentAreaResponse
xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:createDeploymentAreaResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

createDesignPart

Description

The `createDesignPart` Web service provides a method to create a Dimensions CM design part using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
designPart	parts:NewDesignPart	N	Specifies attributes for creating a new design part.

NewDesignPart* Structure			
Argument	**Type**	**Optional?**	**Description**
partSpec	xsd:string	N	`<product-id>:<part-id>.<variant>;<pcs>` If the `<variant>` omitted, the default (specified when the product was defined) is used. If the `<pcs>` if omitted, the PCS for new variants (specified when the product was defined) is used.
category	xsd:string	N	Specifies the category of design part.
parentPartSpec	xsd:string	N	`<product-id>:<part-id>.<variant>;<pcs>` The `<variant>` may be omitted if only one exists. The `<pcs>` if ignored; the current PCS is always used.
description	xsd:string	N	Mandatory text to describe the function of the design part.
attributes	basic:AttributeArray	Y	Specifies user defined attributes for a design part that have also been declared as usable for a specified product.

Response

Argument	Type	Optional?	Description
designPartSpec	xsd:string	N	Returns the part Id that was created.

Usage

`createDesignPart` provides a method to create a Dimensions CM design part.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1" xmlns:ns1="http://serena.com/
    dimensions/parts/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createDesignPart>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:designPart>
        <ns1:partSpec>QLARIUS:MY_PART</ns1:partSpec>
        <ns1:category>MODULE</ns1:category>
        <ns1:parentPartSpec>QLARIUS:QLARIUS.A;1</ns1:parentPartSpec>
        <ns1:description>Some description</ns1:description>
      </dmw:designPart>
    </dmw:createDesignPart>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createDesignPartResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:designPartSpec>QLARIUS:MY_PART.;</ns2:designPartSpec>
    </ns2:createDesignPartResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createDesignPartBaseline

Description

The `createDesignPartBaseline` Web service provides a method to create a Dimensions CM baseline using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselineSpec	xsd:string	N	Specifies the id for new baseline.
parentPartSpec	xsd:string	N	Specifies the parent baseline for which a new baseline is being assigned.
parentPartLevel	xsd:int	Y	Optionally, you may elect to restrict the baseline to a given number of levels in the design tree structure. The default of `0` (zero) signifies all levels below the design part selected by `/PART`. For example, `1` signifies that items related to the selected design part only will be processed.
type	xsd:string	Y	Specifies the type of baseline being created.If omitted, the default type is `RELEASE` if a `<template-id>` has been specified. (If `/TYPE` is not specified, then by default, a design baseline is created, which is simply a snapshot of the current stage of product development, and is not therefore expected to need an approval lifecycle.)
templateId	xsd:string	Y	This is the identity of the item or request baseline-template. (The latter type of template was introduced in the Dimensions Release 9.1, see the sub-section Request Baseline Templates in the *Command-Line Reference* for more details.) This must be: ■ Specified when creating a release-baseline. ■ Omitted when creating a design baseline.

Argument	Type	Optional?	Description
projectSpec	xsd:string	Y	If specified, only the items in the project are considered for baselining; otherwise, only the items in the user's current project are considered. See also, the Dimensions CM LCK command in the *Command-Line Reference* concerning the comment about locking the project when baselining. If `/SCOPE=WORKSET` is specified, this qualifier specifies the project/stream to be baselined, not the project/stream to be used to constrain part-based revision selection.
attributes	basic:AttributeArray	Y	Specifies attributes for creating a new baseline.
requestIds	basic:RequestIdArray	Y	Identifies a request to which the new baseline is to be related `In Response To`. If the template specified is a request template, the above request identification will be overridden to specify the list of parent requests that will be used for the `CBL` command.
recursive	xsd:boolean	Y	By default, requests that are related as dependent will be processed by the `CBL` command. If `/SCOPE=WORKSET` is specified, this qualifier controls whether child collections are baselined. The default is `TRUE`.
includeInfo	xsd:boolean	Y	Optionally allow the inclusion of items related to requests via an `Info` relationship. By default, only items that are related to requests via an `In Response To` relationship will be included. If `/SCOPE=WORKSET` is specified, this qualifier controls whether child collections with `Info` relationships are baselined. A child is considered to have an `Info` relationship if the relationship does not specify a relative location.
includeClosed	xsd:boolean	Y	Include closed requests when processing requests for request baselines. The default is `/NOINCLUDE_CLOSED` (do not include requests).
requirementIds	basic:RequirementSpecArray	Y	Optionally specifies a comma separate list of Dimensions RM requirements.

Response

Argument	Type	Optional?	Description
baselineSpec	xsd:string	N	Returns the baseline Id that was created.
baselineURL	xsd:string	N	Returns the url for a new created baseline.

Usage

`createDesignPartBaseline` provides a method to create a Dimensions CM baseline.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createDesignPartBaseline>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselineSpec>QLARIUS:TEST3</dmw:baselineSpec>
      <dmw:parentPartSpec>QLARIUS:QLARIUS.A;1</dmw:parentPartSpec>
      <dmw:parentPartLevel>1</dmw:parentPartLevel>
      <dmw:type>BASELINE</dmw:type>
      <dmw:templateId>MYTEMPLATE</dmw:templateId>
      <dmw:projectSpec>QLARIUS:VS_TYPICAL_1.0</dmw:projectSpec>
      <dmw:includeInfo>false</dmw:includeInfo>
      <dmw:includeClosed>false</dmw:includeClosed>
      <dmw:requirementIds>
        <ns:requirementSpec>MODEL.MODEL_10;10{ UW DESKTOP NET 1.0+ALPHA+1012T4+HTTP://
    ACME/RTMBROWSER }</ns:requirementSpec>
```

```
      </dmw:requirementIds>
    </dmw:createDesignPartBaseline>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createDesignPartBaselineResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:baselineSpec>QLARIUS:TEST3</ns2:baselineSpec>
      <ns2:baselineURL>http://acme:8080/
    dimensions?jsp=api&command=openb&object_id=QLARIUS%3ATEST3&DB_CONN=1012T4&DB_NAME=
    CM_TYPICAL</ns2:baselineURL>
    </ns2:createDesignPartBaselineResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createProject

Description

The `createProject` Web service provides a method to create a Dimensions CM project using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
project	prj:NewProject	N	Specifies attributes for a new project.

NewProject* Structure			
Argument	**Type**	**Optional?**	**Description**
projectSpec	xsd:string	N	Specifies the project specification.
description	xsd:string	N	Specifies the description to be attached to the project definition.
type	xsd:string	Y	Specifies the type of the project. If this qualifier is not specified, the type name `WORKSET` is used.
sourceProjectSpec	xsd:string	Y	Specifies the project on which to base the new project.
sourceBaseline Spec	xsd:string	Y	Specifies the baseline on which to base the new project.
attributes	basic:AttributeArray	Y	Specifies the user-defined attribute values for this project.
status	prj:ProjectStatus	Y	Allows the new project to be created in either a Locked or Unlocked (default) state.
revisionType	prj:ProjectRevisionType	Y	Specifies the project revision type: Branch or Trunk
autoRevision	xsd:boolean	Y	Optional qualifier to tell Dimensions CM to automatically generate a new revision each time an <item-spec> is edited/ updated.
Branches	prj:ProjectBranches	Y	List of branches names within a project.
defaultBranch	xsd:string	Y	Selects from the valid list of `<branch-ids>`. If a default `<branch-id >` is not defined, the first `<branch-id>` in the valid-list of `<branch-ids>` is taken as the default.
pathControl	xsd:boolean	Y	Specifies whether a request is required to perform refactoring operations in this project.

NewProject* Structure			
Argument	**Type**	**Optional?**	**Description**
copyOnDeploy	xsd:boolean	Y	An optional qualifier that specifies whether files that are deployed from an earlier stage to a newer stage will be copied between areas rather than moved (they are moved by default).
keepStage	xsd:boolean	Y	When creating a project based on another project, specify this optional qualifier to avoid resetting the stage of item revisions in the new project to the initial stage but to instead keep the stage of the item revisions from the source project. The default behavior (when this qualifier is not specified) is to reset the stage of all items in the new project to the initial stage. **NOTE** This qualifier can only be used when the new project uses the manual deployment model.
deploymentModel	xsd:string	Y	Specifies whether the project uses a manual or automatic deployment model.
deploymentMethod	xsd:string	Y	Specifies whether the project uses request/item deployment or baseline/ item deployment methods. Can only be specified if the project uses the manual deployment model.

Response

Argument	Type	Optional?	Description
projectSpec	xsd:string	N	Returns the project Id that was created.
projectURL	xsd:string	N	Returns the url for a new created project.

Usage

`createProject` provides a method to create a Dimensions CM project.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/projects/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:createProject>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:project>
            <ns1:projectSpec>QLARIUS:MyProject</ns1:projectSpec>
            <ns1:description>Some description</ns1:description>
            <ns1:type>WORKSET</ns1:type>
            <ns1:sourceProjectSpec>QLARIUS:JAVA_BRANCHA_PRJ</ns1:sourceProjectSpec>
            <ns1:status>Unlocked</ns1:status>
            <ns1:autoRevision>true</ns1:autoRevision>
            <ns1:deploymentModel>Manual</ns1:deploymentModel>
            <ns1:deploymentMethod>Request</ns1:deploymentMethod>
            <ns1:keepStage>true</ns1:keepStage>
         </dmw:project>
      </dmw:createProject>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createProjectResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:projectSpec>QLARIUS:MyProject</ns2:projectSpec>
<ns2:projectURL>http://acme:8080/dimensions?jsp=api&command=openw
&object_id=QLARIUS%3AMyProject&DB_CONN=DIM10&DB_NAME=CM_TYPICAL
</ns2:projectURL>
    </ns2:createProjectResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createProjectBaseline

IMPORTANT! When creating a "project" baseline (a tip baseline), there is no way to specify a template using the `createProjectBaseline` Web service because such functionality is not supported by the Dimensions CM server command utilized by the Web service. (Project baselines are always full baselines.)

If you want to create a baseline using a template then you must use the `createDesignPartBaseline` web service.

Description

The `createProjectBaseline` Web service provides a method to create a Dimensions CM baseline using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselineSpec	xsd:string	N	Specifies the id for a new baseline.
parentProject Spec	xsd:string	N	`<project-spec>` comprises: `<product-id>:<project-id>` and is optional. If specified, only the items in the project/stream are considered for baselining; otherwise, only the items in the user's current project/stream are considered. See also, the Dimensions CM LCK command in the *Command-Line Reference* concerning the comment about locking the project when baselining. If `/SCOPE=WORKSET` is specified, this qualifier specifies the project/stream to be baselined, not the project/stream to be used to constrain part-based revision selection.
type	xsd:string	Y	Specifies the type of baseline being created. If omitted, the default type is `RELEASE` if a `<template-id>` has been specified. (If `/TYPE` is not specified, then by default, a design baseline is created, which is simply a snapshot of the current stage of product development, and is not therefore expected to need an approval lifecycle.)
attributes	basic:AttributeArray	Y	Specifies attributes for creating a new baseline.

Argument	Type	Optional?	Description
requestIds	basic:RequestIdArray	Y	Identifies a request to which the new baseline is to be related `In Response To`.
recursive	xsd:boolean	Y	`/CANCEL_TRAVERSE` halts the traversal of dependent requests. By default, requests that are related as dependent will be processed by the `CBL` command.If `/SCOPE=WORKSET` is specified, this qualifier controls whether child collections are baselined. The default is `TRUE`.
startingBaseline Spec	xsd:string	Y	`/BASELINE`. If you are creating a new baseline via a request/change document baseline template, this option creates a reference, or starting point, to which requests identified by a template can be applied.

Response

Argument	Type	Optional?	Description
baselineSpec	xsd:string	N	Returns the baseline Id that was created.
baselineURL	xsd:string	N	Returns the url for a new created baseline.

Usage

`createProjectBaseline` provides a method to create a Dimensions CM baseline.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createProjectBaseline>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselineSpec>QLARIUS:Mybaseline</dmw:baselineSpec>
      <dmw:parentProjectSpec>QLARIUS:UW_JAVA_2.0</dmw:parentProjectSpec>
      <dmw:type>BASELINE</dmw:type>
      <dmw:recursive>true</dmw:recursive>
    </dmw:createProjectBaseline>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createProjectBaselineResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:baselineSpec>QLARIUS:Mybaseline</ns2:baselineSpec>
<ns2:baselineURL>http://acme:8080/dimensions?jsp=api&command=openb
&object_id=QLARIUS%3AMybaseline&DB_CONN=DIM10&DB_NAME=CM_TYPICAL
</ns2:baselineURL>
    </ns2:createProjectBaselineResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createRequest

Description

The `createRequest` Web service provides a method to create a Dimensions CM request object using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
request	requests:NewRequest	N	The request object to be created.
draft	xsd:boolean	Y	This flag indicates if the change request is to be created as a draft. The default is that the change request is not created as a draft.

NewRequest* Structure			
Argument	**Type**	**Optional?**	**Description**
type	xsd:string	N	Specifies the type of request to be created.
productId	xsd:string	N	Specifies the product that is to own the new request.
title	xsd:string	N	Specifies the request title.
detailedDescription	xsd:string	N	Specifies the detailed description of the request.
attributes	basic:AttributeArray	Y	Specifies user defined attributes.
relatedParts	basic:RequestRelatedPart Array	Y	Specifies the list of related parts to request.
relatedItems	basic:RequestRelatedItem Array	Y	Specifies the list of related items to request.
relatedRequests	basic:RequestRelated RequestArray	Y	Specifies the list of related requests to request.
relatedBaselines	basic:RequestRelated BaselineArray	Y	Specifies the list of related baselines to request.
related Requirements	basic:RequestRelated RequirementArray	Y	Specifies the list of related requirements to request.
relatedProjectSpec	xsd:string	Y	Specifies the list of related projects to request.
basedOn	requests:BasedOnRequest**	Y	Details of based on request.

BasedOnRequest** Structure			
Argument	**Type**	**Optional?**	**Description**
requestId	xsd:string	N	Used to base (i.e. prime) the creation of the new request on the attributes of the specified request.
relationshipType	xsd:string	Y	Specifies the relationship type between the newly created request and the base request. The relationship is a bi-directional link, the new request is the child and the base request is the parent.

Response

Argument	Type	Optional?	Description
requestId	xsd:string	N	Id of the request created.
requestURL	xsd:string	N	URL to the request created.

Usage

`createRequest` provides a method to create a Dimensions CM request object.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/requests/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createRequest>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:request>
        <ns1:type>CR</ns1:type>
        <ns1:productId>QLARIUS</ns1:productId>
        <ns1:title>WS2 Test 12/9/2008 16:20</ns1:title>
        <ns1:detailedDescription>description string</ns1:detailedDescription>
        <ns1:attributes>
          <ns:attribute>
            <ns:name>Severity</ns:name>
            <ns:datatype>Char</ns:datatype>
            <ns:type>Single_Value</ns:type>
            <ns:value>1</ns:value>
</ns:attribute>
        </ns1:attributes>
      </dmw:request>
    </dmw:createRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:requestId>QLARIUS_CR_420</ns2:requestId>
      <ns2:requestURL>http://servername:8080/dimensions?jsp=api&
command=opencd&object_id=QLARIUS_CR_420&DB_CONN=DIM10&
DB_NAME=QLARIUS_CM</ns2:requestURL>
    </ns2:createRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createRevisedBaseline

Description

The `createRevisedBaseline` Web service provides a method to create a Dimensions CM revised baseline using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
newBaselineSpec	xsd:string	N	The identity to be given to the revised baseline being created.
existingBaseline Spec	xsd:string	N	The identity of the existing release baseline to be used to create the revised baseline.
type	xsd:string	N	Specifies the type of the revised baseline being created. If omitted, the type will be the same as that of `<existing-baseline-spec>`.
updateRequestIds	basic:RequestIdArray	Y	Identifies one or more request trees (see the `recursive` argument for details) within which there are item revisions related `In Response To`.
removeRequestIds	basic:RequestIdArray	Y	Identifies one or more request trees (see the `recursive` argument for details) within which there are item revisions related as `Affected`.
update Requirement Specs	basic:RequirementSpecArray	Y	Optionally specifies a comma separated list of Dimensions RM requirements to be updated, where each requirement comprises: `<requirement_spec>{container_name+project_name+dbname+rm_browser_url}`
remove Requirement Specs	basic:RequirementSpecArray	Y	Optionally specifies a comma separated list of Dimensions RM requirements to be removed, where each requirement comprises: `<requirement_spec>{container_name+project_name+dbname+rm_browser_url}`

Argument	Type	Optional?	Description
projectSpec	xsd:string	Y	This optionally specifies the project/stream to be used for this command; failing this, the user's current project/stream will be taken. When an item has been selected for addition to the baseline, the project file name will be taken from this project/stream if another revision is not already in the project/stream.
attributes	basic:AttributeArray	Y	Sets values for one of more attributes of the new baseline, where each `<attrN>` is the `Variable Name` defined for one of the user-defined attributes for baselines, which has also been declared as usable for this `<product-id>` and `<baseline-type>`, and `<valueN>` is the substitution value to be given to this attribute.
recursive	xsd:boolean	Y	Indicates that, in the above processing for `/UPDATE` and `/REMOVE`, traversal of tree structures is not to be performed, and that only the requests cited in the lists are to be inspected for item revisions that have been related, respectively, `In Response To` and as `Affected`. If omitted, the processing is of tree-structures as mentioned above. The default is `TRUE`.

Response

Argument	Type	Optional?	Description
baselineSpec	xsd:string	N	Id of the created baseline.
baselineURL	xsd:string	N	URL to the created baseline.

Usage

`createRevisedBaseline` provides a method to create a Dimensions CM revised baseline.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createRevisedBaseline>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:newBaselineSpec>QLARIUS:CREATEREVISEDBASELINE</dmw:newBaselineSpec>
     <dmw:existingBaselineSpec>QLARIUS:TEST1</dmw:existingBaselineSpec>
      <dmw:type>BASELINE</dmw:type>
      <dmw:updateRequirementSpecs>
        <ns:requirementSpec>MODEL.MODEL_11;11{ UW DESKTOP NET 1.0+ALPHA+1012T4+HTTP://
    ACME/RTMBROWSER/ }</ns:requirementSpec>
      </dmw:updateRequirementSpecs>
      <dmw:projectSpec>QLARIUS:VS_TYPICAL_1.0</dmw:projectSpec>
      <dmw:recursive>false</dmw:recursive>
    </dmw:createRevisedBaseline>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createRevisedBaselineResponse
xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:baselineSpec>QLARIUS:CREATEREVISEDBASELINE</ns2:baselineSpec>
      <ns2:baselineURL>http://acme:8080/dimensions?jsp=api&command=openb&
object_id=QLARIUS%3ACREATEREVISEDBASELINE
&DB_CONN=1012T4&DB_NAME=CM_TYPICAL</ns2:baselineURL>
    </ns2:createRevisedBaselineResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createScheduleJob

Description

The `createScheduleJob` Web service provides a method to create a Dimensions CM scheduled job using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
scheduleJob	basic:NewScheduleJob*	N	This structure specifies the details for a new job.

NewScheduleJob* Structure			
Argument	**Type**	**Optional?**	**Description**
jobName	xsd:string	N	Specifies the job name.
startTime	xsd:dateTime	N	Specifies the job starting time in the SOAP format `[-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]`.
status	impl:JobStatus	Y	Specifies an optional job status. The available statuses are: ■ Inactive (default) ■ Active
repeat	impl:JobRepeat	Y	`repeatTime` specifies an optional repeat time. It can be one of: ■ `0`, `10`, `20`, `30`, `40`, `50`, or `60` for minutes ■ `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, or `12` for hours ■ `0`, `1`, `2`, `3`, `4`, `5`, `6`, or `7` for days `repeatType` specifies a repeat type of `repeat time` and can be one of: ■ Minutes ■ Hours ■ Days
description	xsd:string	Y	Specifies the job description.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`createScheduleJob` provides a method to create a schedule job in Dimensions CM. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise, false.
objectSpec	xsd:string	Y	Contains a newly created job name.
url	xsd:string	Y	Not used.
description	xsd:string	Y	Not used.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createScheduleJob>
     <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:scheduleJob>
        <ns:jobName>MyJob</ns:jobName>
        <ns:startTime>2011-01-01T02:00:00Z</ns:startTime>
        <ns:status>Active</ns:status>
```

```
        <ns:repeat>
          <ns:repeatTime>2</ns:repeatTime>
          <ns:repeatType>Days</ns:repeatType>
        </ns:repeat>
        <ns:description>My description</ns:description>
      </dmw:scheduleJob>
    </dmw:createScheduleJob>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createScheduleJobResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:objectSpec>MyJob</ns2:objectSpec>
      </ns2:result>
    </ns2:createScheduleJobResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createStream

Description

The `createStream` Web Service provides a method to create a Dimensions CM Stream using the data supplied.

Arguments

Argument	Type	Optional?	Description
`connectionDetails`	`basic:ConnectionDetails`	Y	Connection details for the Dimensions server.
`stream`	`prj:NewStream*`	N	Attributes for the new stream.

NewStream* Structure			
Argument	**Type**	**Optional?**	**Description**
`streamSpec`	`xsd:string`	N	Specifies the stream name.
`description`	`xsd:string`	N	Specifies the description to be attached to the stream definition.
`sourceStreamSpec`	`xsd:string`	Y	Specifies the stream on which to base the new stream.
`sourceBaselineSpec`	`xsd:string`	Y	Specifies the baseline on which to base the new stream.
`attributes`	`basic:AttributeArray`	Y	Specifies the user-defined attribute values for this stream.
`defaultBranch`	`xsd:string`	Y	Specifies the default branch for new item revisions in the stream. This must be unique within the base database. If omitted, defaults to the stream-id.
`CMRules`	`prj:ProjectCMRulesState`	Y	Incorporates these parameters: `or/off` - Specifies whether a request is required when creating new item revisions in the stream. Note that this option does not check whether there is a valid relationship between the request type and item type. `default` - Specifies whether CM rules are fully validated for the supplied request type and that a valid relationship exists between the item type and request type.
`pathControl`	`xsd:boolean`	Y	Specifies whether a request is required to perform refactoring operations in this stream.

NewStream* Structure			
Argument	**Type**	**Optional?**	**Description**
importsAllRevisions	xsd:boolean	Y	If creating the stream from another stream imports all revisions of the items in the parent stream.
copyConfigs	xsd:boolean	Y	Specifies whether build configurations will be copied from the stream referenced by the sourceStreamSpec parameter to the new stream. The default behavior depends on the value of the parameter DM_BUILD_COPY_CONFIGS in the DM.CFG file. This not set by default. If it is set, the the default is COPY_CONFIGS, otherwise it is NOCOPY_CONFIGS.
force	xsd:boolean	Y	When copying build information from the stream referenced by the sourceStreamSpec parameter to the new stream, this option specifies whether the createStream will abort when the first error is encountered, or whether the process will continue to produce as many diagnostic messages as possible. No force means that the process will stop after the first error. The default behavior depends on the value of the parameter DM_BUILD_COPY_FORCE in the DM.CFG file. This not set by default. If it is set, the the default is force, otherwise it is no force.

Response

Argument	Type	Optional?	Description
streamSpec	xsd:string	N	Returns the stream specification that was created
streamURL	xsd:string	N	Returns the url for a new created stream

Usage

createStream provides a method to create a Dimensions CM Stream. If a failure occurs a SOAP fault is returned. This fault contains the faultCode and faultString. The faultCode identifies whether the fault occurs on the client (for example because of an incorrectly defined service endpoint) or on the server (for example because of SOAP issue or Dimensions server issue such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The faultString will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/dimensions/
    basic/1" xmlns:ns1="http://serena.com/dimensions/projects/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:createStream>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:stream>
        <ns1:streamSpec>QLARIUS:STREAM_01</ns1:streamSpec>
        <ns1:description>Some description for a new stream 01</ns1:description>
        <ns1:sourceBaselineSpec>QLARIUS:BL_JAVA_BRANCHA_STR_02</ns1:sourceBaselineSpec>
        <ns1:attributes>
          <ns:attribute>
            <ns:name>ATTR01</ns:name>
            <ns:datatype>Char</ns:datatype>
            <ns:type>Single_Value</ns:type>
            <ns:value>Some value</ns:value>
          </ns:attribute>
        </ns1:attributes>
        <ns1:defaultBranch>str_01</ns1:defaultBranch>
        <ns1:CMRules>off</ns1:CMRules>
        <ns1:pathControl>false</ns1:pathControl>
        <ns1:importsAllRevisions>false</ns1:importsAllRevisions>
        <ns1:copyConfigs>false</ns1:copyConfigs>
        <ns1:force>false</ns1:force>
      </dmw:stream>
    </dmw:createStream>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:createStreamResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:streamSpec>QLARIUS:STREAM_01</ns2:streamSpec>
      <ns2:streamURL>http://UA6231:8080/
    dimensions?jsp=api&amp;command=openw&amp;object_id=QLARIUS%3ASTREAM_01&amp;DB_CONN=
    DIM10&amp;DB_NAME=CM_TYPICAL</ns2:streamURL>
    </ns2:createStreamResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

createWorkArea

Description

The createStream Web Service creates a new work area.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	Connection details for the Dimensions server.
name	xsd:string	N	Name of the new work area.
networkNode	xsd:string	N	Machine hosting the area.
directory	xsd:string	N	The directory or partitioned data set where the area is located.
detailedDescription	xsd:string	Y	Description of the work area.
user	areas:User	Y	Login information for the OS user account or credential set that will own files transferred into the area. The may be a userid or password combination, or the ID of a credential set held by the server.
fetchExpanded	xsd:boolean	Y	Whether item header substitution variables will be expanded with item file are fetched to the are. If this is not specified, then it defaults to true.
owner	xsd:string	Y	User or group that will be the owner of the new area. If /OWNER is not specified, the user who created the area is set as owner. The area owner has the right to manage the definition of the area.
userList	basic:UserNameArray	Y	Users and / or groups that are granted the right to use this area. If the list is empty, any user can specify the area as a working location and get, check out, or check in items using the area.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`createWorkArea` provides a method to create a new work area. If a failure occurs a SOAP fault is returned. This fault contains the `faultCode` and `faultString`. The `faultCode` identifies whether the fault occured on the client (for example because of an incorrectly defined service endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The `faultString` identify\ies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:createWorkArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:workArea>
            <ns1:name>sample_workarea</ns1:name>
            <ns1:networkNode>localhost</ns1:networkNode>
            <ns1:directory>c:\sample_workarea</ns1:directory>
            <ns1:detailedDescription>This is a work
area</ns1:detailedDescription>
            <ns1:userList>
               <ns:userName>BOBBY</ns:userName>
               <ns:userName>TED</ns:userName>
            </ns1:userList>
         </dmw:workArea>
      </dmw:createWorkArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:createWorkAreaResponse xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:createWorkAreaResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

delegateRequest

Description

The `delegateRequest` Web service provides a method to delegate a Dimensions CM request to specified users using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	Identifies the request for which the delegation is to be made.
userNames	basic:UserNameArray	N	Identifies by login user name one or more Dimensions CM users to whom delegation of the role is to be made for this request.
role	xsd:string	N	Identifies the role title to be delegated.
capability	basic:UserRoleCapability	N	Specifies that this role delegation is to be `Primary` for primary capability, `Secondary` for secondary capability (default), or `Leader` for leader capability. Only one user and only `/ADD` or `/REPLACE` are valid for delegation of primary capability.
mode	basic:DelegateRequestMode	N	Specifies that the users are to have this role for this request in addition to those who already have it. (`Add`, `Replace`, or `Delete`)
delegateItems	xsd:boolean	N	When delegating a request to a user, also automatically delegate those items related as `Affected` and `In Response To` to that user as well.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`delegateRequest` Web service provides a method to delegate a Dimensions CM request to specified users.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:delegateRequest>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_TASK_7</dmw:requestId>
      <dmw:userNames>
        <ns:userName>DAN</ns:userName>
        <ns:userName>DAWN</ns:userName>
      </dmw:userNames>
      <dmw:role>IMPLEMENTOR</dmw:role>
      <dmw:capability>Secondary</dmw:capability>
      <dmw:mode>Add</dmw:mode>
      <dmw:delegateItems>true</dmw:delegateItems>
    </dmw:delegateRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:delegateRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:delegateRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

delegateRequestForReplication

Description

The `delegateRequestForReplication` Web service provides a method to delegate a Dimensions CM request to a replication site using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	Identifies the request for which the delegation is to be made.
site	xsd:string	N	Specifies a replication site to which a request will be delegated.
mode	basic:DelegateRequestMode	N	Specifies that the users are to have this role for this request in addition to those who already have it. (`Add`, `Replace`, or `Delete`)
delegateItems	xsd:boolean	N	When delegating a request to a user, also automatically delegate those items related as `Affected` and `In Response To` to that user as well.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`delegateRequestForReplication` Web service provides a method to delegate a Dimensions CM request to a replication site.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:delegateRequestForReplication>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_TASK_7</dmw:requestId>
      <dmw:site>ACME:cm_typical_SECOND@DIM10</dmw:site>
      <dmw:mode>Replace</dmw:mode>
      <dmw:delegateItems>true</dmw:delegateItems>
    </dmw:delegateRequestForReplication>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:delegateRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:delegateRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

demoteBaselines

Description

The `demoteBaselines` Web service provides a method to demote one or more Dimensions CM baselines using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselines	basic: BaselineSpecArray	N	The list of baseline specifications that will be processed by this command.
projectSpec	xsd:string	N	The name of the project the baseline or baselines are in.
comment	xsd:string	Y	Comment text associated with the demotion operation.
stage	xsd:string	Y	Specifies the target stage to demote the object to. This must be a stage from the global stage lifecycle.
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
deploy	xsd: boolean	Y	Cannot be combined with `<areas>` and indicates that `<areas>` is not included on purpose as no deployment is to occur.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`demoteBaselines` Web service provides a method to demote one or more Dimensions CM baselines. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.

Argument	Type	Optional?	Description
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:demoteBaselines>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselines>
        <ns:baselineSpec>QLARIUS:STREAM_A_BASELINE_1.0</ns:baselineSpec>
      </dmw:baselines>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promote Baseline</dmw:comment>
      <dmw:stage>DEVELOPMENT</dmw:stage>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:deploy>true</dmw:deploy>
      <dmw:deployStartTime>2010-10-29T11:30:00+03:00</dmw:deployStartTime>
    </dmw:demoteBaselines>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:demoteBaselinesResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: File 'Qlarius Underwriter/Qlarius Underwriter.ecl;1' was demoted
          File 'Qlarius Underwriter/build-user.xml;1' was demoted
          File 'Qlarius Underwriter/.classpath;1' was demoted
          File 'Qlarius Underwriter/qlarius/utilities/DBIO.java;1' was demoted
          File 'Qlarius Underwriter/qlarius/utilities/FileIO.java;1' was demoted
          File 'Qlarius Underwriter/.project;1' was demoted
          File 'Qlarius Underwriter/qlarius/utilities/PendingQuote.java;1' was demoted
          File 'Qlarius Underwriter/build.xml;1' was demoted
          File 'Qlarius Underwriter/qlarius/sampledata/sample.xml;1' was demoted
          File 'Qlarius Underwriter/qlarius/utilities/SampleXMLParser.java;1' was
    demoted
          File 'Qlarius Underwriter/qlarius/interfaces/AutoQuote.java;1' was demoted
          File 'Qlarius Underwriter/qlarius/interfaces/HealthQuote.java;1' was demoted
          File 'Qlarius Underwriter/qlarius/interfaces/LifeQuote.java;1' was demoted
          File 'Documents/dmcm_introduction.doc;1' was demoted
          14 file(s) were processed
          Schedule Job DEPLOYMENT_JOB_4217293 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:demoteBaselinesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

demoteItems

Description

The `demoteItems` Web service provides a method to demote one or more Dimensions CM items using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
items	basic:ItemSpecArray	Y	The list of item specifications which will be processed by this command. This will be used in preference to `fileName` in the command parameters if specified.
files	basic:FileNameArray	Y	List of items specified through path to items.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
comment	xsd:string	Y	Comment text associated with the demotion operation.
stage	xsd:string	Y	Specifies the target stage to demote the object to. This must be a stage from the global stage lifecycle.
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
deploy	xsd:boolean	Y	Cannot be combined with `<areas>` and indicates that `<areas>` is not included on purpose as no deployment is to occur. By default is true.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`demoteItems` Web service provides a method to demote one or more Dimensions CM items. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:demoteItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:items>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1</ns:itemSpec>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1</ns:itemSpec>
      </dmw:items>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promoting items</dmw:comment>
      <dmw:stage>DEVELOPMENT</dmw:stage>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
```

```
      <dmw:deploy>true</dmw:deploy>
      <dmw:deployStartTime>2010-10-22T14:22:00+03:00</dmw:deployStartTime>
    </dmw:demoteItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:demoteItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Demoting "QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1"...
          Demoting "QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1"...
          Schedule Job DEPLOYMENT_JOB_4218118 has been succesfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:demoteItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

demoteRequests

Description

The `demoteRequests` Web service provides a method to demote one or more Dimensions CM requests using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requests	basic:RequestIdArray	N	The list of request ids that will be processed by this command.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
comment	xsd:string	Y	Comment text associated with the promotion operation.
stage	xsd:string	Y	Specifies the target stage to demote the object to. This must be a stage from the global stage lifecycle.
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
recursive	xsd:boolean	Y	Whether child requests should be promoted as well
deploy	xsd:boolean	Y	Cannot be combined with `<areas>` and indicates that the `<areas>` is not included on purpose as no deployment is to occur. By default is true.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`demoteRequests` Web service provides a method to demote one or more Dimensions CM requests. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:demoteRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requests>
        <ns:requestId>QLARIUS_CR_13</ns:requestId>
        <ns:requestId>QLARIUS_TASK_6</ns:requestId>
      </dmw:requests>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promoting requests</dmw:comment>
      <dmw:stage>DEVELOPMENT</dmw:stage>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:recursive>false</dmw:recursive>
```

```
      <dmw:deploy>true</dmw:deploy>
      <dmw:deployStartTime>2010-10-21T15:40:00Z</dmw:deployStartTime>
    </dmw:demoteRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:demoteRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Demoting "QLARIUS_CR_13"...
          Demoting "QLARIUS_TASK_6"...
          Schedule Job DEPLOYMENT_JOB_4218036 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:demoteRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

deployBaseline

IMPORTANT! Starting in Dimensions CM 12.1, the `submitDeployBaselines` Web service (see page 465), supersedes the `deployBaseline` Web service. deployBaseline is maintained for backward compatibility, but its use is deprecated.

Description

The `deployBaseline` Web service provides a method to deploy a Dimensions CM baseline using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselineSpec	xsd:string	N	Baseline specification.
projectSpec	xsd:string	Y	Specifies the root project/stream to be used for this command. If the `/WORKSET` qualifier is omitted, the `DPB` command uses the current project/stream.
stage	xsd:string	Y	Specifies the target stage. This must be a stage from the global stage lifecycle, and there must be a transition from the current stage to the new stage in the stage lifecycle in order for the `DPB` command to succeed.
comment	xsd:string	Y	Textual comment.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`deployBaseline` Web service provides a method to deploy a Dimensions CM baseline.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:deployBaseline>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselineSpec>QLARIUS:BL_QUOTE_WEB_1.0</dmw:baselineSpec>
      <dmw:projectSpec>QLARIUS:UW_JAVA_2.0</dmw:projectSpec>
      <dmw:stage>SYSTEM TEST</dmw:stage>
      <dmw:comment>Some comments</dmw:comment>
    </dmw:deployBaseline>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:deployBaselineResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:deployBaselineResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

deployRequest

IMPORTANT! Starting in Dimensions CM 12.1, the `submitDeployRequests` Web service (see page 471), supersedes the `deployRequest` Web service. deployRequest is maintained for backward compatibility, but its use is deprecated.

Description

The `deployRequest` Web service provides a method to deploy a Dimensions CM request using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	Request specification.
projectSpec	xsd:string	Y	Specifies a project or stream. If it is omitted the current user project/stream will be used.
stage	xsd:string	N	Specifies the target stage. This must be a stage from the global stage lifecycle, and there must be a transition from the current stage to the new stage in the stage lifecycle in order for the `DPB` command to succeed.
comment	xsd:string	N	Textual comment.
traverseChildren	xsd:boolean	N	Specifies whether the hierarchy of child requests is traversed.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

`deployRequest` Web service provides a method to deploy a Dimensions CM request.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or

- on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:deployRequest>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_DR_1</dmw:requestId>
      <dmw:stage>SYSTEM TEST</dmw:stage>
      <dmw:comment>Some comments</dmw:comment>
      <dmw:traverseChildren>true</dmw:traverseChildren>
    </dmw:deployRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:deployRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:deployRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

getItems

Description

The `getItems` Web service provides a method to download a Dimensions CM project/stream or baseline items to a specified location using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
projectSpec	xsd:string	Y	The project/stream from which to download the files (The projectSpec or baselineSpec must be specified).
baselineSpec	xsd:string	Y	The baseline from which to download the files. (The `projectSpec` or `baselineSpec` must be specified.)
directory	xsd:string	Y	This parameter specifies a directory path. It is matched to the corresponding project/stream or baseline directory, the files in that project/stream or baseline directory are enumerated, and each one that has not been modified is downloaded.
fileNames	basic:FileNameArray	Y	This optional qualifier specifies a list of file names to be downloaded from the project/stream or baseline.
itemSpecs	basic:ItemSpecArray	Y	This optional qualifier specify a list of item specifications to be downloaded from the project or baseline. This qualifier allows you to efficiently download an arbitrary set of items from Dimensions CM using the complete item specifications.
filter	xsd:string	Y	Specifies that `DOWNLOAD` will get only files that satisfy the criteria.
targetDirectory	xsd:string	Y	Specifies the target directory in "work area" to download files into.
transferScripts	basic:TransferScripts	Y	Specifies the transfer script set to be executed as items are gotten during the command's execution.
codepage	xsd:string	Y	The code page to be associated with the item.
expand	xsd:boolean	Y	Specifies substitution variable expansion.

Argument	Type	Optional?	Description
recursive	xsd:boolean	Y	If a directory is specified and this qualifier is not present, all files in all directories beneath the one specified are downloaded. `/NORECURSIVE` specifies that only files at the specified directory level are downloaded.
touch	xsd:boolean	Y	Sets the modification time of the user file to the current system time. (This parameter works only if `targetDirectory` is specified.)
overwrite	xsd:boolean	Y	By default, `DOWNLOAD` does not overwrite files in the operation root directory that: ■ have no metadata, ■ are locally modified, ■ are checked out to the operation root directory, or ■ correspond to an item different from the one being gotten (that is, files that have different `<product>:<item-id>.<variant>-<type>` pairs). If `/OVERWRITE` is specified, `DOWNLOAD` overwrites such files with the content of corresponding project/stream or baseline item revisions. (This parameter works only if `targetDirectory` is specified.)
conflictCheck	xsd:boolean	Y	Specifies that `DOWNLOAD` will search for unresolved merge conflicts for each item revision to be gotten. If any unresolved conflicts are found, Dimensions CM issues a warning and ignores the corresponding revision. By default, `DOWNLOAD` does not perform this check. (This parameter works only if `targetDirectory` is specified.)
noMetadata	xsd:boolean	Y	This parameter disables creation and usage of metadata files in the local work area.

Response

Argument	Type	Optional?	Description
items	basic:ItemFileFolder	Y	Returns a list of downloaded items.
result	xsd:string	Y	Returns detailed description of operation.

Usage

`getItems` Web service provides a method to download a Dimensions CM project/stream or baseline items to a specified location.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

NOTE There may be a limit to the size of files that can be returned by getItems. For example, the limit for the default Tomcat configuration on 32 bit Windows is 224 MB.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:getItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:projectSpec>QLARIUS:QUOTE_WEB_1.0</dmw:projectSpec>
      <dmw:fileNames>
      <ns:fileName>Qlarius_Home\Global.asax</ns:fileName>
      </dmw:fileNames>
      <dmw:expand>false</dmw:expand>
      <dmw:recursive>true</dmw:recursive>
      <dmw:noMetadata>false</dmw:noMetadata>
    </dmw:getItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:items>
        <ns1:name xmlns:ns1="http://serena.com/dimensions/basic/1"/>
      </ns2:items>
      <ns2:items>
        <ns1:name xmlns:ns1="http://serena.com/dimensions/basic/1">
          Qlarius_Home\
        </ns1:name>
        <file xmlns="http://serena.com/dimensions/basic/1">
          <name>Global.asax</name>
          <modificationTime>2006-01-7T00:14:34+02:00</modificationTime>
    <fileContents>PCVAIEFwcGxpY2F0aW9uIENvZGViZWhpbmQ9Ikdsb2JhbC5h
          c2F4LnZiIiBJbmhlcml0cz0iUWxhcml1c19Ib21lLkdsb2JhbCIgJT4=
          </fileContents>
        </file>
      </ns2:items>
      <ns2:items>
        <ns1:name xmlns:ns1="http://serena.com/dimensions/basic/1">
          Qlarius_Home\.metadata\
        </ns1:name>
        <file xmlns="http://serena.com/dimensions/basic/1">
          <name>Global.asax.metadata-item</name>
          <modificationTime>2008-10-06T15:44:33+02:00</modificationTime>
    <fileContents>dmVyc2lvbj04LjUKaXRlbS11aWQ9NDIwMjU3NgppdGVtLXNw
ZWM9NTE0QzQxNTI0OTU1NTMzQTM0NEIzMjMzMzcyMDQ3NEM0RjQyNDE0QzIwND
E1MzQxNTgyRTQxMkQ1MzUyNDMzQjMxCmZpbGUtdmVyc2lvbj0xCmNoZWNrc3Vt
PTdjNzczNjk2ZWYwMDJkZGUwYTNmZDQ4YjZjNDI1MmQzCmZldGNoLXNpemU9Nz
cKbXRpbWU9MTEzODMxMzY3NAppcy1leHRyYWN0ZWQ9MAppcy1oZWFkZXJzdWJz
dD0wCm1vdmVkLWZyb209Cml0ZW0tc3RhdHVzPTQxNTA1MDUyNEY1NjQ1NDQK
          </fileContents>
        </file>
      </ns2:items>
    </ns2:getItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

getRequestDescription

Description

The `getRequestDescription` Web Service returns a detailed description of the specified request.

Arguments

Argument	Type	Optional	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	RequestId for which a detailed description should be returned.

Response

Argument	Type	Optional	Description
description	xsd:string	N	Detailed description of the request.

Usage

`getRequestDescription` provides a method to obtain a Dimensions CM request's detailed description. If a failure occurs a SOAP fault will be returned.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/
    dimensions/basic/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:getRequestDescription>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>dim10</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:requestId>QLARIUS_CR_16</dmw:requestId>
      </dmw:getRequestDescription>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getRequestDescriptionResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:description>The Qlarius UnderWriter system shall electronically captures
    application details with complex validation of entered data.</ns2:description>
    </ns2:getRequestDescriptionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

getRequests

Description

The `getRequest` Web service provides a method to return a list of Dimensions CM request attributes using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	Specifies the request.
projectSpec	xsd:string	Y	Specifies a project from which related items will be populated.

Response

Argument	Type	Optional?	Description
request	requests:Request	N	List of attributes for the specified requests.

Usage

`getRequest` Web service provides a method to return a list of Dimensions CM request attributes.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:getRequests>
     <dmw:connectionDetails>
   <ns:username>dmsys</ns:username>
   <ns:password>dmsys_test</ns:password>
   <ns:dbName>cm_typical</ns:dbName>
   <ns:dbConnection>dim10</ns:dbConnection>
   <ns:server>localhost</ns:server>
     </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_CR_10</dmw:requestId>
      <dmw:projectSpec>QLARIUS:STREAM_C1</dmw:projectSpec>
    </dmw:getRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:request>
        <ns3:requestId
xmlns:ns3="http://serena.com/dimensions/requests/1">QLARIUS_CR_10</ns3:requestId>
        <ns3:status
xmlns:ns3="http://serena.com/dimensions/requests/1">RAISED</ns3:status>
        <attributes xmlns="http://serena.com/dimensions/requests/1">
          <attribute xmlns="http://serena.com/dimensions/basic/1">
            <name>CONTACT_DETAILS</name>
            <datatype>Char</datatype>
            <type>Multi_Value</type>
            <multivalue>
              <value/>
            </multivalue>
          </attribute>
          <attribute xmlns="http://serena.com/dimensions/basic/1">
            <name>PRIME_SEVERITY</name>
            <datatype>Char</datatype>
            <type>Single_Value</type>
            <value/>
          </attribute>
          <attribute xmlns="http://serena.com/dimensions/basic/1">
            <name>DATE_TEST_PASSED</name>
            <datatype>Date</datatype>
            <type>Single_Value</type>
            <value/>
          </attribute>
```

```
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>COMPLETION_DATE</name>
              <datatype>Date</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>ACTUAL_COMP_DATE</name>
              <datatype>Date</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>SEVERITY</name>
              <datatype>Char</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>DESCRIPTION</name>
              <datatype>Char</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>ACTUAL_DEV_EFFORT</name>
              <datatype>Number</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>EST_DEV_EFFORT</name>
              <datatype>Number</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>DETAILS_OF_THE_SOLUTION</name>
              <datatype>Char</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>DEFER_REASON</name>
              <datatype>Char</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
            <attribute xmlns="http://serena.com/dimensions/basic/1">
              <name>DATE_OF_DEPLOYMENT</name>
              <datatype>Date</datatype>
              <type>Single_Value</type>
              <value/>
            </attribute>
```

```
          <attribute xmlns="http://serena.com/dimensions/basic/1">
            <name>DATE_OF_DEPLOYMENT</name>
            <datatype>Date</datatype>
            <type>Single_Value</type>
            <value/>
          </attribute>
          <attribute xmlns="http://serena.com/dimensions/basic/1">
            <name>RAISED_BY</name>
            <datatype>Char</datatype>
            <type>Single_Value</type>
            <value/>
          </attribute>
        </attributes>
        <relatedParts xmlns="http://serena.com/dimensions/requests/1">
          <RelatedPart xmlns="http://serena.com/dimensions/basic/1">
            <partSpec>QLARIUS:QLARIUS.A;1</partSpec>
            <relationshipType>Affected</relationshipType>
          </RelatedPart>
        </relatedParts>
        <relatedItems xmlns="http://serena.com/dimensions/requests/1"/>
        <relatedRequests xmlns="http://serena.com/dimensions/requests/1"/>
        <relatedBaselines xmlns="http://serena.com/dimensions/requests/1"/>
        <relatedRequirements xmlns="http://serena.com/dimensions/requests/1"/>
        <ns3:relatedProjectSpec xmlns:ns3="http://serena.com/dimensions/requests/
    1">QLARIUS:STREAM_C</ns3:relatedProjectSpec>
        <pendingUsers xmlns="http://serena.com/dimensions/requests/1">
          <pendingUser xmlns="http://serena.com/dimensions/basic/1">
            <userName>CHRIS</userName>
            <role>LEADER</role>
            <capability>Secondary</capability>
          </pendingUser>
          <pendingUser xmlns="http://serena.com/dimensions/basic/1">
            <userName>CLEM</userName>
            <role>DEVELOPER</role>
            <capability>Secondary</capability>
          </pendingUser>
          <pendingUser xmlns="http://serena.com/dimensions/basic/1">
            <userName>DMSYS</userName>
            <role>DEVELOPER</role>
            <capability>Secondary</capability>
          </pendingUser>
          <pendingUser xmlns="http://serena.com/dimensions/basic/1">
            <userName>JOE</userName>
            <role>DEVELOPER</role>
            <capability>Secondary</capability>
          </pendingUser>
          <pendingUser xmlns="http://serena.com/dimensions/basic/1">
            <userName>RICK</userName>
            <role>DEVELOPER</role>
            <capability>Secondary</capability>
          </pendingUser>
```

```
        <pendingUser xmlns="http://serena.com/dimensions/basic/1">
          <userName>RON</userName>
          <role>LEADER</role>
          <capability>Secondary</capability>
        </pendingUser>
      </pendingUsers>
      <ns3:archive xmlns:ns3="http://serena.com/dimensions/requests/1">false</
  ns3:archive>
```

getVersion

Description

The `getVersion` Web service provides a method to determine version information about Dimensions CM or using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM or server.

Response

Argument	Type	Optional?	Description
serverType	xsd:string	N	Server type (Dimensions CM or).
highLevelRelease	xsd:string	N	High level release.
lowLevelRelease	xsd:string	N	Low level release.

Usage

`getVersion` Web service provides a method to determine version information about Dimensions CM or .

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:getVersion>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
    </dmw:getVersion>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getVersionResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:serverType>CM</ns2:serverType>
      <ns2:highLevelRelease>10.2.1</ns2:highLevelRelease>
      <ns2:lowLevelRelease>1</ns2:lowLevelRelease>
    </ns2:getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

listDeploymentAreas

Description

The listDeploymentAreas web service retrieves all work areas, or a single deployment area by name.

Arguments

Argument	Type	Optional?	Description
`connectionDetails`	`basic:ConnectionDetails`	Y	Connection details for the Dimensions server.
`filter`	`areas:DeploymentAreaFilter`	Y	Options for filtering the returned list. Currently only filtering by name is supported.

Response

Argument	Type	Optional?	Description
`deploymentArea`	`areas:WorkArea`	N	Each workArea element represents the details of a retrieved work area.

Usage

`listDeploymentAreas` provides a method to retrieve existing deployment areas. If a failure occurs, a SOAP fault is returned. This fault contains the `faultCode` and `faultString`. The `faultCode` identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The `faultString` identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:listDeploymentAreas>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>Secret001</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>aidy-win2008</ns:server>
         </dmw:connectionDetails>
         <dmw:filter>
            <ns1:name>sample_deploymentarea</ns1:name>
         </dmw:filter>
      </dmw:listDeploymentAreas>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:listDeploymentAreasResponse
xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:deploymentArea xsi:type="ns2:DeploymentArea"
xmlns:ns2="http://serena.com/dimensions/areas/1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <ns2:name>SAMPLE_DEPLOYMENTAREA</ns2:name>
            <ns2:networkNode>AIDY-WIN2008</ns2:networkNode>
            <ns2:directory>C:\sample_deploymentarea\</ns2:directory>
            <ns2:created>
               <ns1:when xmlns:ns1="http://serena.com/dimensions/basic/1">2012-
09-28T10:38:24.000+01:00</ns1:when>
               <ns1:by
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS</ns1:by>
            </ns2:created>
            <ns2:updated>
               <ns1:when xmlns:ns1="http://serena.com/dimensions/basic/1">2012-
09-28T10:59:55.000+01:00</ns1:when>
               <ns1:by
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS</ns1:by>
            </ns2:updated>
            <ns2:detailedDescription>This is a sample deployment
area</ns2:detailedDescription>
            <ns2:owner>DMSYS</ns2:owner>
            <ns2:stage>SIT</ns2:stage>
            <ns2:status>ONLINE</ns2:status>
         </ns4:deploymentArea>
      </ns4:listDeploymentAreasResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

listProjectItems

Description

The `listProjectItems` Web service provides a method to receive a list of items in a specified Dimensions CM project/stream using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
projectSpec	xsd:string	N	Specifies the project/stream from which list of items will be returned.
directory	xsd:string	Y	Specifies a directory path. It is matched to the corresponding project/stream, and the files in that project directory are enumerated.
recursive	basic:ItemTypeArray	Y	Recursively list all project directories from the point defined above.
allRevisions	xsd:boolean	Y	List all or latest revisions of items.
types	basic:ItemTypeArray	Y	A list of item types.
userName	xsd:string	Y	A user name, required only for retrieving of items which are in the user inbox.

Response

Argument	Type	Optional?	Description
items	basic:ItemInfoFolder	N	List of project items.

Usage

`listProjectItems` Web service provides a method to receive a list of items in a specified Dimensions CM project/stream.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:listProjectItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:projectSpec>QLARIUS:UW_DOTNET_1.0</dmw:projectSpec>
      <dmw:directory>Qlarius Underwriter\Qlarius Underwriter</dmw:directory>
      <dmw:recursive>false</dmw:recursive>
      <dmw:allRevisions>true</dmw:allRevisions>
      <dmw:types>
        <ns:itemType>SRC</ns:itemType>
      </dmw:types>
    </dmw:listProjectItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:listProjectItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:items>
        <ns1:name xmlns:ns1="http://serena.com/dimensions/basic/1">Qlarius
    Underwriter\Qlarius Underwriter\</ns1:name>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM ASSEMBLYINFO VB.A-SRC;1.0</itemSpec>
          <fileName>AssemblyInfo.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM DBIO VB.A-SRC;1.0</itemSpec>
          <fileName>dbIO.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
         <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FILEIO VB.A-SRC;1.0</itemSpec>
          <fileName>fileIO.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
```

```
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN AUTOQUOTE RESX.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_autoQuote.resx</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN AUTOQUOTE VB.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_autoQuote.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN HEALTHQUOTE RESX.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_healthQuote.resx</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN HEALTHQUOTE VB.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_healthQuote.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN LIFEQUOTE RESX.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_lifeQuote.rsx</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM FRMMAIN LIFEQUOTE VB.A-SRC;1.0</itemSpec>
          <fileName>FrmMain_lifeQuote.vb</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM QLARIUS UNDERWRITER VBPROJ.A-SRC;1.0</itemSpec>
          <fileName>Qlarius_Underwriter.vbproj</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item>
        <item xmlns="http://serena.com/dimensions/basic/1">
          <itemSpec>QLARIUS:4K3FM QLARIUS UNDERWRITER VBPROJ VSPSCC.A-SRC;1.0</itemSpec>
          <fileName>Qlarius_Underwriter.vbproj.vspscc</fileName>
          <status>APPROVED</status>
          <revision>1.0</revision>
        </item></ns2:items>
    </ns2:listProjectItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

listProjectRequests

Description

The `listProjectRequests` Web service provides a method to receive a list of requests in a specified Dimensions CM project/stream using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
projectSpec	xsd:string	N	Specifies the project/stream from which a list of requests will be returned.
userName	xsd:string	Y	A user name, required only for the retrieving of requests which are in the user inbox.
types	basic:RequestTypeArray	Y	A list of request types.

Response

Argument	Type	Optional?	Description
requestId	xsd:string	N	List of project/stream requests.

Usage

`listProjectRequests` Web service provides a method to receive a list of requests in a specified Dimensions CM project/stream.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:listProjectRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:projectSpec>QLARIUS:UW_DOTNET_1.0</dmw:projectSpec>
      <dmw:types>
        <ns:requestType>BR</ns:requestType>
      </dmw:types>
    </dmw:listProjectRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:listProjectRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:requestId>QLARIUS_BR_2</ns2:requestId>
    </ns2:listProjectRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

listProjects

Description

The `listProjects` Web service provides a method to list projects/streams in Dimensions CM.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.

Response

Argument	Type	Optional?	Description
result	prj:ProjectDetails	N	Specifies the project/stream details of each listed project/stream.

Usage

`listProjects` Web service provides a method to get a list of projects/streams in Dimensions CM. If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:listProjects>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
    </dmw:listProjects>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:listProjectsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns4:projectSpec
xmlns:ns4="http://serena.com/dimensions/projects/1">$GENERIC:$GLOBAL</ns4:projectSpec>
        <ns4:projectType
xmlns:ns4="http://serena.com/dimensions/projects/1">WORKSET</ns4:projectType>
        <ns4:projectStatus
xmlns:ns4="http://serena.com/dimensions/projects/1">UNLOCKED</ns4:projectStatus>
        <ns4:owner
xmlns:ns4="http://serena.com/dimensions/projects/1">DMSYS</ns4:owner>
      </ns2:result>
      <ns2:result>
        <ns4:projectSpec
xmlns:ns4="http://serena.com/dimensions/projects/1">QLARIUS:STREAM_A</ns4:projectSpec>
        <ns4:projectType
xmlns:ns4="http://serena.com/dimensions/projects/1">STREAM</ns4:projectType>
        <ns4:projectStatus
xmlns:ns4="http://serena.com/dimensions/projects/1">OPEN</ns4:projectStatus>
        <ns4:owner
```

```
xmlns:ns4="http://serena.com/dimensions/projects/1">DMSYS</ns4:owner>
        <ns4:projectManager
xmlns:ns4="http://serena.com/dimensions/projects/1">JOE</ns4:projectManager>
        <ns4:projectManager
xmlns:ns4="http://serena.com/dimensions/projects/1">DMSYS</ns4:projectManager>
        <ns4:projectManager
xmlns:ns4="http://serena.com/dimensions/projects/1">MSMITH</ns4:projectManager>
        <ns4:parentProjectSpec
xmlns:ns4="http://serena.com/dimensions/projects/1">QLARIUS:MAINLINE_JAVA
</ns4:parentProjectSpec>
      </ns2:result>
    </ns2:listProjectsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

listScheduleJobs

Description

The `listScheduleJobs` Web service provides a method to list projects/streams in Dimensions CM.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
jobName	xsd:string	Y	Specifies a job name.
fromTime	xsd:dateTime	Y	Displays schedule jobs whose start time is greater than, or equal to, the value that is specified.
toTime	xsd:dateTime	Y	Displays schedule jobs whose start time is less than, or equal to, the value that is specified.
status	basic:JobStatusArray	Y	Filters job statuses and only displays the statuses that are specified. Can be one or more of the following: ■ Unknown ■ Active ■ Inactive ■ Running ■ Cancelling ■ Cancelled ■ Finished.
originator	xsd:string	Y	Displays jobs created by the user.
orderName	basic:JobOrderName	Y	Specifies the way that job lists are ordered, can be one or more of the following values: ■ Name: jobs are ordered by name (default). ■ Date: jobs are ordered by scheduled date/time (start time). ■ Status: jobs are ordered by current status.

Argument	Type	Optional?	Description
orderType	basic:JobOrderType		Specifies Ascending or Descending ordering. Can be one of the following values: ■ ASC. ■ DESC.
history	xsd:boolean	Y	Displays the execution histories of jobs.
commands	xsd:boolean	Y	Displays all the commands related to the scheduled job(s).

Response

Argument	Type	Optional?	Description
scheduleJob	basic:ScheduleJob	Y	Specifies an array of scheduled jobs.

Usage

`listScheduleJobs` Web service provides a method to get a list of scheduled jobs in Dimensions CM. The result for this Web service is an array, of *scheduleJobs* that represents such details:

Argument	Type	Description
jobUid	xsd:long	Specifies an array of scheduled jobs.
jobName	xsd:string	Job name.
startTime	xsd:dateTime	Starting time for this job.
createTime	xsd:dateTime	Creation time of this job.
updateTime	xsd:dateTime	Updating time of this job.
status	basic:JobStatus	Current status.
originator	xsd:string	User who created this job.
repeat	basic:JobRepeat	Repeat time for this job.
description	xsd:string	Description of this job.
relatedCommands	basic:RelatedScheduleJobCommandArray	All related commands to this job.
reletedHistories	basic:RelatedScheduleJobHistoryArray	All related histories to this job.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:listScheduleJobs>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:fromTime>2010-01-01T00:00:00.000+02:00</dmw:fromTime>
      <dmw:toTime>2016-01-01T00:00:00.000+02:00</dmw:toTime>
      <dmw:status>
        <ns:status>Inactive</ns:status>
      </dmw:status>
      <dmw:originator>DMSYS2</dmw:originator>
      <dmw:sorting>
        <ns:sorting>
          <ns:orderName>Date</ns:orderName>
          <ns:orderType>DESC</ns:orderType>
        </ns:sorting>
      </dmw:sorting>
    </dmw:listScheduleJobs>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:listScheduleJobsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:scheduleJob>
        <ns1:jobId
xmlns:ns1="http://serena.com/dimensions/basic/1">4216589</ns1:jobId>
        <ns1:jobName
xmlns:ns1="http://serena.com/dimensions/basic/1">MyJobName</ns1:jobName>
        <ns1:startTime xmlns:ns1="http://serena.com/dimensions/basic/1">2015-12-
    31T21:59:59.000+02:00</ns1:startTime>
        <ns1:createTime xmlns:ns1="http://serena.com/dimensions/basic/1">2009-12-
    21T14:25:37.000+02:00</ns1:createTime>
        <ns1:updateTime xmlns:ns1="http://serena.com/dimensions/basic/1">2009-12-
    21T14:25:37.000+02:00</ns1:updateTime>
        <ns1:status
xmlns:ns1="http://serena.com/dimensions/basic/1">Inactive</ns1:status>
```

```
        <ns1:originator
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS2</ns1:originator>       <repeat
    xmlns="http://serena.com/dimensions/basic/1">
          <repeatTime>30</repeatTime>
          <repeatType>Minutes</repeatType>
        </repeat>
        <ns1:description
xmlns:ns1="http://serena.com/dimensions/basic/1">Schedule job description</
    ns1:description>
      </ns2:scheduleJob>
      <ns2:scheduleJob>
        <ns1:jobId
xmlns:ns1="http://serena.com/dimensions/basic/1">4216802</ns1:jobId>
        <ns1:jobName
xmlns:ns1="http://serena.com/dimensions/basic/1">JobName#3</ns1:jobName>
        <ns1:startTime xmlns:ns1="http://serena.com/dimensions/basic/1">2014-08-
    14T21:59:59.000+03:00</ns1:startTime>
        <ns1:createTime xmlns:ns1="http://serena.com/dimensions/basic/1">2010-02-
    03T12:45:45.000+02:00</ns1:createTime>
        <ns1:updateTime xmlns:ns1="http://serena.com/dimensions/basic/1">2010-02-
    03T12:45:45.000+02:00</ns1:updateTime>
        <ns1:status
xmlns:ns1="http://serena.com/dimensions/basic/1">Inactive</ns1:status>
           <ns1:originator
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS2</ns1:originator>
        <repeat xmlns="http://serena.com/dimensions/basic/1">
          <repeatTime>10</repeatTime>
          <repeatType>Minutes</repeatType>
        </repeat>
<ns1:description
xmlns:ns1="http://serena.com/dimensions/basic/1">description</ns1:description>
      </ns2:scheduleJob>
      <ns2:scheduleJob>
        <ns1:jobId
xmlns:ns1="http://serena.com/dimensions/basic/1">4216803</ns1:jobId>
        <ns1:jobName
xmlns:ns1="http://serena.com/dimensions/basic/1">JobName#4</ns1:jobName>

       <ns1:startTime xmlns:ns1="http://serena.com/dimensions/basic/1">2014-08-
    14T21:59:59.000+03:00</ns1:startTime>
        <ns1:createTime xmlns:ns1="http://serena.com/dimensions/basic/1">2010-02-
    03T12:45:45.000+02:00</ns1:createTime>
<ns1:updateTime
xmlns:ns1="http://serena.com/dimensions/basic/1">2010-02-03T12:45:45.000+02:00</
    ns1:updateTime>
        <ns1:status xmlns:ns1="http://serena.com/dimensions/basic/1">Inactive</
    ns1:status>
        <ns1:originator
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS2</ns1:originator>
        <repeat xmlns="http://serena.com/dimensions/basic/1">
          <repeatTime>10</repeatTime>
          <repeatType>Minutes</repeatType>
        </repeat>
```

```
        <ns1:description
xmlns:ns1="http://serena.com/dimensions/basic/1">description</ns1:description>
      </ns2:scheduleJob>
    </ns2:listScheduleJobsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

listWorkAreas

Description

The cistWorkAreas web service retrieves all work areas, or a single work area by name.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	Connection details for the Dimensions server.
filter	areas:WorkAreaFilter	Y	Options for filtering the returned list. Currently only filtering by name is supported.

Response

Argument	Type	Optional?	Description
workArea	areas:WorkArea	N	Each workArea element represents the details of a retrieved work area.

Usage

`listWorkAreas` provides a method to retrieve existing work areas. If a failure occurs, a SOAP fault is returned. This fault contains the `faultCode` and `faultString`. The `faultCode` identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The `faultString` identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:listWorkAreas>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:filter>
            <ns1:name>sample_workarea</ns1:name>
         </dmw:filter>
      </dmw:listWorkAreas>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:listWorkAreasResponse xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:workArea xsi:type="ns2:WorkArea"
xmlns:ns2="http://serena.com/dimensions/areas/1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <ns2:name>SAMPLE_WORKAREA</ns2:name>
            <ns2:networkNode>LOCALHOST</ns2:networkNode>
            <ns2:directory>c:\sample_workarea\</ns2:directory>
            <ns2:created>
               <ns1:when xmlns:ns1="http://serena.com/dimensions/basic/1">2012-
09-28T10:13:00.000+01:00</ns1:when>
               <ns1:by
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS</ns1:by>
            </ns2:created>
            <ns2:updated>
               <ns1:when xmlns:ns1="http://serena.com/dimensions/basic/1">2012-
09-28T10:52:20.000+01:00</ns1:when>
               <ns1:by
xmlns:ns1="http://serena.com/dimensions/basic/1">DMSYS</ns1:by>
            </ns2:updated>
            <ns2:owner>DMSYS</ns2:owner>
         </ns4:workArea>
      </ns4:listWorkAreasResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

moveItemToPart

Description

The movItemToPart Web service provides a method to move a Dimensions CM item to a part using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
itemSpec	xsd:string	N	Specifies the item that will be moved to a part.
fileName	xsd:string	Y	Specifies the name of the library file name.
partSpec	xsd:string	N	Specifies the part to which the item will be moved.
projectSpec	xsd:string	Y	This optionally specifies the project/stream to be used for this command; failing this, the user's current project/stream will be taken. Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project/stream.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

movItemToPart Web service provides a method to move a Dimensions CM item to a part.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:moveItemToPart>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:itemSpec>QLARIUS:SAMPLEXMLPARSER JAVA.A-SRC;1.0</dmw:itemSpec>
       <dmw:fileName>qlarius\utilities\SampleXMLParser.java</dmw:fileName>
      <dmw:partSpec>QLARIUS:IT.A;1</dmw:partSpec>
       <dmw:projectSpec>QLARIUS:UW_JAVA_2.0</dmw:projectSpec>
    </dmw:moveItemToPart>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:moveItemToPartResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>true</ns2:result>
    </ns2:moveItemToPartResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

promoteBaselines

Description

The `promoteBaselines` Web service provides a method to promote one or more Dimensions CM baselines using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baseline	basic: BaselineSpecArray	N	The list of baseline specifications that will be processed by this command.
projectSpec	xsd:string	N	The name of the project the baseline or baselines are in.
comment	xsd:string	Y	Comment text associated with the promotion operation.
stage	xsd:string	Y	Specifies the target stage to promote the object to. This must be a stage from the global stage lifecycle, and there must be a transition from the current stage to the new stage in the stage lifecycle in order for the command to succeed
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
deploy	xsd: boolean	Y	Cannot be combined with `<areas>` and indicates that `<areas>` is not included on purpose as no deployment is to occur.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`promoteBaselines` Web service provides a method to promote one or more `Dimensions CM baselines`.The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:promoteBaselines>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselines>
       <ns:baselineSpec>QLARIUS:STREAM_A_BASELINE_1.0</ns:baselineSpec>
      </dmw:baselines>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promote Baseline</dmw:comment>
      <dmw:stage>QA</dmw:stage>
      <dmw:areas>
        <ns:area>QAAREA1</ns:area>
      </dmw:areas>
      <dmw:deploy>true</dmw:deploy>
      <dmw:deployStartTime>2010-10-29T11:25:00+03:00</dmw:deployStartTime>
    </dmw:promoteBaselines>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:promoteBaselinesResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: File 'Qlarius Underwriter/Qlarius Underwriter.ecl;1' was promoted
          File 'Qlarius Underwriter/build-user.xml;1' was promoted
          File 'Qlarius Underwriter/.classpath;1' was promoted
          File 'Qlarius Underwriter/qlarius/utilities/DBIO.java;1' was promoted
          File 'Qlarius Underwriter/qlarius/utilities/FileIO.java;1' was promoted
          File 'Qlarius Underwriter/.project;1' was promoted
          File 'Qlarius Underwriter/qlarius/utilities/PendingQuote.java;1' was promoted
          File 'Qlarius Underwriter/build.xml;1' was promoted
          File 'Qlarius Underwriter/qlarius/sampledata/sample.xml;1' was promoted
          File 'Qlarius Underwriter/qlarius/utilities/SampleXMLParser.java;1' was
    promoted
          File 'Qlarius Underwriter/qlarius/interfaces/AutoQuote.java;1' was promoted
          File 'Qlarius Underwriter/qlarius/interfaces/HealthQuote.java;1' was promoted
          File 'Qlarius Underwriter/qlarius/interfaces/LifeQuote.java;1' was promoted
          File 'Documents/dmcm_introduction.doc;1' was promoted
          14 file(s) were processed
          Schedule Job DEPLOYMENT_JOB_4217270 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:promoteBaselinesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

promoteItems

Description

The `promoteItems` Web service provides a method to promote one or more Dimensions CM items using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
items	basic:ItemSpecArray	Y	The list of item specifications that will be processed by this command. This will be used in preference to fileName in the command parameters if specified.
files	basic:FileNameArray	Y	List of items specified through path to items.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
comment	xsd:string	Y	Comment text associated with the promotion operation.
stage	xsd:string	Y	Specifies the target stage to promote the object to. This must be a stage from the global stage lifecycle.
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
deploy	xsd:boolean	Y	Cannot be combined with `<areas>` and indicates that `<areas>` is not included on purpose as no deployment is to occur. By default is true.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`promoteItems` Web service provides a method to promote one or more `Dimensions CM items`.The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/
    dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:promoteItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:items>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1</ns:itemSpec>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1</ns:itemSpec>
      </dmw:items>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promoting items</dmw:comment>
      <dmw:stage>QA</dmw:stage>
      <dmw:areas>
        <ns:area>QAAREA1</ns:area>
      </dmw:areas>
```

```
      <dmw:deploy>true</dmw:deploy>
      <dmw:deployStartTime>2010-10-22T14:12:00+03:00</dmw:deployStartTime>
    </dmw:promoteItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:promoteItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Promoting "QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1"...
          Promoting "QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1"...
          Schedule Job DEPLOYMENT_JOB_4218093 has been succesfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:promoteItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

promoteRequests

Description

The `promoteRequests` Web service provides a method to promote one or more Dimensions CM requests using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestIds	basic:RequestIdArray	N	The list of request ids that will be processed by this command.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
comment	xsd:string	Y	Comment text associated with the promotion operation.
stage	xsd:string	Y	Specifies the target stage to promote the object to. This must be a stage from the global stage lifecycle.
areas	basic:AreaArray	Y	A list of target deployment areas to deploy to.
recursive	xsd:boolean	Y	Whether child requests should be promoted as well
deploy	xsd:boolean	Y	Cannot be combined with `<areas>` and indicates that `<areas>` is not included on purpose as no deployment is to occur. By default is true.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`promoteRequests` Web service provides a method to promote one or more `Dimensions CM requests`.The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2"
    xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:promoteRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requests>
        <ns:requestId>QLARIUS_CR_13</ns:requestId>
        <ns:requestId>QLARIUS_TASK_6</ns:requestId>
      </dmw:requests>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:comment>Promoting requests</dmw:comment>
      <dmw:stage>QA</dmw:stage>
      <dmw:areas>
        <ns:area>QAAREA1</ns:area>
      </dmw:areas>
```

```
 <ns:area>QAAREA1</ns:area>
      </dmw:areas>
      <dmw:recursive>false</dmw:recursive>
      <dmw:deployStartTime>2010-10-21T15:37:00+03:00</dmw:deployStartTime>
    </dmw:promoteRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:promoteRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Promoting "QLARIUS_CR_13"...
          Promoting "QLARIUS_TASK_6"...
          Schedule Job DEPLOYMENT_JOB_4217809 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:promoteRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

relateItemsToParts

Description

The `relateItemsToParts` Web service provides a method to relate Dimensions CM items to parts in a specified project/stream using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
itemSpecs	basic:ItemSpecArray	N	Specifies the array of items to be related to parts.
partSpecs	basic:PartSpecArray	N	Specifies the array of parts to which items are to be related.
projectSpec	xsd:string	Y	This optionally specifies the project/stream to be used for this command; failing this, the user's current project/stream will be taken. Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project/stream.
continueOnError	xsd:boolean	Y	■ If the value of `continueOnError` is false: If one of the commands fails, the Web service does not continue with the execution of the subsequent commands and returns an error. ■ If the value of `continueOnError` is true: If one of the commands fails, the Web service continues to try to execute the subsequent commands and returns errors for each of the failing commands.

Response

Argument	Type	Optional?	Description
results	impl:DimensionsResultArray	N	Array of results of each single operation.

Usage

relateItemsToParts Web service provides a method to relate Dimensions CM items to parts in a specified project/stream. The result for this Web service is represented as an array.

Each element of the array contains detailed information of each single operation of this Web service and has constituents such as follow:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:relateItemsToParts>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:itemSpecs>
        <ns:itemSpec>QLARIUS:SAMPLE XML.A-SRC;1.0</ns:itemSpec>
        <ns:itemSpec>INCORRECT_ITEM_SPEC</ns:itemSpec>
       </dmw:itemSpecs>
       <dmw:partSpecs>
       <ns:partSpec>QLARIUS:QUOTATION.A;1</ns:partSpec>
        <ns:partSpec>INCORRECT_PART_SPEC</ns:partSpec>
      </dmw:partSpecs>
        <ns:partSpec>QLARIUS:QUOTATION.A;1</ns:partSpec>
        <ns:partSpec>INCORRECT_PART_SPEC</ns:partSpec>
      </dmw:partSpecs>
      <dmw:projectSpec>QLARIUS:UW_JAVA_2.0</dmw:projectSpec>
```

```
      <dmw:continueOnError>true</dmw:continueOnError>
    </dmw:relateItemsToParts>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>One or more operations have failed! See additional details in
    response.</faultstring>
      <detail>
        <ns2:DimensionsBatchFault xmlns:ns2="http://serena.com/dmwebservices2">
          <ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
             <ns2:description>Error: The specified item revision is not in the project
    $GENERIC:$GLOBAL COR0005314E Error: Item type NULL does not exist</ns2:description>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Design Part was not found!</ns2:description>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Design Part was not found!</ns2:description>
            </ns2:result>
          </ns2:result>
        </ns2:DimensionsBatchFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

relateRequestToRequests

Description

The `relateRequestsToRequests` Web service provides a method to relate Dimensions CM request to requests using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	This is the identity of the request that is the parent in the relationship to be removed.
childRequestIds	basic:RequestIdArray	N	Specifies the identities of one or more requests to be children in the relationship to be created.
relationshipType	basic:RequestRequestValidRels	Y	Specifies the relationship class as `DEPENDENT` or `INFO`, or as the name of one of the subclasses of relationship defined as equivalent to either of these.
continueOnError	xsd:boolean	Y	■ If the value of `continueOnError` is false: If one of the commands fails, the Web service does not continue with the execution of the subsequent commands and returns an error. ■ If the value of `continueOnError` is true: If one of the commands fails the Web service continues to try to execute the subsequent commands and returns errors for each of the failing commands.

Response

Argument	Type	Optional?	Description
results	impl:DimensionsResultArray	N	Array of results of each single operation

Usage

relateRequestToRequests Web service provides a method to relate a Dimensions CM request to requests. The result for this Web service is an array, each element of which contains detailed information on each single operation of the Web service:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:relateRequestToRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_CR_48</dmw:requestId>
      <dmw:childRequestIds>
        <ns:requestId>QLARIUS_CR_30</ns:requestId>
        <ns:requestId>QLARIUS_CR_31</ns:requestId>
        <ns:requestId>QLARIUS_CR_32</ns:requestId>
        <ns:requestId>QLARIUS_CR_33</ns:requestId>
      </dmw:childRequestIds>
      <dmw:continueOnError>true</dmw:continueOnError>
    </dmw:relateRequestToRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:relateRequestToRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:results>
        <ns2:result>
          <ns2:result>true</ns2:result>
        </ns2:result>
        <ns2:result>
          <ns2:result>true</ns2:result>
        </ns2:result>
        <ns2:result>
          <ns2:result>true</ns2:result>
        </ns2:result>
        <ns2:result>
          <ns2:result>true</ns2:result>
        </ns2:result>
      </ns2:results>
    </ns2:relateRequestToRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

removeDeploymentArea

Description

The removeDeploymentArea web service retrieves a single work area by name.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	Connection details for the Dimensions server.
name	xsd:string	N	Name of the deployment area to remove.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

removeDeploymentArea provides a method to remove an existing deployment area. The area cannot be removed if it is in use. If a failure occurs, a SOAP fault is returned. This fault contains the faultCode and faultString. The faultCode identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The faultString identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:removeDeploymentArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>Secret001</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>aidy-win2008</ns:server>
         </dmw:connectionDetails>
         <dmw:name>sample_deploymentarea</dmw:name>
      </dmw:removeDeploymentArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:removeDeploymentAreaResponse
xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:removeDeploymentAreaResponse>
   </soapenv:Body>
</soapenv:
```

removeWorkArea

Description

The removeWorkArea web service retrieves a single work area by name.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	Connection details for the Dimensions server.
name	xsd:string	N	Name of the work area to remove.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

removeWorkArea provides a method to remove an existing work area. The area cannot be removed if it is in use. If a failure occurs, a SOAP fault is returned. This fault contains the faultCode and faultString. The faultCode identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The faultString identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:removeWorkArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:name>sample_workarea</dmw:name>
      </dmw:removeWorkArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:removeWorkAreaResponse xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:removeWorkAreaResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

runCommand

Description

The `runCommand` Web Service provides the capability of running any Dimensions CM command-line command.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
command	xsd:string	Y	The command must be quoted and escaped as necessary. **NOTE** Not all commands can currently be run: for example, the commands '`upload`' and '`download`' cannot be used with this Web service.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of the execution specified in the command-line command.

Usage

runCommand Web service provides the capability of running any Dimensions CM command-line command.

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful, otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used.
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).

- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:runCommand>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:command>
        CC "QLARIUS" "CR"/ATTRIBUTES=(Title="RequestFromWS#2")  /
    WORKSET="QLARIUS:STREAM_A"
      </dmw:command>
    </dmw:runCommand>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:runCommandResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: The request QLARIUS_CR_16 has been forwarded to CHRIS, CLEM, DMSYS,
    JOE, RICK, RON

          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:runCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitDeployBaselines

Description

The `submitDeployBaselines` Web service submits deploying Dimensions CM baselines using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselines	basic:BaselineSpecArray	N	The list of baselines that will be processed by this command.
projectSpec	xsd:string	N	The name of the project the baseline or baselines are in.
areas	basic:AreaArray	N	A list of target deployment areas to deploy to.
comment	xsd:string	Y	Comment text associated with the deploying operation.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`submitDeployBaselines` Web service provides a method to submit deploying Dimensions CM baselines to deploy operations. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitDeployBaselines>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselines>
        <ns:baselineSpec>QLARIUS:STREAM_A_BASELINE_1.0</ns:baselineSpec>
      </dmw:baselines>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:comment>Deploying baseline</dmw:comment>
      <dmw:deployStartTime>2010-10-29T11:36:00+03:00</dmw:deployStartTime>
</dmw:submitDeployBaselines>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitDeployBaselinesResponse
xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitDeployBaselinesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitDeployItems

Description

The `submitDeployItems` Web service submits Dimensions CM items to deploy using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
items	basic:ItemSpecArray	Y	Array of item specifications that will be submitted to deploy.
files	basic:FileNameArray	Y	List of items specified through path to items.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
areas	basic:AreaArray	N	A list of target deployment areas to deploy to.
comment	xsd:string	Y	Comment text associated with the deploying operation.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResultArray	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`submitDeployItems` Web service provides a method to submit deploying Dimensions CM items to specified areas. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitDeployItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:items>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1</ns:itemSpec>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1</ns:itemSpec>
      </dmw:items>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:comment>Deploying</dmw:comment>
      <dmw:deployStartTime>2010-10-22T14:31:00+03:00</dmw:deployStartTime>
    </dmw:submitDeployItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitDeployItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Submitting "QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1"...
          Submitting "QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1"...
          Schedule Job DEPLOYMENT_JOB_4218168 has been succesfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitDeployItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitDeployRequests

Description

The `submitDeployRequests` Web service submits deploying Dimensions CM requests using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requests	basic:RequestIdArray	N	The list of request ids that will be processed by this command.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
areas	basic:AreaArray	N	A list of target deployment areas to deploy to.
comment	xsd:string	Y	Comment text associated with the deploying operation.
recursive	xsd:boolean	Y	Whether child requests should be deployed as well
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin
returnSubmittedDeploymentJobs	xsd:boolean	Y	When true, returns submitted deployment jobs along with the result.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.
deploymentJobs	basic:SubmittedDeploymentJobArray	Y	List of deployment job attributes in the form of deployment job UID/job type pairs.

Usage

`submitDeployRequests` Web service provides a method to submit deploying Dimensions CM requests. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/dimensions/
    basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitDeployRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requests>
        <ns:requestId>QLARIUS_CR_13</ns:requestId>
        <ns:requestId>QLARIUS_TASK_6</ns:requestId>
      </dmw:requests>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
        <ns:area>DEVAREA2</ns:area>
      </dmw:areas>
      <dmw:comment>Deploying with scheduled time</dmw:comment>
      <dmw:recursive>false</dmw:recursive>
      <dmw:deployStartTime>2010-10-22T11:24:00+03:00</dmw:deployStartTime>
    </dmw:submitDeployRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitDeployRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Submitting "QLARIUS_CR_13"...
          Submitting "QLARIUS_TASK_6"...
          Schedule Job DEPLOYMENT_JOB_4217949 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitDeployRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitRollbackBaselines

Description

The `submitRollbackBaselines` Web service submits one or more Dimensions CM baselines to roll back using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
baselines	basic:BaselineSpecArray	N	The list of baselines that will be processed by this command.
projectSpec	xsd:string	N	The name of the project the baseline or baselines are in.
areas	basic:AreaArray	N	A list of target deployment areas to roll back from.
comment	xsd:string	Y	Comment text associated with the rollback operation.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`submitRollbackBaselines` Web service provides a method to submits one or more Dimensions CM baselines to roll back. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/
    dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitRollbackBaselines>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:baselines>
        <ns:baselineSpec>QLARIUS:STREAM_A_BASELINE_1.0</ns:baselineSpec>
      </dmw:baselines>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:comment>deploying baseline</dmw:comment>
      <dmw:deployStartTime>2010-10-29T11:40:00+03:00</dmw:deployStartTime>>
    </dmw:submitRollbackBaselines>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitRollbackBaselinesResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Schedule Job DEPLOYMENT_JOB_4217363 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitRollbackBaselinesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitRollbackItems

Description

The `submitRollbackItems` Web service submits one or more Dimensions CM items to roll back from specified areas using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
items	basic:ItemSpecArray	Y	Array of item specifications that will be submitted to rollback.
files	basic:FileNameArray	Y	List of items specified through path to items.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
areas	basic:AreaArray	N	A list of target deployment areas to roll back from.
comment	xsd:string	Y	Comment text associated with the submit rollback operation.
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin.

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResultArray	N	Result of operation.

Usage

`submitRollbackItems` Web service provides a method to submits one or more Dimensions CM items to roll back from specified areas. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/
    dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitRollbackItems>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:items>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1</ns:itemSpec>
        <ns:itemSpec>QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1</ns:itemSpec>
      </dmw:items>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
      </dmw:areas>
      <dmw:comment>Deploying</dmw:comment>
      <dmw:deployStartTime>2010-10-22T14:36:00+03:00</dmw:deployStartTime>
    </dmw:submitRollbackItems>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitRollbackItemsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Rolling back "QLARIUS:SCHEDULEDITEM1-4218069.A-DAT;stream_a#1"...
          Rolling back "QLARIUS:SCHEDULEDITEM2-4218066.A-DAT;stream_a#1"...
          Schedule Job DEPLOYMENT_JOB_4218191 has been succesfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitRollbackItemsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

submitRollbackRequests

Description

The `submitRollbackRequests` Web service submits one or more Dimensions CM request to roll back from specified areas using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requests	basic:RequestIdArray	N	The list of request ids that will be processed by this command.
projectSpec	xsd:string	N	Specifies the project to be used for this command.
areas	basic:AreaArray	N	A list of target deployment areas to roll back from.
comment	xsd:string	Y	Comment text associated with the rollback operation.
recursive	xsd:boolean	Y	Whether child requests should be rolled back as well
deployStartTime	xsd:dateTime	Y	Start time for the file deployment operation to begin

Response

Argument	Type	Optional?	Description
result	impl:DimensionsResult	N	Result of operation.

Usage

`submitRollbackRequests` Web service provides a method to submits one or more Dimensions CM requests to roll back from specified areas. The result returned by this Web service is a complex type having four elements as follows:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:submitRollbackRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requests>
        <ns:requestId>QLARIUS_CR_13</ns:requestId>
        <ns:requestId>QLARIUS_TASK_6</ns:requestId>
      </dmw:requests>
      <dmw:projectSpec>QLARIUS:STREAM_A</dmw:projectSpec>
      <dmw:areas>
        <ns:area>DEVAREA1</ns:area>
        <ns:area>DEVAREA2</ns:area>
      </dmw:areas>
      <dmw:comment>Rollback with scheduled time</dmw:comment>
      <dmw:recursive>false</dmw:recursive>
      <dmw:deployStartTime>2010-10-22T11:26:00+03:00</dmw:deployStartTime>
    </dmw:submitRollbackRequests>
  </soapenv:Body>
 </soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:submitRollbackRequestsResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:result>
        <ns2:result>true</ns2:result>
        <ns2:description>
          SUCCESS: Rolling back "QLARIUS_CR_13"...
          Rolling back "QLARIUS_TASK_6"...
          Schedule Job DEPLOYMENT_JOB_4217976 has been successfully submitted
          Operation completed
          Operation completed
        </ns2:description>
      </ns2:result>
    </ns2:submitRollbackRequestsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

unrelateItemsFromParts

Description

The unrelateItemsFromParts Web service provides a method to unrelate Dimensions CM items from parts in a specified project/stream using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
itemSpecs	basic:ItemSpecArray	N	Specifies the array of items to be unrelated from parts.
partSpecs	basic:PartSpecArray	N	Specifies the array of parts to which items are to be unrelated.
projectSpec	xsd:string	Y	This optionally specifies the project/stream to be used for this command; failing this, the user's current project/stream will be taken. Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project/stream.
continueOnError	xsd:boolean	Y	■ If the value of `continueOnError` is false: If one of the commands fails, the Web service does not continue with the execution of the subsequent commands and returns an error. ■ If the value of `continueOnError` is true: If one of the commands fails the Web service continues to try to execute the subsequent commands and returns errors for each of the failing commands.

Response

Argument	Type	Optional?	Description
results	impl:DimensionsResultArray	N	Array of results of each single operation.

Usage

unrelateItemsFromParts Web service provides a method to relate Dimensions CM items to parts in a specified project/stream. The result for this Web service is an array, each element of which contains detailed information on each single operation of the Web service:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:relateItemsToParts>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:itemSpecs>
        <ns:itemSpec>QLARIUS:SAMPLE XML.A-SRC;1.0</ns:itemSpec>
        <ns:itemSpec>INCORRECT_ITEM_SPEC</ns:itemSpec>
      </dmw:itemSpecs>
      <dmw:partSpecs>
        <ns:partSpec>QLARIUS:QUOTATION.A;1</ns:partSpec>
        <ns:partSpec>INCORRECT_PART_SPEC</ns:partSpec>
      </dmw:partSpecs>
       <dmw:projectSpec>QLARIUS:UW_JAVA_2.0</dmw:projectSpec>
      <dmw:continueOnError>true</dmw:continueOnError>
    </dmw:relateItemsToParts>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>One or more operations have failed! See additional details in
    response.</faultstring>
      <detail>
        <ns2:DimensionsBatchFault xmlns:ns2="http://serena.com/dmwebservices2">
          <ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Error: The specified item revision is not in the project
    $GENERIC:$GLOBAL COR0005314E Error: Item type NULL does not exist</ns2:description>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Design Part was not found!</ns2:description>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Design Part was not found!</ns2:description>
            </ns2:result>
          </ns2:result>
        </ns2:DimensionsBatchFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

unrelateRequestFromRequests

Description

The unrelateRequestFromRequests Web service provides a method to unrelate a Dimensions CM request from requests using the data supplied.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	This is the identity of the request that is the parent in the relationship to be removed.
childRequestIds	basic:RequestIdArray	N	Specifies the identities of one or more requests to be children in the relationship to be unrelated.
relationshipType	basic:RequestRequestValidRels	Y	Specifies the relationship class as `DEPENDENT` or `INFO`, or as the name of one of the subclasses of relationship defined as equivalent to either of these.
continueOnError	xsd:boolean	Y	■ If the value of continueOnError is false: If one of the commands fails the Web service does not continue with the execution of the subsequent commands and returns an error. ■ if the value of continueOnError is true: If one of the commands fails the Web service continues to try to execute the subsequent commands and returns errors for each of the failing commands.

Response

Argument	Type	Optional?	Description
results	impl:DimensionsResultArray	N	Array of results of each single operation.

Usage

unrelateRequestFromRequests Web service provides a method to unrelate a Dimensions CM request from requests. The result for this Web service is an array, each element of which contains detailed information on each single operation of the Web service:

Argument	Type	Optional?	Description
result	xsd:boolean	N	True if operation completed successful; otherwise false.
objectSpec	xsd:string	Y	Not used.
url	xsd:string	Y	Not used,
description	xsd:string	Y	Contains description.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:unrelateRequestFromRequests>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_CR_48</dmw:requestId>
      <dmw:childRequestIds>
        <ns:requestId>QLARIUS_CR_30</ns:requestId>
        <ns:requestId>QLARIUS_CR_31</ns:requestId>
        <ns:requestId>QLARIUS_CR_32</ns:requestId>
        <ns:requestId>QLARIUS_CR_333</ns:requestId>
      </dmw:childRequestIds>
      <dmw:continueOnError>true</dmw:continueOnError>
    </dmw:unrelateRequestFromRequests>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>One or more operations have failed! See additional details in
    response.</faultstring>
      <detail>
        <ns2:DimensionsBatchFault xmlns:ns2="http://serena.com/dmwebservices2">
          <ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>true</ns2:result>
            </ns2:result>
            <ns2:result>
              <ns2:result>false</ns2:result>
              <ns2:description>Request was not found!</ns2:description>
            </ns2:result>
          </ns2:result>
        </ns2:DimensionsBatchFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

updateDeploymentArea

Description

The updateDeploymentArea web service updates the details of an existing deployment area.

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	Connection details for the Dimensions server.
name	xsd:string	N	Name of the area to be updated.
newName	xsd:string	Y	New name of the area.
networkNode	xsd:string	Y	Machine hosting the area. This cannot be un-set.
directory	xsd:string	Y	Directory or partitioned data set where the area is located. This cannot be un-set.
detailedDescription	xsd:string	Y	Description of the deployment area.
user	areas:User	Y	Login information for the OS user account or credential set that will own files transferred into the area. This may be a userid and password combination, or the ID of a credential set held by the server.
fetchExpanded	xsd:boolean	Y	Whether header substitution variables will be expanded with files are fetched to the area. If this is not specified, it defaults to true.
owner	xsd:string	Y	User or group that will become the owner of the area. If this is not specified, the user who created the area is set as the owner. The owner has the right to manage the area definition.
stage	xsd:String	Y	Stage with which the new area is associated. This cannot be un-set.
transferScripts	basic:TransferScripts	Y	Specifies the transfer script set of pre / post / fail scripts.
scriptParameters	basic:ScriptParameters	Y	Set of named script parameters to be passed to scripts. Each parameter may be single or multi-valued.

Argument	Type	Optional?	Description
LibraryCacheArea	xsd:string	Y	Library cache area that is defined by the CLCA (Create Library Cache Area) command. When executing a command that gets files, Dimensions checks whether the library cache area associated with the project or stream already contains a coy of the files. If so, Dimensions copies the files from the cache to the area rather than from the item library, improving performance in certain cases.
status	areas:Status	Y	Status of the area. If the status is ONLINE, the area may participate in file transfer options. If the status is OFFLINE, the area is excluded from any file transfers. If this is not specified, an area with the status ONLINE is created.
filter	xsd:string	Y	Name of the area filter to use when deploying to this area.

Response

Argument	Type	Optional?	Description
result	xsd:boolean	N	Result of the operation.

Usage

udpateDeploymentArea provides a method to update the details of an existing deployment area. Where allowed, existing values can be un-set by providing an empty element. Arrays or lists completely replace the existing contents. If a failure occurs, a SOAP fault is returned. This fault contains the faultCode and faultString. The faultCode identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issue, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The faultString identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:updateDeploymentArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:deploymentArea>
            <ns1:name>sample_deploymentarea</ns1:name>
            <ns1:stage>SIT</ns1:stage>
            <ns1:status>ONLINE</ns1:status>
         </dmw:deploymentArea>
      </dmw:updateDeploymentArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:updateDeploymentAreaResponse
xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:updateDeploymentAreaResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

updateRequest

Description

The updateRequest Web service provides a method to update a Dimensions CM request's attributes using the data supplied.

IMPORTANT! The `updateRequest` Web service is designed to update *only* the request's attributes.

To update certain of the other arguments, other Web services should be used, for example:

- `status` - `actionRequest`
- `relatedRequests` - `relatedRequestToRequests`

Arguments

Argument	Type	Optional?	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
request	requests:Request	N	Specifies the request attributes that will be updated.

Request* Structure			
Argument	**Type**	**Optional?**	**Description**
requestId	xsd:string	N	Specifies the type of request to be created.
status	xsd:string	Y	Specifies the product that is to own the new request.
attributes	basic:AttributeArray	Y	Specifies user defined attributes.
relatedParts	basic:RequestRelated PartArray	Y	Specifies the list of related parts to request.
relatedItems	basic:RequestRelatedItem Array	Y	Specifies the list of related items to request.
relatedRequests	basic:RequestRelated RequestArray	Y	Specifies the list of related requests to request.
relatedBaselines	basic:RequestRelated BaselineArray	Y	Specifies the list of related baselines to request.
related Requirements	basic:RequestRelated RequirementArray	Y	Specifies the list of related requirements to request.
relatedProjectSpec	xsd:string	Y	Specifies the list of related projects to request.
pendingUser	basic:PendingUserArray	Y	Specifies the list of users.

Request* Structure			
Argument	**Type**	**Optional?**	**Description**
archive	xsd:boolean	Y	Specifies whether it requires an archive or not.

Response

Argument	Type	Optional?	Description
requestId	xsd:string	N	Request Id of the updated request.

Usage

updateRequest Web service provides a method to update a Dimensions CM request's attributes.

If a failure occurs, a SOAP fault will be returned containing the `faultCode` and `faultString`:

- The `faultCode` identifies whether the fault occurred:
  - on the client (for example, because of an incorrectly defined service endpoint), or
  - on the server because of a SOAP issue or a Dimensions CM server issue (for example, because of an incorrectly named SOAP sub-element or an incorrectly named Dimensions CM product).
- The `faultString` will identify the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/requests/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:updateRequest>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:request>
        <ns1:requestId>QLARIUS_CR_48</ns1:requestId>
        <ns1:attributes>
```

```
          <ns:attribute>
            <ns:name>Severity</ns:name>
            <ns:datatype>Char</ns:datatype>
            <ns:type>Single_Value</ns:type>
            <ns:value>3</ns:value>
          </ns:attribute>
         <ns:attribute>
            <ns:name>PLAN_FINISH</ns:name>
            <ns:datatype>Date</ns:datatype>
            <ns:type>Single_Value</ns:type>
            <ns:value>2005-11-07T15:31:25.000+03:00</ns:value>
          </ns:attribute>
          <ns:attribute>
            <ns:name>EST_DEV_EFFORT</ns:name>
            <ns:datatype>Number</ns:datatype>
            <ns:type>Single_Value</ns:type>
            <ns:value>24</ns:value>
          </ns:attribute>
        </ns1:attributes>
      </dmw:request>
    </dmw:updateRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:updateRequestResponse xmlns:ns2="http://serena.com/dmwebservices2">
      <ns2:requestId>QLARIUS_CR_48</ns2:requestId>
    </ns2:updateRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

updateRequestDescription

Description

The `updateRequestDescription` service updates the detailed description of the specified request.

Arguments

Argument	Type	Optional	Description
connectionDetails	basic:ConnectionDetails	Y	This structure specifies the connection details for the Dimensions CM server.
requestId	xsd:string	N	The RequestId for which a detailed description should be returned.
description	xsd:string	N	Contents of the detailed description.

Response

No response present.

Usage

`updateRequestDescription` provides a method to update a Dimensions CM request's detailed description. If a failure occurs a SOAP fault will be returned.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:dmw="http://serena.com/dmwebservices2" xmlns:ns="http://serena.com/dimensions/
    basic/1">
  <soapenv:Header/>
  <soapenv:Body>
    <dmw:updateRequestDescription>
      <dmw:connectionDetails>
        <ns:username>dmsys</ns:username>
        <ns:password>dmsys_test</ns:password>
        <ns:dbName>cm_typical</ns:dbName>
        <ns:dbConnection>dim10</ns:dbConnection>
        <ns:server>localhost</ns:server>
      </dmw:connectionDetails>
      <dmw:requestId>QLARIUS_CR_16</dmw:requestId>
      <dmw:description>
        The Qlarius UnderWriter system shall electronically captures application details
    with complex validation of entered data.
        And also show funny pcitures together with indication process.
      </dmw:description>
    </dmw:updateRequestDescription>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:updateRequestDescriptionResponse xmlns:ns2="http://serena.com/dmwebservices2"/>
  </soapenv:Body>
</soapenv:Envelope>
```

updateWorkArea

Description

The `updateWorkArea` service updates the details of an existing work area.

Arguments

Argument	Type	Optional?	Description
`connectionDetails`	`basic:ConnectionDetails`	Y	Connection details for the Dimensions server.
`name`	`xsd:string`	N	Name of the work area to be updated.
`newName`	`xsd:string`	Y	New name for the area.
`networkNode`	`xsd:string`	Y	Machine hosting the area.
`directory`	`xsd:string`	Y	Directory or partitioned data set where the area is located.
`detailedDescription`	`xsd:string`	Y	Description of the work area.
`user`	`areas:User`	Y	Login information for the OS user account or credential set that will own files transferred into the area. This may be a userid and password combination, or the ID of a credential set held by the server.
`fetchExpanded`	`xsd:boolean`	Y	Whether item header substitution variables will be expanded when files are fetched to the area. This defaults to true if it is not specified.
`owner`	`xsd:string`	Y	The user or group that will become the owner of the area. If not specified, the user who created the area is set as the owner. The owner has the right to manage the area definition.
`userList`	`basic:UserNameArray`	Y	List of users and / or groups that are granted the right to use the area. If this is empty, any user can set the area as a working location.

Response

Argument	Type	Optional?	Description
`result`	`xsd:boolean`	N	Result of the operation.

Usage

updateWorkArea provides a method to update the details of an existing work area. Existing values can be un-set where allowed by providing an empty element. Arrays or lists will completely replace the existing content. If a failure occurs, a SOAP fault is returned. This fault contains the faultCode and faultString. The faultCode identifies whether the fault occurs on the client (for example because of an incorrectly defined endpoint) or on the server (for example because of a SOAP or Dimensions server issues, such as an incorrectly named SOAP sub-element or an incorrectly named Dimensions product). The faultString identifies the actual fault.

Example SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dmw="http://serena.com/dmwebservices2"
xmlns:ns="http://serena.com/dimensions/basic/1"
xmlns:ns1="http://serena.com/dimensions/areas/1">
   <soapenv:Header/>
   <soapenv:Body>
      <dmw:updateWorkArea>
         <dmw:connectionDetails>
            <ns:username>dmsys</ns:username>
            <ns:password>dmsys_test</ns:password>
            <ns:dbName>cm_typical</ns:dbName>
            <ns:dbConnection>Dim12</ns:dbConnection>
            <ns:server>localhost</ns:server>
         </dmw:connectionDetails>
         <dmw:workArea>
            <ns1:name>sample_workarea</ns1:name>
            <!-- un-set the description -->
            <ns1:detailedDescription/>
            <ns1:userList>
               <ns:userName>BOBBY</ns:userName>
               <ns:userName>TED</ns:userName>
               <ns:userName>AMY</ns:userName>
            </ns1:userList>
         </dmw:workArea>
      </dmw:updateWorkArea>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns4:updateWorkAreaResponse xmlns:ns4="http://serena.com/dmwebservices2">
         <ns4:result>
            <ns4:result>true</ns4:result>
         </ns4:result>
      </ns4:updateWorkAreaResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Chapter 19

Application Lifecycle Framework Events Reference

Introduction to ALF Events	500
ALF Event Structure	500
Baseline ALF Events	502
Deployment ALF Events	508
Item ALF Events	519
Project ALF Events	528
Request ALF Events	536
Schedule Job ALF Events	545

Introduction to ALF Events

Serena® Dimensions® CM can generate Application Lifecycle Framework (ALF) events for a restricted set of operations. ALF is a former Eclipse project, in which Serena played a leading role, to develop an open integration and interoperability framework for application lifecycle management tools.

Each Dimensions CM operation fires at most a single ALF event and in accordance with ALF guidelines these events are generated post operation. This is different from the Dimensions CM events callout interface (see *Developer's Reference*), which may generate a number of specific events during the course of a single operation.

IMPORTANT! Deploy ALF events are not generated on creating or delivering files, because in those particular cases "Change Set" is deployed and ALF events only support Dimensions CM objects of type item, request, and baseline.

The data provided by each event is a reflection of the Dimensions CM command that has issued the ALF event. Therefore, the data provided by the various events may differ; not all events will provide the same amount of data to the event clients.

The ALF events are added to with each release of Dimensions CM. As of Dimensions CM 12.2.2 the ALF events available are as detailed in this chapter.

ALF Event Structure

ALF defines:

- A set of standard event types.
- The basic data for an event, defined as the `EventBaseType`.

The basic data includes:

- The type of event.
- Additional basic descriptive data, such as:
  - The tool and object that caused the event.
  - Some information about the context in which the event occurred, for example:
    - The user who was operating the tool, or
    - the credentials of a process acting on behalf of a user.

In addition to the basic event data, events can carry data elements specific to a given tool.

The following is a simple ALF event with filled-only base fields::

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <EventNotice xmlns="http://www.eclipse.org/alf/schema/EventBase/1">
      <EventNotice xmlns="">
        <ns1:Base xmlns:ns1="http://www.eclipse.org/alf/schema/EventBase/1">
          <ns1:EventId>7a83e069-ba78-46df-a957-ef89192af145</ns1:EventId>
          <ns1:Timestamp>2007-09-19T15:37:38.781Z</ns1:Timestamp>
          <ns1:EventType>Object Created</ns1:EventType>
          <ns1:ObjectType>MyObject</ns1:ObjectType>
          <ns1:ObjectId>123</ns1:ObjectId>
          <ns1:Source>
            <ns1:Product>MyProduct</ns1:Product>
            <ns1:ProductVersion>1.0</ns1:ProductVersion>
            <ns1:ProductInstance>MyProductInstance</ns1:ProductInstance>
          </ns1:Source>
          <ns1:User />
          <ns1:EventControl>
            <ns1:EmEventId></ns1:EmEventId>
            <ns1:EmTimestamp>2007-09-19T15:37:38.796Z</ns1:EmTimestamp>
            <ns1:PrecedingEmEventId></ns1:PrecedingEmEventId>
            <ns1:ApplicationName> </ns1:ApplicationName>
            <ns1:EventMatchName></ns1:EventMatchName>
            <ns1:ServiceFlowName></ns1:ServiceFlowName>
            <ns1:ServiceFlowId></ns1:ServiceFlowId>
            <ns1:Callback>false</ns1:Callback>
            <ns1:Environment> </ns1:Environment>
            <ns1:EmUser />
          </ns1:EventControl>
        </ns1:Base>
      </EventNotice>
    </EventNotice>
  </soapenv:Body>
</soapenv:Envelope>
```

The most important fields in the basic event data are:

Field	Description
<EventID>	A unique identifier for the event instance.
<Timestamp>	The date and timestamp (in XML dateTime format) when the event is formed.
<EventType>	A string indicating the type of event. EventType designates the "verb", that is, what action happened to the Objects that triggered the event. Dimensions CM uses event types like: check in, check out, action, and build submitted.
<ObjectType>	The type of entity involved,such as Project, Baseline, Item, and Request.
<ObjectID>	An identifier of the entity that changed within a tool. The identifier must be unique for that installation of the tool. The primary purpose is to allow subsequent ServiceFlows to uniquely identify (and perhaps access) the object that triggered the event. Dimensions CM uses object Id like "UW_JAVA_2.0" for project and "AUTOQUOTE JAVA" for item.

Field	Description	
<Source>	A container describing the source of the event:	
	<Product>	A descriptive name for the tool that emitted the event. Dimensions CM uses "DimensionsBaseline", "DimensionsItem", "DimensionsProject", or "DimensionsRequest".
	<ProductVersion>	The release version of the product.
	<ProductInstance>	A unique string identifying the instance of the tool. This is useful when there may be multiple instances of a product working with ALF. Dimensions CM uses the following format: DB_ID@SERVER-DB_CONNECTION For example: QLARIUS_CM@MYHOST0123-DIM10. **NOTE** The identifier must be in UPPER CASE.
	<ProductCallbackURI>	The web service endpoint for tools that support callbacks from ServiceFlows for additional information. The element content is optional. Dimensions CM always sets EventControl::Callback to false so this parameter should not be used.

Each Dimensions CM object (project/stream, baseline, item, request, etc) define its own extension data, which is described in the following sections of this chapter.

NOTE If Single Sign On (SSO) authentication is enabled for Dimensions CM, then all ALF events will contain an SSO token.

Baseline ALF Events

Event Extension for Baseline

All ALF events from a Dimensions CM baseline provide the DMBaselineExtensionType type in the ALF event extension, and each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

DMBaselineExtensionType contains:

Field	Description
Database	The database that originated the event, in the format DB_ID@SERVER-DB_CONNECTION.
ProjectSpec	The specification of the Dimensions CM project/stream context for the event, in the format ProductId:ProjectId.
User	The Dimensions CM user who originated the event.

Field	Description	
Baseline	uid	A Unique integer identifier for the Dimensions CM baseline (bln_catalogue.obj_uid).
	specification	The full specification of the baseline.
	type	The type of the baseline.
	description	The description of the baseline.
	attributes	An array of user defined attributes and values.
	relatedBaselines	An array of related baselines.
	relatedRequests	An array of related requests.
	buildJobId	Id of build job.
	buildJobUrl	URL to the build job status.
	buildJobStatus	Status of built project (Success, Failed, or Cancelled).

Build-Submitted and Build-Completed Baseline ALF Events

These events are fired on performing a Dimensions CM Build Baseline (BLDB) command.

All the fields of the baseline event extension for ALF events are filled.

The following is a sample of a baseline build-submitted event (a build-completed event looks the same, only <EventType> is different):

```
?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <EventNotice xmlns="http://www.eclipse.org/alf/schema/EventBase/1">
      <EventNotice xmlns="">
        <ns1:Base xmlns:ns1="http://www.eclipse.org/alf/schema/EventBase/1">
          <ns1:EventId>71dd0580-8a2b-11dd-9b1c-8ee10287e59a</ns1:EventId>
          <ns1:Timestamp>2008-09-24T11:25:08.952Z</ns1:Timestamp>
          <ns1:EventType>build-completed</ns1:EventType>
          <ns1:ObjectType>Baseline</ns1:ObjectType>
          <ns1:ObjectId>PRJ BL 01</ns1:ObjectId>
          <ns1:Source>
            <ns1:Product>Serena Dimensions CM</ns1:Product>
            <ns1:ProductVersion>2009</ns1:ProductVersion>
            <ns1:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns1:ProductInstance>
            <ns1:ProductCallbackURI>uri:dummy</ns1:ProductCallbackURI>
          </ns1:Source>
          <ns1:User>
            <ns1:LoginID>dmsys</ns1:LoginID>
            <ns1:Certificate></ns1:Certificate>
          </ns1:User>
          <ns1:EventControl>
            <ns1:EmEventId></ns1:EmEventId>
            <ns1:EmTimestamp>2008-09-24T11:25:08.952Z</ns1:EmTimestamp>
            <ns1:PrecedingEmEventId></ns1:PrecedingEmEventId>
            <ns1:ApplicationName></ns1:ApplicationName>
            <ns1:EventMatchName></ns1:EventMatchName>
            <ns1:ServiceFlowName></ns1:ServiceFlowName>
            <ns1:ServiceFlowId></ns1:ServiceFlowId>
            <ns1:Callback>false</ns1:Callback>
            <ns1:Environment></ns1:Environment>
            <ns1:EmUser/>
          </ns1:EventControl>
        </ns1:Base>
```

```
        <ns2:Extension xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
          <ns3:Database
xmlns:ns3="http://serena.com/dimensions/ALF/DMEventExtentions/1">QLARIUS_CM@UA003679-
DIM10</ns3:Database>
          <ns4:ProjectSpec xmlns:ns4="http://serena.com/dimensions/ALF/DMEventExtentions/
1">
QLARIUS:UW_JAVA_2.0</ns4:ProjectSpec>
          <ns5:User
xmlns:ns5="http://serena.com/dimensions/ALF/DMEventExtentions/1">dmsys</ns5:User>
          <ns6:Baseline
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1">
            <ns7:uid
xmlns:ns7="http://serena.com/dimensions/baselines/1">4206276</ns7:uid>
            <ns8:specification
xmlns:ns8="http://serena.com/dimensions/baselines/1">QLARIUS:PRJ BL
01</ns8:specification>
            <ns9:type
xmlns:ns9="http://serena.com/dimensions/baselines/1">PRODUCTION</ns9:type>
            <ns10:description
xmlns:ns10="http://serena.com/dimensions/baselines/1">Baseline PRJ BL 01.AAAA</
ns10:description>
            <ns11:attributes
xmlns:ns11="http://serena.com/dimensions/baselines/1">
              <ns12:attribute xmlns:ns12="http://serena.com/dimensions/basic/1">
                <ns12:name>PROD_INSTALL_DATE</ns12:name>
                <ns12:datatype>Date</ns12:datatype>
                <ns12:type>Single_Value</ns12:type>
                <ns12:value></ns12:value>
              </ns12:attribute>
              <ns13:attribute xmlns:ns13="http://serena.com/dimensions/basic/1">
                <ns13:name>PROD_REMOVE_DATE</ns13:name>
                <ns13:datatype>Date</ns13:datatype>
                <ns13:type>Single_Value</ns13:type>
                <ns13:value></ns13:value>
              </ns13:attribute>
              <ns14:attribute xmlns:ns14="http://serena.com/dimensions/basic/1">
                <ns14:name>FAILURE_REASON</ns14:name>
                <ns14:datatype>Char</ns14:datatype>
                <ns14:type>Single_Value</ns14:type>
                <ns14:value>Other</ns14:value>
              </ns14:attribute>
              <ns15:attribute xmlns:ns15="http://serena.com/dimensions/basic/1">
                <ns15:name>FAILURE_DESC</ns15:name>
                <ns15:datatype>Char</ns15:datatype>
                <ns15:type>Single_Value</ns15:type>
                <ns15:value>Some failure description...</ns15:value>
              </ns15:attribute>
            </ns11:attributes>
            <ns16:relatedRequests
xmlns:ns16="http://serena.com/dimensions/baselines/1">
              <ns17:relatedRequest
xmlns:ns17="http://serena.com/dimensions/basic/1">
                <ns17:requestId>QLARIUS_CR_31</ns17:requestId>
                <ns17:relationshipType>In Response To</ns17:relationshipType>
              </ns17:relatedRequest>
              <ns18:relatedRequest>
```

```
xmlns:ns18="http://serena.com/dimensions/basic/1">
                <ns18:requestId>QLARIUS_CR_30</ns18:requestId>
                <ns18:relationshipType>Information</ns18:relationshipType>
              </ns18:relatedRequest>
            </ns16:relatedRequests>
            <ns19:buildJobId
xmlns:ns19="http://serena.com/dimensions/baselines/1">229</ns19:buildJobId>
          <ns20:buildJobUrl
xmlns:ns20="http://serena.com/dimensions/baselines/1">
http://acme:8080/bws/Monitor_job_info_modal.do?current=true&amp;job_id=229
</ns20:buildJobUrl><ns14:buildJobStatus xmlns:ns14="http://serena.com/dimensions/
baselines/1">Cancelled</ns14:buildJobStatus>
          </ns6:Baseline>
        </ns2:Extension>
      </EventNotice>
    </EventNotice>
  </soapenv:Body>
</soapenv:Envelope>
```

Deploy Baseline ALF Event

This event is fired on performing a Dimensions CM Deploy Baseline (DPB) command. The event details the baseline that was deployed and other relevant information. The event is received when deployment completes.

NOTE The use of the Deploy Baseline Alf event is deprecated, it not being recommended for use in new orchestrations. Instead, use the Deploy Baseline To Area ALF event (see "Deploy Baseline/Item/Request to Area ALF Events" on page 510).

The following event extension data have values:

- Database
- ProjectSpec
- User
- Baseline::uid
- Baseline::specification
- Baseline::type
- Baseline::description
- Baseline::Attributes.

The following is a sample of a baseline build-submitted event (the build-completed event looks the same, only <EventType> is different):

:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1"
xmlns:alfn7="http://serena.com/dimensions/baselines/1"
xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1"
xmlns:schjob="http://serena.com/dimensions/scheduled.job/1"
xmlns:deplobj="http://serena.com/dimensions/deploy.obj/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:alfn4="http://serena.com/dimensions/ALF/DMEventExtensions/1"
xmlns:alfn5="http://serena.com/dimensions/basic/1"
xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>2c510796-351e-11e0-b9c4-4181caa099bb</alfs:EventId>
          <alfs:Timestamp>2011-02-10T14:00:52+00:00</alfs:Timestamp>
          <alfs:EventType>deploy</alfs:EventType>
          <alfs:ObjectType>Baseline</alfs:ObjectType>
          <alfs:ObjectId>MAIN_JAVA_TIP</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsBaseline</alfs:Product>
            <alfs:ProductVersion>12.2.2</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT03:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2011-02-10T14:00:52+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
<alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:MAINLINE_JAVA_STR</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:Baseline>
            <alfn7:uid>4217389</alfn7:uid>
            <alfn7:specification>MAIN_JAVA_TIP</alfn7:specification>
            <alfn7:type>BASELINE</alfn7:type>
            <alfn7:description>Baseline MAIN_JAVA_TIP.AAAA</alfn7:description>
         </alfn4:Baseline>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Deployment ALF Events

Event Extension for Deployment

All ALF events from a Dimensions CM deployment functional area provide `DMDeployExtensionType` type in the ALF event extension. Each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

`DMDeployExtensionType` contains:

Field	Description	
`Database`	The database that originated the event, in the format `DB_ID@SERVER-DB_CONNECTION`.	
`ProjectSpec`	The specification of the Dimensions CM project/stream context for the event, in the format `ProductId:ProjectId`.	
`User`	The Dimensions CM user who originated the event.	
`DeployObject`	`comment`	The comment that was specified for operation in / COMMENT parameter.
	`toStage`	Specifies the target stage to promote the object to.
	`traverseChildren`	The "Promote child requests" option value that was specified for the operation. Used with requests only.
	`areas`	The array of areas that were affected by operation and the result of the operation if known. Each area sub-element can contain: ■ `name` - the name of the deployment area. ■ `success` - the result of the deployment or rollback operation in the area. ■ `version` - the version of the area that was specified in the SRAV command. Only provided by rollback-area-version events. ■ `finalVersion` - the version that was applied to the area as a result of a successful deployment or rollback operation.
	`startTime`	Start time for the deployment operation to begin. Used with scheduled operations only.
	`items`	The array of target Items to deploy.
	`requests`	The array of requests to deploy.
	`baselines`	The array of baselines to deploy.
	`jobState`	The final aggregative state of a deployment job (`Succeeded` or `Failed`).
	`statusMessage`	The descriptions of the atomic file operations that were performed during deployment and their individual results as well as the result of whole job.
	`commandType`	Type of deployment operation that triggered the event to fire (deploy, rollback, promote, or demote).

Promote/Demote Item Events

These events are fired on performing Promote Item (`PMI`) and Demote Item (`DMI`) commands.

The following fields of the deploy event extension for ALF events have values:

- `comment`
- `toStage`
- `areas`
- `items`

The following is a sample of a Promote Item event:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1"
xmlns:alfn7="http://serena.com/dimensions/baselines/1"
xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1"
xmlns:schjob="http://serena.com/dimensions/scheduled.job/1"
xmlns:deplobj="http://serena.com/dimensions/deploy.obj/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:alfn4="http://serena.com/dimensions/ALF/DMEventExtensions/1"
xmlns:alfn5="http://serena.com/dimensions/basic/1"
xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>599806f7-3a08-11e0-b5bf-fb5beae3ebed</alfs:EventId>
          <alfs:Timestamp>2011-02-16T20:07:15+00:00</alfs:Timestamp>
          <alfs:EventType>promote-item</alfs:EventType>
          <alfs:ObjectType>Deploy</alfs:ObjectType>
          <alfs:ObjectId>QLARIUS:VS_BRANCHA_STR</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsDeploy</alfs:Product>
            <alfs:ProductVersion>12.2.2</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT03:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
```

```
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2011-02-16T20:07:15+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:VS_BRANCHA_STR</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:DeployObject>
            <deplobj:comment>promote 34757836589734659783465983756 9</deplobj:comment>
            <deplobj:toStage>SIT</deplobj:toStage>
            <deplobj:areas>
              <alfn5:area>
                <alfn5:name>LCL_SIT_VSBRNCHA_AREA01</alfn5:name>
              </alfn5:area>
            </deplobj:areas>
            <deplobj:items>
              <alfn1:item>
                <alfn1:uid>4217249</alfn1:uid>
                <alfn1:specification>QLARIUS:QLARIUS UNDERWRITER SLN.A-SRC;vs_s1_0#1
</alfn1:specification>
                <alfn1:type>SRC</alfn1:type>
                <alfn1:revision>vs_s1_0#1</alfn1:revision>
                <alfn1:projectFilename>Qlarius_Underwriter/Qlarius_Underwriter.sln
</alfn1:projectFilename>
                <alfn1:description>Item uploaded into Dimensions</alfn1:description>
                <alfn1:comment></alfn1:comment>
              </alfn1:item>
            </deplobj:items>
          </alfn4:DeployObject>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Deploy Baseline/Item/Request to Area ALF Events

These events are fired on the completion of Submit Deploy Baseline (SDPBL), Submit Deploy Item (SDPI), and Submit Deploy Request (SDPRQ) commands. They can also be the result of Promote Item (PMI), Demote Item (DMI), Promote Request (PMRQ), Demote Request (DMRQ), Promote Baseline (PMBL), and Demote Baseline (DMBL) commands.

These ALF events will contain the final area version if the deployment was successful.

The following fields of the deploy event extension for ALF events have values:

- toStage
- traverseChildren
- areas
- items

- requests
- baselines
- jobState
- statusMessage
- commandType

IMPORTANT! There are some extra restrictions that can prevent fields being populated in the ALF event. For example, whereas the Deploy Baseline to Area ALF event can contain both items and requests arrays:

- The Deploy Item to Area ALF event cannot contain requests or baselines arrays.
- The Deploy Request to Area ALF event cannot contain baseline arrays.

The following is a sample of a Deploy Item event:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:alfn6 = "http://serena.com/dimensions/projects/1"
    xmlns:alfn7 = "http://serena.com/dimensions/baselines/1"
    xmlns:alfn1 = "http://serena.com/dimensions/items/1"
    xmlns:alfn2 = "http://serena.com/dimensions/requests/1"
    xmlns:schjob = "http://serena.com/dimensions/scheduled.job/1"
    xmlns:deplobj = "http://serena.com/dimensions/deploy.obj/1"
    xmlns:alfn3 = "http://serena.com/dimensions/ALF/DMVocabulary/1"
    xmlns:alfn4 = "http://serena.com/dimensions/ALF/DMEventExtensions/1"
    xmlns:alfn5 = "http://serena.com/dimensions/basic/1"
    xmlns:alfs = "http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version = "1.0">
```

```
 <alfs:Base>
          <alfs:EventId>b01646b8-9b0a-11e3-a640-bdccb0b7fdd3</alfs:EventId>
          <alfs:Timestamp>2014-02-21T15:13:15+00:00</alfs:Timestamp>
          <alfs:EventType>deploy-item</alfs:EventType>
          <alfs:ObjectType>Deploy</alfs:ObjectType>
          <alfs:ObjectId>4228801</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsDeploy</alfs:Product>
            <alfs:ProductVersion>12.1</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT05:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2014-02-21T15:13:15+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:JAVA_BRANCHA_STR</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:DeployObject>
            <deplobj:comment>Sample deploy-item ALF event activated by PMI command</
deplobj:comment>
            <deplobj:toStage>SIT</deplobj:toStage>
            <deplobj:areas>
              <alfn5:area>
                <alfn5:name>LCL_SIT_JBRNCHA_AREA03</alfn5:name>
                <alfn5:success>true</alfn5:success>
                <alfn5:finalVersion>16</alfn5:finalVersion>
              </alfn5:area>
            </deplobj:areas>
            <deplobj:items>
              <alfn1:item>
                <alfn1:uid>4228048</alfn1:uid>
                <alfn1:specification>QLARIUS:MAIN10 JAVA-130678471X6076X22.A-
SRC;java_s1_1#1</alfn1:specification>
                <alfn1:type>SRC</alfn1:type>
                <alfn1:revision>java_s1_1#1</alfn1:revision>
                <alfn1:projectFilename>main10.java</alfn1:projectFilename>
                <alfn1:description>Item uploaded into Dimensions</alfn1:description>
                <alfn1:comment></alfn1:comment>
              </alfn1:item>
            </deplobj:items>
            <deplobj:jobState>Failed</deplobj:jobState>
            <deplobj:statusMessage>- Copied file 'main10.java'&#xA;(status "UNDER WORK",
file version 1,&#xA;file modification time "Fri Feb 21 11:26:32 2014 GMT", item revision
"java_s1_1#1")&#xA;to "SIT" deployment area "LCL_SIT_JBRNCHA_AREA03", directory
'C:\Serena_Workareas\cm_typical\SIT\JBRNCHA_AREA_03\' ...&#xA;(copied
okay).&#xA;Successfully applied changes for DEPLOY job 4228801 to area
```

```
LCL_SIT_JBRNCHA_AREA03.&#xA;Successfully executed DEPLOY job 4228801 in area
LCL_SIT_JBRNCHA_AREA03.&#xA;</deplobj:statusMessage>
            <deplobj:commandType>Promote</deplobj:commandType>
          </alfn4:DeployObject>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
```

Rollback Baseline/Item/Request ALF Events

These events are fired on the completion of Submit Rollback Baseline (SRBL), Submit Rollback Item (SRI), and Submit Rollback Request (SRRQ) commands. They can also be the result of Demote Item (DMI), Demote Request (DMRQ), and Demote Baseline (DMBL) commands.

The following fields of the deploy event extension for ALF events have values:

- toStage
- traverseChildren
- areas
- items
- requests
- baselines
- jobState
- statusMessage
- commandType

The following is a sample of Rollback Request event:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:alfn6 = "http://serena.com/dimensions/projects/1"
    xmlns:alfn7 = "http://serena.com/dimensions/baselines/1"
    xmlns:alfn1 = "http://serena.com/dimensions/items/1"
    xmlns:alfn2 = "http://serena.com/dimensions/requests/1"
    xmlns:schjob = "http://serena.com/dimensions/scheduled.job/1"
    xmlns:deplobj = "http://serena.com/dimensions/deploy.obj/1"
    xmlns:alfn3 = "http://serena.com/dimensions/ALF/DMVocabulary/1"
    xmlns:alfn4 = "http://serena.com/dimensions/ALF/DMEventExtensions/1"
    xmlns:alfn5 = "http://serena.com/dimensions/basic/1"
    xmlns:alfs = "http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version = "1.0">
        <alfs:Base>
          <alfs:EventId>fd7e23b8-9f0f-11e3-833d-b34f13a99b28</alfs:EventId>
          <alfs:Timestamp>2014-02-26T18:01:17+00:00</alfs:Timestamp>
          <alfs:EventType>rollback-request</alfs:EventType>
          <alfs:ObjectType>Deploy</alfs:ObjectType>
          <alfs:ObjectId>4228955</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsDeploy</alfs:Product>
            <alfs:ProductVersion>12.1</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT05:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2014-02-26T18:01:17+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
```

```
<alfs:EmUser></alfs:EmUser>
         </alfs:EventControl>
       </alfs:Base>
       <alfs:Extension>
         <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfn4:Database>
         <alfn4:ProjectSpec>QLARIUS:JAVA_BRANCHA_STR</alfn4:ProjectSpec>
         <alfn4:User>DMSYS</alfn4:User>
         <alfn4:DeployObject>
           <deplobj:comment>Sample ALF activated by SRRQ</deplobj:comment>
           <deplobj:toStage>SIT</deplobj:toStage>
           <deplobj:traverseChildren>true</deplobj:traverseChildren>
           <deplobj:areas>
             <alfn5:area>
               <alfn5:name>LCL_SIT_JBRNCHA_AREA03</alfn5:name>
               <alfn5:success>true</alfn5:success>
               <alfn5:finalVersion>18</alfn5:finalVersion>
             </alfn5:area>
           </deplobj:areas>
           <deplobj:requests>
             <alfn2:request>
               <alfn2:uid>4228866</alfn2:uid>
               <alfn2:specification>QLARIUS_CR_46</alfn2:specification>
               <alfn2:type>CR</alfn2:type>
               <alfn2:status>RAISED</alfn2:status>
               <alfn2:attributes>
                 <alfn5:attribute>
                   <alfn5:name>TITLE</alfn5:name>
                   <alfn5:datatype>Char</alfn5:datatype>
                   <alfn5:type>Single_Value</alfn5:type>
                   <alfn5:value>Deploy request ALF event</alfn5:value>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>SEVERITY</alfn5:name>
                   <alfn5:datatype>Char</alfn5:datatype>
                   <alfn5:type>Single_Value</alfn5:type>
                   <alfn5:value>Medium</alfn5:value>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>EST_DEV_EFFORT</alfn5:name>
                   <alfn5:datatype>Number</alfn5:datatype>
                   <alfn5:type>Single_Value</alfn5:type>
                   <alfn5:value>8</alfn5:value>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>EST_COMP_DATE</alfn5:name>
                   <alfn5:datatype>Date</alfn5:datatype>
                   <alfn5:type>Single_Value</alfn5:type>
                   <alfn5:value>2014-02-27T00:00:00+00:00</alfn5:value>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>RAISED_BY</alfn5:name>
                   <alfn5:datatype>Char</alfn5:datatype>
                   <alfn5:type>Single_Value</alfn5:type>
                   <alfn5:value>DMSYS</alfn5:value>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>CONTACT_DETAILS</alfn5:name>
                   <alfn5:datatype>Char</alfn5:datatype>
                   <alfn5:type>Multi_Value</alfn5:type>
                   <alfn5:multivalue></alfn5:multivalue>
                 </alfn5:attribute>
                 <alfn5:attribute>
                   <alfn5:name>DETAILS_OF_THE_SOLUTION</alfn5:name>
```

```
 <alfn5:datatype>Char</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>ACTUAL_DEV_EFFORT</alfn5:name>
                    <alfn5:datatype>Number</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>ACTUAL_COMP_DATE</alfn5:name>
                    <alfn5:datatype>Date</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>DEFER_REASON</alfn5:name>
                    <alfn5:datatype>Char</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>DATE_OF_DEPLOYMENT</alfn5:name>
                    <alfn5:datatype>Date</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>DATE_TEST_PASSED</alfn5:name>
                    <alfn5:datatype>Date</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                  <alfn5:attribute>
                    <alfn5:name>COMPLETION_DATE</alfn5:name>
                    <alfn5:datatype>Date</alfn5:datatype>
                    <alfn5:type>Single_Value</alfn5:type>
                    <alfn5:value></alfn5:value>
                  </alfn5:attribute>
                </alfn2:attributes>
              </alfn2:request>
            </deplobj:requests>
            <deplobj:jobState>Succeeded</deplobj:jobState>
            <deplobj:statusMessage>- Removed file 'main.c'&#xA;(status "UNDER WORK",
file version 1,&#xA;file modification time "Wed Feb 26 14:07:34 2014 GMT", item revision
"java_s1_1#1")&#xA;from "SIT" deployment area "LCL_SIT_JBRNCHA_AREA03", directory
'C:\Serena_Workareas\cm_typical\SIT\JBRNCHA_AREA_03\' ...&#xA;(removed
okay).&#xA;Successfully applied changes for ROLLBACK job 4228955 to area
LCL_SIT_JBRNCHA_AREA03.&#xA;Successfully executed ROLLBACK job 4228955 in area
LCL_SIT_JBRNCHA_AREA03.&#xA;</deplobj:statusMessage>
            <deplobj:commandType>Rollback</deplobj:commandType>
          </alfn4:DeployObject>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Rollback Area Version ALF Event

This event is fired on completing the Submit Rollback Area Version (SRAV) command.

The following fields of the deploy event extension for ALF events have values:

- toStage
- traverseChildren
- areas
- items
- requests
- baselines
- jobState
- statusMessage
- commandType

IMPORTANT! traverseChildren and commandType are not populated in rollback-area-version event

The following is a sample of Rollback Area Version event:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:alfn6 = "http://serena.com/dimensions/projects/1"
    xmlns:alfn7 = "http://serena.com/dimensions/baselines/1"
    xmlns:alfn1 = "http://serena.com/dimensions/items/1"
    xmlns:alfn2 = "http://serena.com/dimensions/requests/1"
    xmlns:schjob = "http://serena.com/dimensions/scheduled.job/1"
    xmlns:deplobj = "http://serena.com/dimensions/deploy.obj/1"
    xmlns:alfn3 = "http://serena.com/dimensions/ALF/DMVocabulary/1"
    xmlns:alfn4 = "http://serena.com/dimensions/ALF/DMEventExtensions/1"
    xmlns:alfn5 = "http://serena.com/dimensions/basic/1"
    xmlns:alfs = "http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version = "1.0">
        <alfs:Base>
          <alfs:EventId>d5c56254-9ae7-11e3-8617-a99c3c58fa98</alfs:EventId>
          <alfs:Timestamp>2014-02-21T11:03:46+00:00</alfs:Timestamp>
          <alfs:EventType>rollback-area-version</alfs:EventType>
          <alfs:ObjectType>Deploy</alfs:ObjectType>
          <alfs:ObjectId>4227957</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsDeploy</alfs:Product>
            <alfs:ProductVersion>12.1</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT05:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2014-02-21T11:03:46+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT05-DIM12</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:JAVA_BRANCHA_STR</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:DeployObject>
            <deplobj:comment>Sample rollback-area-version ALF event activated
```

```
by SRAV command</deplobj:comment>
            <deplobj:toStage>SIT</deplobj:toStage>
            <deplobj:areas>
              <alfn5:area>
                <alfn5:name>LCL_SIT_JBRNCHA_AREA01</alfn5:name>
                <alfn5:success>true</alfn5:success>
                <alfn5:version>10</alfn5:version>
                <alfn5:finalVersion>11</alfn5:finalVersion>
              </alfn5:area>
            </deplobj:areas>
            <deplobj:items>
              <alfn1:item>
                <alfn1:uid>4227002</alfn1:uid>
                <alfn1:specification>QLARIUS:RRRRERWRWRWRW CPP-129917417X5392X0.A-
SRC;java_s1_1#1</alfn1:specification>
                <alfn1:type>SRC</alfn1:type>
                <alfn1:revision>java_s1_1#1</alfn1:revision>
<alfn1:projectFilename>rrrrerwrwrwrw.cpp</alfn1:projectFilename>
                <alfn1:description>Item uploaded into Dimensions</alfn1:description>
                <alfn1:comment></alfn1:comment>
              </alfn1:item>
            </deplobj:items>
            <deplobj:jobState>Succeeded</deplobj:jobState>
            <deplobj:statusMessage>- Removed file 'rrrrerwrwrwrw.cpp'&#xA;(status "UNDER
WORK", file version 1,&#xA;file modification time "Wed Feb 12 15:50:45 2014 GMT", item
revision "java_s1_1#1")&#xA;from "SIT" deployment area "LCL_SIT_JBRNCHA_AREA01",
directory 'C:\Serena_Workareas\cm_typical\SIT\JBRNCHA_AREA_01\' ...&#xA;(removed
okay).&#xA;Successfully applied changes for ROLLBACK job 4227957 to area
LCL_SIT_JBRNCHA_AREA01.&#xA;Successfully executed ROLLBACK job 4227957 in area
LCL_SIT_JBRNCHA_AREA01.&#xA;</deplobj:statusMessage>
          </alfn4:DeployObject>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Item ALF Events

Event Extension for Item

All ALF events from a Dimensions CM item provide `DMItemExtensionType` type in the ALF event extension, and each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

`DMItemExtensionType` contains:

Field	Description
`Database`	The database that originated the event, in the format `DB_ID@SERVER-DB_CONNECTION`.
`ProjectSpec`	The specification of the Dimensions CM project/stream context for the event, in the format `ProductId:ProjectId`.
`User`	The Dimensions CM user who originated the event.

Field	Description	
Item	uid	A Unique integer identifier for the Dimensions CM request (item_catalogue.obj_uid).
	specification	The full specification of the item.
	type	The type of the item.
	revision	The Revision of the item.
	projectFilename	The project filename of the item in the scoped project/stream.
	description	The description of the item.
	comment	The comment of the item.
	attributes	An array of user defined attributes and values.
	relatedParts	An array of related design parts.
	relatedItems	An array of related items.
	relatedRequests	An array of related requests.
	relatedBaselines	An array of related baselines.
	relatedProjects	An array of related projects/streams.

Check-In Item ALF Event

This event is fired on performing Dimensions CM Return Item/Check In (RI) and upload item modification (UPLOAD) commands.

The following fields of the item event extension for ALF events have values:

- Database
- ProjectSpec
- User
- Item::uid
- Item::specification
- Item::type
- Item::revision
- Item::projectFilename
- Item::description
- Item::comments
- Item::Attributes
- Item::relatedRequests

The following is a sample of a check-in event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://serena.com/dimensions/items/1"
xmlns:ns4="http://serena.com/dimensions/requests/1"
xmlns:ns5="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1"
xmlns:ns7="http://serena.com/dimensions/basic/1"
xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <ns2:EventNotice>
      <EventNotice version="1.0">
        <ns2:Base>
          <ns2:EventId>b0665424-8970-11dd-996d-89ab1e34d68a</ns2:EventId>
          <ns2:Timestamp>2008-09-23T16:08:17+03:00</ns2:Timestamp>
          <ns2:EventType>check-in</ns2:EventType>
          <ns2:ObjectType>Item</ns2:ObjectType>
          <ns2:ObjectId>0A4</ns2:ObjectId>
          <ns2:Source>
            <ns2:Product>Serena Dimensions CM</ns2:Product>
            <ns2:ProductVersion>2009</ns2:ProductVersion>
            <ns2:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns2:ProductInstance>
            <ns2:ProductCallbackURI>
http://acme:8080/dmwebservice/services/dimwebservices</ns2:ProductCallbackURI>
          </ns2:Source>
          <ns2:User>
            <ns2:LoginID>DMSYS</ns2:LoginID>
            <ns2:Certificate>13763913...<Skipped>...8525A6EC</ns2:Certificate>
          </ns2:User>
            <ns2:EventControl>
            <ns2:EmEventId></ns2:EmEventId>
            <ns2:EmTimestamp>2008-09-23T16:08:17+03:00</ns2:EmTimestamp>
            <ns2:PrecedingEmEventId></ns2:PrecedingEmEventId>
            <ns2:ApplicationName></ns2:ApplicationName>
            <ns2:EventMatchName></ns2:EventMatchName>
            <ns2:ServiceFlowName></ns2:ServiceFlowName>
```

```
            <ns2:ServiceFlowId></ns2:ServiceFlowId>
            <ns2:Callback>false</ns2:Callback>
            <ns2:Environment></ns2:Environment>
            <ns2:EmUser></ns2:EmUser>
          </ns2:EventControl>
        </ns2:Base>
<ns2:Extension>
          <ns6:Database>QLARIUS_CM@UA003679-DIM10</ns6:Database>
          <ns6:ProjectSpec>QLARIUS:UW_JAVA_2.0</ns6:ProjectSpec>
          <ns6:User>DMSYS</ns6:User>
          <ns6:Item>
            <ns3:uid>4207432</ns3:uid>
            <ns3:specification>QLARIUS:0A4.A-SRC;1.1</ns3:specification>
            <ns3:type>SRC</ns3:type>
            <ns3:revision>1.1</ns3:revision>
            <ns3:projectFilename>qlarius/interfaces/readmetoo.txt</ns3:projectFilename>
            <ns3:description>Item 0A4.A</ns3:description>
            <ns3:comment>Some comment on changing Item</ns3:comment>
            <ns3:attributes>
              <ns7:attribute>
                <ns7:name>REVIEW_TYPE</ns7:name>
                <ns7:datatype>Char</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value></ns7:value>
              </ns7:attribute>
            </ns3:attributes>
            <ns3:relatedRequests>
              <ns7:relatedRequest>
                <ns7:requestId>QLARIUS_CR_84</ns7:requestId>
                <ns7:relationshipType>In-Response To</ns7:relationshipType>
              </ns7:relatedRequest>
            </ns3:relatedRequests>
          </ns6:Item>
        </ns2:Extension>
      </EventNotice>
    </ns2:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Check-Out Item ALF Event

This event is fired on performing Dimensions CM Extract Item for Update/Check Out (EI) and upload item modification (UPLOAD) commands.

The following fields of the item event extension for ALF events have values:

- Database
- ProjectSpec
- User
- Item::uid
- Item::specification
- Item::type
- Item::revision
- Item::projectFilename
- Item::description

- Item::Attributes
- Item::relatedRequests

The following is a sample of a check-out event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://serena.com/dimensions/items/1"
xmlns:ns4="http://serena.com/dimensions/requests/1"
xmlns:ns5="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1"
xmlns:ns7="http://serena.com/dimensions/basic/1"
xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <ns2:EventNotice>
      <EventNotice version="1.0">
        <ns2:Base>
          <ns2:EventId>a7f3df46-8970-11dd-8afc-89ab1e34d68a</ns2:EventId>
          <ns2:Timestamp>2008-09-23T16:08:03+03:00</ns2:Timestamp>
          <ns2:EventType>check-out</ns2:EventType>
          <ns2:ObjectType>Item</ns2:ObjectType>
          <ns2:ObjectId>0A4</ns2:ObjectId>
          <ns2:Source>
            <ns2:Product>Serena Dimensions CM</ns2:Product>
            <ns2:ProductVersion>2009</ns2:ProductVersion>
            <ns2:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns2:ProductInstance>
            <ns2:ProductCallbackURI>
http://acme:8080/dmwebservice/services/dimwebservices</ns2:ProductCallbackURI>
          </ns2:Source>
          <ns2:User>
            <ns2:LoginID>DMSYS</ns2:LoginID>
            <ns2:Certificate>E19B17C9...<Skipped>...7001DBB2</ns2:Certificate>
          </ns2:User>
          <ns2:EventControl>
            <ns2:EmEventId></ns2:EmEventId>
            <ns2:EmTimestamp>2008-09-23T16:08:03+03:00</ns2:EmTimestamp>
            <ns2:PrecedingEmEventId></ns2:PrecedingEmEventId>
            <ns2:ApplicationName></ns2:ApplicationName>
            <ns2:EventMatchName></ns2:EventMatchName>
            <ns2:ServiceFlowName></ns2:ServiceFlowName>
            <ns2:ServiceFlowId></ns2:ServiceFlowId>
            <ns2:Callback>false</ns2:Callback>
            <ns2:Environment></ns2:Environment>
            <ns2:EmUser></ns2:EmUser>
```

```
        </ns2:EventControl>
       </ns2:Base>
       <ns2:Extension>
         <ns6:Database>QLARIUS_CM@UA003679-DIM10</ns6:Database>
         <ns6:ProjectSpec>QLARIUS:UW_JAVA_2.0</ns6:ProjectSpec>
         <ns6:User>DMSYS</ns6:User>
         <ns6:Item>
           <ns3:uid>4207432</ns3:uid>
           <ns3:specification>QLARIUS:0A4.A-SRC;1.1</ns3:specification>
           <ns3:type>SRC</ns3:type>
           <ns3:revision>1.1</ns3:revision>
           <ns3:projectFilename>qlarius/interfaces/readmetoo.txt</ns3:projectFilename>
           <ns3:description>Item 0A4.A</ns3:description>
           <ns3:comment></ns3:comment>
           <ns3:attributes>
             <ns7:attribute>
               <ns7:name>REVIEW_TYPE</ns7:name>
               <ns7:datatype>Char</ns7:datatype>
               <ns7:type>Single_Value</ns7:type>
               <ns7:value></ns7:value>
             </ns7:attribute>
           </ns3:attributes>
           <ns3:relatedRequests>
             <ns7:relatedRequest>
               <ns7:requestId>QLARIUS_CR_84</ns7:requestId>
               <ns7:relationshipType>In-Response To</ns7:relationshipType>
             </ns7:relatedRequest>
           </ns3:relatedRequests>
         </ns6:Item>
       </ns2:Extension>
     </EventNotice>
   </ns2:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create Item ALF Event

This event is fired on performing a Dimensions CM Create Item `CI` command.

The following fields of the item event extension for ALF events have values:

- `Database`
- `ProjectSpec`
- `User`
- `Item::uid`
- `Item::specification`
- `Item::type`
- `Item::revision`
- `Item::projectFilename`
- `Item::description`
- `Item::comments`
- `Item::Attributes`
- `Item::relatedRequests`

The following is a sample of a create event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://serena.com/dimensions/items/1"
xmlns:ns4="http://serena.com/dimensions/requests/1"
xmlns:ns5="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1"
xmlns:ns7="http://serena.com/dimensions/basic/1"
xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <ns2:EventNotice>
      <EventNotice version="1.0">
        <ns2:Base>
          <ns2:EventId>9b6f86c6-8970-11dd-abf9-89ab1e34d68a</ns2:EventId>
          <ns2:Timestamp>2008-09-23T16:07:42+03:00</ns2:Timestamp>
          <ns2:EventType>create</ns2:EventType>
          <ns2:ObjectType>Item</ns2:ObjectType>
          <ns2:ObjectId>0A4</ns2:ObjectId>
          <ns2:Source>
            <ns2:Product>Serena Dimensions CM</ns2:Product>
            <ns2:ProductVersion>2009</ns2:ProductVersion>
            <ns2:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns2:ProductInstance>
            <ns2:ProductCallbackURI>
http://acme:8080/dmwebservice/services/dimwebservices</ns2:ProductCallbackURI>
          </ns2:Source>
          <ns2:User>
            <ns2:LoginID>DMSYS</ns2:LoginID>
            <ns2:Certificate>EE4E1768...<Skipped>...4E96CD7</ns2:Certificate>
          </ns2:User>
          <ns2:EmEventId></ns2:EmEventId>
            <ns2:EmTimestamp>2008-09-23T16:07:42+03:00</ns2:EmTimestamp>
            <ns2:PrecedingEmEventId></ns2:PrecedingEmEventId>
            <ns2:ApplicationName></ns2:ApplicationName>
            <ns2:EventMatchName></ns2:EventMatchName>
            <ns2:ServiceFlowName></ns2:ServiceFlowName>
            <ns2:ServiceFlowId></ns2:ServiceFlowId>
            <ns2:Callback>false</ns2:Callback>
            <ns2:Environment></ns2:Environment>
            <ns2:EmUser></ns2:EmUser>
          </ns2:EventControl>
        </ns2:Base>
```

```
        <ns2:Extension>
          <ns6:Database>QLARIUS_CM@UA003679-DIM10</ns6:Database>
          <ns6:ProjectSpec>QLARIUS:UW_JAVA_2.0</ns6:ProjectSpec>
          <ns6:User>DMSYS</ns6:User>
          <ns6:Item>
            <ns3:uid>4207427</ns3:uid>
            <ns3:specification>QLARIUS:0A4.A-SRC;1.0</ns3:specification>
            <ns3:type>SRC</ns3:type>
            <ns3:revision>1.0</ns3:revision>
            <ns3:projectFilename>qlarius/interfaces/readmetoo.txt</ns3:projectFilename>
            <ns3:description>Item 0A4.A</ns3:description>
            <ns3:comment>Some comment for Initial Revision</ns3:comment>
            <ns3:attributes>
              <ns7:attribute>
                <ns7:name>REVIEW_TYPE</ns7:name>
                <ns7:datatype>Char</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value></ns7:value>
              </ns7:attribute>
            </ns3:attributes>
            <ns3:relatedRequests>
              <ns7:relatedRequest>
                <ns7:requestId>QLARIUS_CR_84</ns7:requestId>
                <ns7:relationshipType>In-Response To</ns7:relationshipType>
              </ns7:relatedRequest>
            </ns3:relatedRequests>
          </ns6:Item>
        </ns2:Extension>
      </EventNotice>
    </ns2:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Undo-Check-Out Item ALF Event

This event is fired on performing a Dimensions CM Cancel Item Update/Undo Check Out (`CIU`) command.

The following fields of the item event extension for ALF events have values:

- `Database`
- `ProjectSpec`
- `User`
- `Item::uid`
- `Item::specification`
- `Item::type`
- `Item::revision`
- `Item::projectFilename`
- `Item::description`
- `Item::Attributes`
- `Item::relatedRequests`

The following is a sample of an undo-check-out event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://serena.com/dimensions/items/1"
xmlns:ns4="http://serena.com/dimensions/requests/1"
xmlns:ns5="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1"
xmlns:ns7="http://serena.com/dimensions/basic/1"
xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <ns2:EventNotice>
      <EventNotice version="1.0">
        <ns2:Base>
          <ns2:EventId>61934890-8a1d-11dd-9475-89ab1e34d68a</ns2:EventId>
          <ns2:Timestamp>2008-09-24T12:44:28+03:00</ns2:Timestamp>
          <ns2:EventType>undo-check-out</ns2:EventType>
          <ns2:ObjectType>Item</ns2:ObjectType>
          <ns2:ObjectId>AUTOQUOTE JAVA</ns2:ObjectId>
           <ns2:Source>
            <ns2:Product>Serena Dimensions CM</ns2:Product>
            <ns2:ProductVersion>2009</ns2:ProductVersion>
            <ns2:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns2:ProductInstance>
            <ns2:ProductCallbackURI>http://acme:8080/dmwebservice/services/
dimwebservices</ns2:ProductCallbackURI>
          </ns2:Source>
          <ns2:User>
            <ns2:LoginID>DMSYS</ns2:LoginID>
            <ns2:Certificate>BD509D369...<Skipped>...BBE37B5C</ns2:Certificate>
          </ns2:User>          <ns2:EventControl>
            <ns2:EmEventId></ns2:EmEventId>
            <ns2:EmTimestamp>2008-09-24T12:44:28+03:00</ns2:EmTimestamp>
            <ns2:PrecedingEmEventId></ns2:PrecedingEmEventId>
            <ns2:ApplicationName></ns2:ApplicationName>
            <ns2:EventMatchName></ns2:EventMatchName>
            <ns2:ServiceFlowName></ns2:ServiceFlowName>
            <ns2:ServiceFlowId></ns2:ServiceFlowId>
            <ns2:Callback>false</ns2:Callback>
            <ns2:Environment></ns2:Environment>
            <ns2:EmUser></ns2:EmUser>
          </ns2:EventControl>
         </ns2:Base><ns2:Extension>
          <ns6:Database>QLARIUS_CM@UA003679-DIM10</ns6:Database>
          <ns6:ProjectSpec>QLARIUS:UW_JAVA_2.0</ns6:ProjectSpec>
          <ns6:User>DMSYS</ns6:User>
          <ns6:Item>
            <ns3:uid>4207447</ns3:uid>
            <ns3:specification>QLARIUS:AUTOQUOTE JAVA.A-SRC;2.1</ns3:specification>
```

```
            <ns3:type>SRC</ns3:type>
            <ns3:revision>2.1</ns3:revision>
            <ns3:projectFilename>qlarius/interfaces/AutoQuote.java</ns3:projectFilename>
            <ns3:description>Item uploaded into Dimensions</ns3:description>
            <ns3:comment></ns3:comment>
            <ns3:attributes>
              <ns7:attribute>
                <ns7:name>REVIEW_TYPE</ns7:name>
                <ns7:datatype>Char</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value></ns7:value>
              </ns7:attribute>
            </ns3:attributes>
          </ns6:Item>
        </ns2:Extension>
      </EventNotice>
    </ns2:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Project ALF Events

Event Extension for Project

All ALF events from a Dimensions CM project provide the `DMProjectExtensionType` type, and each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

`DMProjectExtensionType` contains:

Field	Description
`Database`	The database that originated the event, in the format `DB_ID@SERVER-DB_CONNECTION`.
`ProjectSpec`	The specification of the Dimensions CM project/stream context for the event, in the format `ProductId:ProjectId`.
`User`	The Dimensions CM user who originated the event.

Field	Description	
Project	uid	A Unique integer identifier for the Dimensions CM project/stream (ws_catalogue.obj_uid).
	specification	The full specification of the project/stream.
	type	The type of the project/stream.
	status	The status of the project/stream.
	description	The description of the project/stream.
	comment	The comment of the item.
	attributes	An array of user defined attributes and values.
	relatedItems	An array of related items.
	relatedRequests	An array of related requests.
	buildJobId	Id of build job.
	buildJobUrl	URL to the build job status.
	buildJobStatus	Status of the built project (Success, Failed, or Cancelled).
	repositoryVersion	Current repository version of this project.
	basedOn	Details of the project, stream, or baseline that this project is based on.
	deliverChanges	Array of changes resulting from a deliver operation

Build-Submitted and Build-Completed Project ALF Events

These events are fired on performing a Dimensions CM Build Project (BLD) command.

NOTES These events are always fired if the DM_ALF_ENDPOINT parameter is specified in dm.cfg.

All the fields of the project event extensions are filled with the exception of deliverChanges.

The following is a sample of a project build-submitted event. A build-completed event is identical with the exception of a different <EventType>:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
  <soapenv:Body>
    <EventNotice xmlns="http://www.eclipse.org/alf/schema/EventBase/1">
      <EventNotice xmlns="">
        <ns1:Base xmlns:ns1="http://www.eclipse.org/alf/schema/EventBase/1">
          <ns1:EventId>222ece00-8973-11dd-9b1c-8ee10287e59a</ns1:EventId>
          <ns1:Timestamp>2008-09-23T13:25:47.872Z</ns1:Timestamp>
          <ns1:EventType>build-submitted</ns1:EventType>
          <ns1:ObjectType>Project</ns1:ObjectType>
          <ns1:ObjectId>UW_JAVA_2.0</ns1:ObjectId>
          <ns1:Source>
            <ns1:Product>Serena Dimensions CM</ns1:Product>
            <ns1:ProductVersion>2009</ns1:ProductVersion>
            <ns1:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns1:ProductInstance>
            <ns1:ProductCallbackURI>uri:dummy</ns1:ProductCallbackURI>
          </ns1:Source>
          <ns1:User>
            <ns1:LoginID>dmsys</ns1:LoginID>
            <ns1:Certificate></ns1:Certificate>
          </ns1:User>
          <ns1:EventControl>
            <ns1:EmEventId></ns1:EmEventId>
            <ns1:EmTimestamp>2008-09-23T13:25:47.872Z</ns1:EmTimestamp>
            <ns1:PrecedingEmEventId></ns1:PrecedingEmEventId>
            <ns1:ApplicationName></ns1:ApplicationName>
```

```
            <ns1:EventMatchName></ns1:EventMatchName>
            <ns1:ServiceFlowName></ns1:ServiceFlowName>
            <ns1:ServiceFlowId></ns1:ServiceFlowId>
            <ns1:Callback>false</ns1:Callback>
            <ns1:Environment></ns1:Environment>
            <ns1:EmUser/>
          </ns1:EventControl>
        </ns1:Base>
        <ns2:Extension xmlns:ns2="http://www.eclipse.org/alf/schema/EventBase/1">
          <ns3:Database
xmlns:ns3="http://serena.com/dimensions/ALF/DMEventExtentions/1">QLARIUS_CM@UA003679-
DIM10</ns3:Database>
          <ns4:ProjectSpec
xmlns:ns4="http://serena.com/dimensions/ALF/DMEventExtentions/1">
QLARIUS:UW_JAVA_2.0</ns4:ProjectSpec>
          <ns5:User
xmlns:ns5="http://serena.com/dimensions/ALF/DMEventExtentions/1">dmsys</ns5:User>
          <ns6:Project xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1">
            <ns7:uid xmlns:ns7="http://serena.com/dimensions/projects/1">4203131</
ns7:uid>
            <ns8:specification
xmlns:ns8="http://serena.com/dimensions/projects/1">QLARIUS:UW_JAVA_2.0</
ns8:specification>
            <ns9:type
xmlns:ns9="http://serena.com/dimensions/projects/1">PROJECT</ns9:type>
            <ns10:status xmlns:ns10="http://serena.com/dimensions/projects/
1">IMPLEMENTATION</ns10:status>
            <ns11:description xmlns:ns11="http://serena.com/dimensions/projects/1">
Underwriter Desktop Java development project</ns11:description>
            <ns12:attributes xmlns:ns12="http://serena.com/dimensions/projects/1">
              <ns13:attribute xmlns:ns13="http://serena.com/dimensions/basic/1">
                <ns13:name>PROJ_CONCEPT_TARGET</ns13:name>
                <ns13:datatype>Date</ns13:datatype>
                <ns13:type>Single_Value</ns13:type>
                <ns13:value>2005-10-17T08:33:22Z</ns13:value>
              </ns13:attribute>
              <ns14:attribute xmlns:ns14="http://serena.com/dimensions/basic/1">
                <ns14:name>PROJ_PLANNING_TARGET</ns14:name>
                <ns14:datatype>Date</ns14:datatype>
                <ns14:type>Single_Value</ns14:type>
                <ns14:value>2005-10-23T21:00:00Z</ns14:value>
              </ns14:attribute>
              ...<Skipped>...
            </ns12:attributes>
            <ns22:relatedItems xmlns:ns22="http://serena.com/dimensions/projects/1">
              <ns23:relatedItem xmlns:ns23="http://serena.com/dimensions/basic/1">
                <ns23:itemSpec>QLARIUS:BUILD USER XML.A-SRC;1.0</ns23:itemSpec>
                <ns23:relationshipType>Owned</ns23:relationshipType>
              </ns23:relatedItem>
              <ns24:relatedItem xmlns:ns24="http://serena.com/dimensions/basic/1">
                <ns24:itemSpec>QLARIUS:BUILD XML.A-SRC;1.1</ns24:itemSpec>
                <ns24:relationshipType>Owned</ns24:relationshipType>
              </ns24:relatedItem>
              ...<Skipped>...
```

```
            </ns22:relatedItems>
            <ns37:relatedRequests xmlns:ns37="http://serena.com/dimensions/projects/1">
              <ns38:relatedRequest xmlns:ns38="http://serena.com/dimensions/basic/1">
                <ns38:requestId>QLARIUS_CR_84</ns38:requestId>
                <ns38:relationshipType>Owned</ns38:relationshipType>
              </ns38:relatedRequest>
              <ns39:relatedRequest xmlns:ns39="http://serena.com/dimensions/basic/1">
                <ns39:requestId>QLARIUS_CR_83</ns39:requestId>
                <ns39:relationshipType>Owned</ns39:relationshipType>
              </ns39:relatedRequest>
              ...<Skipped>...
            </ns37:relatedRequests>
            <ns61:buildJobId
xmlns:ns61="http://serena.com/dimensions/projects/1">222</ns61:buildJobId>
            <ns62:buildJobUrl xmlns:ns62="http://serena.com/dimensions/projects/1">
http://acme:8080/bws/Monitor_job_info_modal.do?current=true&amp;job_id=222
</ns62:buildJobUrl>
<ns14:buildJobStatus
xmlns:ns14="http://serena.com/dimensions/baselines/1">Cancelled</
ns14:buildJobStatus>          </ns6:Project>
        </ns2:Extension>
      </EventNotice>
    </EventNotice>
  </soapenv:Body>
</soapenv:Envelope>
```

Project deliver Events

A deliver event is fired when performing a deliver operation for a stream.

All of the Projects event fields are completed except for the following:

- `relatedItems`
- `relatedRequests`
- `buildJobId`
- `buildJobUrl`
- `buildJobStatus`

NOTES The deliver event does not include directory changes, such as directory renames, deletions, or additions.

The following is an example of a project deliver event:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:alfn6 = "http://serena.com/dimensions/projects/1"
    xmlns:alfn7 = "http://serena.com/dimensions/baselines/1"
    xmlns:alfn1 = "http://serena.com/dimensions/items/1"
    xmlns:alfn2 = "http://serena.com/dimensions/requests/1"
    xmlns:schjob = "http://serena.com/dimensions/scheduled.job/1"
    xmlns:deplobj = "http://serena.com/dimensions/deploy.obj/1"
    xmlns:alfn3 = "http://serena.com/dimensions/ALF/DMVocabulary/1"
    xmlns:alfn4 = "http://serena.com/dimensions/ALF/DMEventExtensions/1"
    xmlns:alfn5 = "http://serena.com/dimensions/basic/1"
    xmlns:alfs = "http://www.eclipse.org/alf/schema/EventBase/1">
    <SOAP-ENV:Body>
        <alfs:EventNotice>
            <EventNotice version = "1.0">
                <alfs:Base>
                    <alfs:EventId>9c72aea8-71e6-11e1-b8e7-eb1300caba1d</alfs:EventId>
                    <alfs:Timestamp>2012-03-19T17:11:47+00:00</alfs:Timestamp>
                    <alfs:EventType>deliver</alfs:EventType>
                    <alfs:ObjectType>Project</alfs:ObjectType>
                    <alfs:ObjectId>JAVA_BRANCHA_STR</alfs:ObjectId>
                    <alfs:Source>
                        <alfs:Product>DimensionsProject</alfs:Product>
                        <alfs:ProductVersion>12.1</alfs:ProductVersion>
                        <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT06-DIM12</
alfs:ProductInstance>
                       <alfs:ProductCallbackURI>http://UAL-DEV-VGUT06:8080/dmwebservice/
services/dimwebservices</alfs:ProductCallbackURI>
                    </alfs:Source>
                    <alfs:User></alfs:User>
                    <alfs:EventControl>
                        <alfs:EmEventId></alfs:EmEventId>
                        <alfs:EmTimestamp>2012-03-19T17:11:47+00:00</alfs:EmTimestamp>
                        <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
                        <alfs:ApplicationName></alfs:ApplicationName>
                        <alfs:EventMatchName></alfs:EventMatchName>
                        <alfs:ServiceFlowName></alfs:ServiceFlowName>
                        <alfs:ServiceFlowId></alfs:ServiceFlowId>
                        <alfs:Callback>false</alfs:Callback>
                        <alfs:Environment></alfs:Environment>
                        <alfs:EmUser></alfs:EmUser>
                    </alfs:EventControl>
                </alfs:Base>
```

```
 <alfs:Extension>
                    <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT06-DIM12</alfn4:Database>
                    <alfn4:ProjectSpec>QLARIUS:JAVA_BRANCHA_STR</alfn4:ProjectSpec>
                    <alfn4:User>DMSYS</alfn4:User>
                    <alfn4:Project>
                        <alfn6:uid>4217404</alfn6:uid>
                        <alfn6:specification>QLARIUS:JAVA_BRANCHA_STR</
alfn6:specification>
                        <alfn6:type>STREAM</alfn6:type>
                        <alfn6:status>OPEN</alfn6:status>
                        <alfn6:description>Java branch A</alfn6:description>
                        <alfn6:repositoryVersion>
                        <alfn5:version>385</alfn5:version>
                        <alfn5:baseVersion>373</alfn5:baseVersion>
                        </alfn6:repositoryVersion>
                        <alfn6:attributes>
                            <alfn5:attribute>
                                <alfn5:name>NUMBER25</alfn5:name>
                                <alfn5:datatype>Number</alfn5:datatype>
                                <alfn5:type>Single_Value</alfn5:type>
                                <alfn5:value>25</alfn5:value>
                            </alfn5:attribute>
                            <alfn5:attribute>
                                <alfn5:name>DATE23765872346</alfn5:name>
                                <alfn5:datatype>Date</alfn5:datatype>
                                <alfn5:type>Single_Value</alfn5:type>
                                <alfn5:value>2012-02-29T00:00:00+00:00</alfn5:value>
                            </alfn5:attribute>
                        </alfn6:attributes>
<alfn6:deliverChanges>
                            <alfn5:additions></alfn5:additions>
                            <alfn5:modifications>
                                <alfn5:item>
                                    <alfn5:itemSpec>QLARIUS:312452345235234 CPP-
69617044X3324X4.A-SRC;java_s1_1#2</alfn5:itemSpec>
                                    <alfn5:path>312452345235234_rename353453.cpp</
alfn5:path>
                                    <alfn5:requests>
                                        <alfn5:value>QLARIUS_CR_40</alfn5:value>
                                    </alfn5:requests>
                                </alfn5:item>
                            </alfn5:modifications>
                            <alfn5:refactors>
                                <alfn5:item>
                                    <alfn5:itemSpec>QLARIUS:312452345235234 CPP-
69617044X3324X4.A-SRC;java_s1_1#1</alfn5:itemSpec>
                                    <alfn5:path>312452345235234_rename353453.cpp</
alfn5:path>
<alfn5:oldPath>312452345235234.cpp</alfn5:oldPath>
<alfn5:requests>
                                        <alfn5:value>QLARIUS_CR_40</alfn5:value>
                                    </alfn5:requests>
                                </alfn5:item>
                            </alfn5:refactors>
                            <alfn5:deletions></alfn5:deletions>
                            <alfn5:imports></alfn5:imports>
                        </alfn6:deliverChanges>
                    </alfn4:Project>
                </alfs:Extension>
            </EventNotice>
        </alfs:EventNotice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Project create Events

A project create event is fired when the create project (DWS) or create stream (CS) commands are executed.

All of the Projects event fields are completed except for `deliverChanges`.

The following is a sample of Project create event:

```
<EventNotice>
  <Base>
    <EventId>137ab8c4-4160-11e3-a7cc-0f98149f8104</EventId>
    <Timestamp>2013-10-30T05:37:45-07:00</Timestamp>
    <EventType>create</EventType>
    <ObjectType>Project</ObjectType>
    <ObjectId>ALF_STR3</ObjectId>
    <Source>
      <Product>DimensionsProject</Product>
      <ProductVersion>12.1</ProductVersion>
      <ProductInstance>CM_TYPICAL@STL-DEV-VBBA01-DIM12</ProductInstance>
      <ProductCallbackURI>http://STL-DEV-VBBA01:8080/dmwebservice/services/
dimwebservices</ProductCallbackURI>
    </Source>
    <User></User>
    <EventControl>
      <EmEventId></EmEventId>
      <EmTimestamp>2013-10-30T05:37:45-07:00</EmTimestamp>
      <PrecedingEmEventId></PrecedingEmEventId>
      <ApplicationName></ApplicationName>
      <EventMatchName></EventMatchName>
      <ServiceFlowName></ServiceFlowName>
      <ServiceFlowId></ServiceFlowId>
      <Callback>false</Callback>
      <Environment></Environment>
      <EmUser></EmUser>
    </EventControl>
  </Base>
  <Extension>
    <Database>CM_TYPICAL@STL-DEV-VBBA01-DIM12</Database>
    <ProjectSpec>TEST_ALF:STR1</ProjectSpec>
    <User>DMSYS</User>
    <Project>
      <uid>4315520</uid>
      <specification>TEST_ALF:ALF_STR3</specification>
      <type>STREAM</type>
      <status>OPEN</status>
      <description>CS based on STR1</description>
      <repositoryVersion>
        <version>1041</version>
        <baseVersion>1036</baseVersion>
      </repositoryVersion>
      <basedOn>
        <project>
          <specification>TEST_ALF:STR1</specification>
          <repositoryVersion>
            <version>1036</version>
            <baseVersion>0</baseVersion>
          </repositoryVersion>
        </project>
      </basedOn>
    </Project>
  </Extension>
</EventNotice>
```

Request ALF Events

Event Extension for Request

All ALF events from a Dimensions CM request provide `DMRequestExtensionType` type in the ALF event extension, and each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

`DMRequestExtensionType` contains:

Field	Description	
`Database`	The database that originated the event, in the format `DB_ID@SERVER-DB_CONNECTION`.	
`ProjectSpec`	The specification of the Dimensions CM project/stream context for the event, in the format `ProductId:ProjectId`.	
`User`	The Dimensions CM user who originated the event.	
`Request`	`uid`	A unique integer identifier for the Dimensions CM request (cm_catalogue.ch_uid).
	`specification`	The full specification of the request.
	`type`	The type of the request.
	`status`	The status of the request.
	`attributes`	An array of user defined attributes and values.
	`relatedParts`	An array of related design parts.
	`relatedItems`	An array of related items.
	`relatedRequests`	An array of related requests.
	`relatedBaselines`	An array of related baselines.
	`relatedRequirements`	An array of related requirements.
	`relatedProjectSpec`	Related project/stream.

Action Request ALF Event

This event is fired on performing a Dimensions CM Action Request (AC) command.

The following fields of the request event extension for ALF events have values:

- `Database`
- `ProjectSpec`
- `User`
- `Request::uid`
- `Request::specification`
- `Request::type`
- `Request::status`
- `Request::Attributes`

The following is a sample of an action event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://serena.com/dimensions/items/1"
xmlns:ns4="http://serena.com/dimensions/requests/1"
xmlns:ns5="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:ns6="http://serena.com/dimensions/ALF/DMEventExtentions/1"
xmlns:ns7="http://serena.com/dimensions/basic/1" xmlns:ns2="http://www.eclipse.org/alf/
schema/EventBase/1">
  <SOAP-ENV:Body>
    <ns2:EventNotice>
      <EventNotice version="1.0">
        <ns2:Base>
          <ns2:EventId>fb5d5ea2-889b-11dd-b5c1-89ab1e34d68a</ns2:EventId>
          <ns2:Timestamp>2008-09-22T14:45:40+03:00</ns2:Timestamp>
```

```
          <ns2:EventType>action</ns2:EventType>
          <ns2:ObjectType>Request</ns2:ObjectType>
          <ns2:ObjectId>QLARIUS_CR_84</ns2:ObjectId>
<ns2:Product>Serena Dimensions CM</ns2:Product>
            <ns2:ProductVersion>2009</ns2:ProductVersion>
            <ns2:ProductInstance>QLARIUS_CM@UA003679-DIM10</ns2:ProductInstance>
            <ns2:ProductCallbackURI>
http://acme:8080/dmwebservice/services/dimwebservices</ns2:ProductCallbackURI>
          </ns2:Source>
          <ns2:User>
            <ns2:LoginID>DMSYS</ns2:LoginID>
            <ns2:Certificate>A2C8E6E5...<Skipped>...2EDAA878</ns2:Certificate>
          </ns2:User>
          <ns2:EventControl>
            <ns2:EmEventId></ns2:EmEventId>
            <ns2:EmTimestamp>2008-09-22T14:45:40+03:00</ns2:EmTimestamp>
            <ns2:PrecedingEmEventId></ns2:PrecedingEmEventId>
            <ns2:ApplicationName></ns2:ApplicationName>
            <ns2:EventMatchName></ns2:EventMatchName>
            <ns2:ServiceFlowName></ns2:ServiceFlowName>
            <ns2:ServiceFlowId></ns2:ServiceFlowId>
           <ns2:Callback>false</ns2:Callback>
            <ns2:Environment></ns2:Environment>
            <ns2:EmUser></ns2:EmUser>
          </ns2:EventControl>
        </ns2:Base>
        <ns2:Extension><ns6:Database>QLARIUS_CM@UA003679-DIM10</ns6:Database>
          <ns6:ProjectSpec>QLARIUS:UW_JAVA_2.0</ns6:ProjectSpec>
          <ns6:User>DMSYS</ns6:User>
          <ns6:Request>
            <ns4:uid>4207393</ns4:uid>
            <ns4:specification>QLARIUS_CR_84</ns4:specification>
            <ns4:type>CR</ns4:type>
            <ns4:status>WORK</ns4:status>
            <ns4:attributes>
              <ns7:attribute>
                <ns7:name>TITLE</ns7:name>
                <ns7:datatype>Char</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value>WS2 Test 12/09/2008 18:55</ns7:value>
              </ns7:attribute>
              <ns7:attribute>
                <ns7:name>SEVERITY</ns7:name>
                <ns7:datatype>Char</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value>2</ns7:value>
              </ns7:attribute>
                <ns7:name>EST_DEV_EFFORT</ns7:name>
                <ns7:datatype>Number</ns7:datatype>
                <ns7:type>Single_Value</ns7:type>
                <ns7:value></ns7:value>
              </ns7:attribute>
            </ns4:attributes>
          </ns6:Request>
        </ns2:Extension>
</EventNotice>
    </ns2:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create Request ALF Event

This event is fired on performing a Dimensions CM Create Request (CC) command.

The following fields of the request event extension for ALF events have values:

- Database
- ProjectSpec
- User
- Request::uid
- Request::specification
- Request::type
- Request::status
- Request::Attributes

The following is a sample of a create event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1"
xmlns:alfn7="http://serena.com/dimensions/baselines/1"
xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:alfn4="http://serena.com/dimensions/ALF/DMEventExtensions/1"
xmlns:alfn5="http://serena.com/dimensions/basic/1"
xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>bacab9c1-a77c-11de-9e05-8310a805a371</alfs:EventId>
          <alfs:Timestamp>2009-09-22T16:35:01+03:00</alfs:Timestamp>
          <alfs:EventType>create</alfs:EventType>
          <alfs:ObjectType>Request</alfs:ObjectType>
          <alfs:ObjectId>QLARIUS_CR_113</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsRequest</alfs:Product>
            <alfs:ProductVersion>2009</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UA003652-DIM10</alfs:ProductInstance>
         <alfs:ProductCallbackURI>http://UA003652:8080/dmwebservice/services/
dimwebservices
```

```
</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2009-09-22T16:35:01+03:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UA003652-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:GREEN_FOR_USE</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:Request>
           <alfn2:uid>4227805</alfn2:uid>
            <alfn2:specification>QLARIUS_CR_113</alfn2:specification>
            <alfn2:type>CR</alfn2:type>
            <alfn2:status></alfn2:status>
            <alfn2:attributes>
              <alfn5:attribute>
                <alfn5:name>TITLE</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value>My Request</alfn5:value>
              </alfn5:attribute>
              <alfn5:attribute>
                <alfn5:name>DESCRIPTION</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
              <alfn5:attribute>
                <alfn5:name>SEVERITY</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
      <…Some attributes skipped…>
             </alfn5:attribute>
                <alfn5:name>REMEDY_USER</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
              <alfn5:attribute>
                <alfn5:name>REMEDY_SYSTEM</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
```

```
              </alfn5:attribute>
              <alfn5:attribute>
              <alfn5:name>REMEDY_TIMESTAMP</alfn5:name>
               <alfn5:datatype>Char</alfn5:datatype>
               <alfn5:type>Single_Value</alfn5:type>
               <alfn5:value></alfn5:value>
             </alfn5:attribute>
           </alfn2:attributes>
         </alfn4:Request>
       </alfs:Extension>
     </EventNotice>
   </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Deploy Request ALF Event

This event is fired on performing a Dimensions CM Deploy Request (DPR) command.

NOTE The use of the Deploy Request Alf event is deprecated, it not being recommended for use in new orchestrations. Instead, use the Deploy Request to Area ALF event (see "Deploy Baseline/Item/Request to Area ALF Events" on page 510).

The following fields of the request event extension for ALF events have values:

- `Database`
- `ProjectSpec`
- `User`
- `Request::uid`
- `Request::specification`
- `Request::type`
- `Request::status`
- `Request::Attributes`

The following is a sample of a deploy event:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1" xmlns:alfn7="http://serena.com/
dimensions/baselines/1" xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1" xmlns:schjob="http://serena.com/
dimensions/scheduled.job/1" xmlns:deplobj="http://serena.com/dimensions/deploy.obj/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1" xmlns:alfn4="http://
serena.com/dimensions/ALF/DMEventExtensions/1" xmlns:alfn5="http://serena.com/
dimensions/basic/1" xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>0c0143b3-3a03-11e0-92ef-878d50576767</alfs:EventId>
          <alfs:Timestamp>2011-02-16T19:29:18+00:00</alfs:Timestamp>
          <alfs:EventType>deploy</alfs:EventType>
          <alfs:ObjectType>Request</alfs:ObjectType>
          <alfs:ObjectId>QLARIUS_CR_46</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsRequest</alfs:Product>
            <alfs:ProductVersion>12.2.2</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UAL-DEV-VGUT03:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2011-02-16T19:29:18+00:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UAL-DEV-VGUT03-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:VS_BRANCHA_STR</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:Request>
            <alfn2:uid>4223317</alfn2:uid>
            <alfn2:specification>QLARIUS_CR_46</alfn2:specification>
            <alfn2:type>CR</alfn2:type>
            <alfn2:status>RAISED</alfn2:status>
          </alfn4:Request>
          </alfs:Extension>
</EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Update Request ALF Event

This event is fired on performing a Dimensions CM Update Request (UC) command.

The following fields of the request event extension for ALF events have values:

- Database
- ProjectSpec
- User
- Request::uid
- Request::specification
- Request::type
- Request::status
- Request::Attributes

The following is a sample of an update event:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1"
xmlns:alfn7="http://serena.com/dimensions/baselines/1"
xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:alfn4="http://serena.com/dimensions/ALF/DMEventExtensions/1"
xmlns:alfn5="http://serena.com/dimensions/basic/1"
xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
<SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>da8bc7ac-a77d-11de-a35c-8310a805a371</alfs:EventId>
          <alfs:Timestamp>2009-09-22T16:43:04+03:00</alfs:Timestamp>
          <alfs:EventType>update</alfs:EventType>
          <alfs:ObjectType>Request</alfs:ObjectType>
          <alfs:ObjectId>QLARIUS_CR_113</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsRequest</alfs:Product>
            <alfs:ProductVersion>2009</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@UA003652-DIM10</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://UA003652:8080/dmwebservice/services/
dimwebservices
```

```
</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2009-09-22T16:43:04+03:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
            <alfs:EmUser></alfs:EmUser>
          </alfs:EventControl>
        </alfs:Base>
         <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@UA003652-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:GREEN_FOR_USE</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:Request>
          <alfn2:uid>4227805</alfn2:uid>
            <alfn2:specification>QLARIUS_CR_113</alfn2:specification>
            <alfn2:type>CR</alfn2:type>
            <alfn2:status></alfn2:status>
         <alfn2:attributes>
             <alfn5:attribute>
                <alfn5:name>TITLE</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value>My Request</alfn5:value>
              </alfn5:attribute>             <alfn5:attribute>
                <alfn5:name>DESCRIPTION</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value>Added some description</alfn5:value>
              </alfn5:attribute>              <alfn5:attribute>
                <alfn5:name>SEVERITY</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
              <…Some attributes skipped…>
               <alfn5:attribute>
                <alfn5:name>REMEDY_USER</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
              <alfn5:attribute>
                <alfn5:name>REMEDY_SYSTEM</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
```

```
 <alfn5:attribute>
                <alfn5:name>REMEDY_TIMESTAMP</alfn5:name>
                <alfn5:datatype>Char</alfn5:datatype>
                <alfn5:type>Single_Value</alfn5:type>
                <alfn5:value></alfn5:value>
              </alfn5:attribute>
            </alfn2:attributes>
          </alfn4:Request>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Schedule Job ALF Events

Event Extension for Schedule Job

All ALF events from a Dimensions CM schedule job provide `DMRequestExtensionType` type in the ALF event extension, and each event supplies only the pertinent data for the operation firing the event to the client. This means that an individual event may not provide all of the data defined in the extended data to the client.

`DMRequestExtensionType` contains:

Field	Description	
`Database`	The database that originated the event, in the format `DB_ID@SERVER-DB_CONNECTION`.	
`ProjectSpec`	The specification of the Dimensions CM project/stream context for the event, in the format `ProductId:ProjectId`.	
`User`	The Dimensions CM user who originated the event.	
`ScheduledJob`	`uid`	A unique integer identifier for the Dimensions CM scheduled job.
	`name`	The name of the scheduled job.
	`createTime`	The creation time of the scheduled job.
	`startTime`	The start time of the scheduled job.
	`updateTime`	The latest updating time of the scheduled job.
	`status`	The current status of the scheduled job.
	`originator`	The user who created the scheduled job.
	`description`	A description of the scheduled job.
	`repeat`	Specifies a repeat time for the scheduled job.
	`runStatus`	The status of the scheduled job.
	`runCommandsStatuses`	Statuses of all related commands that were ran.

Run Schedule Job ALF Event

This event is fired on performing a Dimensions CM Run Scheduled Job (RSJ) command.

All the fields of the scheduled job event extension for ALF events are filled.

The following is a sample of a scheduled job run event:

```
 <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:alfn6="http://serena.com/dimensions/projects/1"
xmlns:alfn7="http://serena.com/dimensions/baselines/1"
xmlns:alfn1="http://serena.com/dimensions/items/1"
xmlns:alfn2="http://serena.com/dimensions/requests/1"
xmlns:schjob="http://serena.com/dimensions/scheduled.job/1"
xmlns:alfn3="http://serena.com/dimensions/ALF/DMVocabulary/1"
xmlns:alfn4="http://serena.com/dimensions/ALF/DMEventExtensions/1"
xmlns:alfn5="http://serena.com/dimensions/basic/1"
xmlns:alfs="http://www.eclipse.org/alf/schema/EventBase/1">
  <SOAP-ENV:Body>
    <alfs:EventNotice>
      <EventNotice version="1.0">
        <alfs:Base>
          <alfs:EventId>beda7ee4-4ded-11df-aa87-215a3869faf3</alfs:EventId>
          <alfs:Timestamp>2010-04-22T10:02:14+01:00</alfs:Timestamp>
          <alfs:EventType>run</alfs:EventType>
          <alfs:ObjectType>ScheduledJob</alfs:ObjectType>
          <alfs:ObjectId>Job#0001</alfs:ObjectId>
          <alfs:Source>
            <alfs:Product>DimensionsScheduledJob</alfs:Product>
            <alfs:ProductVersion>2010</alfs:ProductVersion>
            <alfs:ProductInstance>CM_TYPICAL@ACME-DIM10</alfs:ProductInstance>
            <alfs:ProductCallbackURI>http://acme:8080/dmwebservice/services/
dimwebservices</alfs:ProductCallbackURI>
          </alfs:Source>
          <alfs:User></alfs:User>
          <alfs:EventControl>
            <alfs:EmEventId></alfs:EmEventId>
            <alfs:EmTimestamp>2010-04-22T10:02:14+01:00</alfs:EmTimestamp>
            <alfs:PrecedingEmEventId></alfs:PrecedingEmEventId>
            <alfs:ApplicationName></alfs:ApplicationName>
            <alfs:EventMatchName></alfs:EventMatchName>
            <alfs:ServiceFlowName></alfs:ServiceFlowName>
            <alfs:ServiceFlowId></alfs:ServiceFlowId>
            <alfs:Callback>false</alfs:Callback>
            <alfs:Environment></alfs:Environment>
```

```
          </alfs:EventControl>
        </alfs:Base>
        <alfs:Extension>
          <alfn4:Database>CM_TYPICAL@ACME-DIM10</alfn4:Database>
          <alfn4:ProjectSpec>QLARIUS:STREAM_A</alfn4:ProjectSpec>
          <alfn4:User>DMSYS</alfn4:User>
          <alfn4:ScheduledJob>
            <schjob:uid>4216820</schjob:uid>
            <schjob:name>Job#0001</schjob:name>
            <schjob:createTime>2010-04-22T10:00:01+01:00</schjob:createTime>
            <schjob:startTime>2010-10-09T23:00:00+01:00</schjob:startTime>
           <schjob:updateTime>2010-04-22T10:02:14+01:00</schjob:updateTime>
           <schjob:status>Active</schjob:status>
            <schjob:originator>DMSYS</schjob:originator>
            <schjob:description>Schedule job description</schjob:description>
            <schjob:repeat>
              <alfn5:repeatTime>10</alfn5:repeatTime>
              <alfn5:repeatType>Minutes</alfn5:repeatType>
            </schjob:repeat>
            <schjob:runStatus>
              <alfn5:historyUid>4216826</alfn5:historyUid>
              <alfn5:status>Active</alfn5:status>
              <alfn5:user>DMSYS</alfn5:user>
              <alfn5:startTime>2010-04-22T10:01:50+01:00</alfn5:startTime>
              <alfn5:finishTime>2010-04-22T10:01:50+01:00</alfn5:finishTime>
            </schjob:runStatus>
            <schjob:runCommandsStatuses>
              <alfn5:commandStatus>
                <alfn5:relationUid>4216822</alfn5:relationUid>
                <alfn5:commandUid>4216821</alfn5:commandUid>
                <alfn5:createTime>2010-04-22T10:01:17+01:00</alfn5:createTime>
                <alfn5:user>DMSYS</alfn5:user>
               <alfn5:command>CC "QLARIUS" "CR"/ATTRIBUTES=(Title="Mikhails request")  /
WORKSET="QLARIUS:JAVA_TYPICAL_2.0"</alfn5:command>
                <alfn5:status>Success</alfn5:status>
              <alfn5:output>The request QLARIUS_CR_10 has been forwarded to CHRIS, CLEM,
DMSYS, JOE, RICK, RON&#xA;Operation completed&#xA;</alfn5:output>
              </alfn5:commandStatus>
            </schjob:runCommandsStatuses>
          </alfn4:ScheduledJob>
        </alfs:Extension>
      </EventNotice>
    </alfs:EventNotice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Chapter 20

RESTful Web Services

Introduction	550
Path Parameters	551
Query Parameters	551
Authentication Methods	552
Example JSON Responses	553
WADL Definition File	555
Traces and Logs	556
Configuring Default Server Parameters	557
Reference	558

Introduction

The primary element of a RESTful (Representational State Transfer) API web service is a *resource*. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. Resources can be grouped into *collections*. Each collection is homogeneous so that it contains only one type of resource. Resources can exist outside a collection and are referred to as singleton resources. Collections are also resources and can exist globally at the top level of an API, but can also be contained inside a single resource known as a sub-collection. Sub-collections are typically used to express a 'contained in' relationship.

For detailed information about RESTful web services see this web site:

http://restful-api-design.readthedocs.org/en/latest/resources.html

The URL used to represent resources is formed of:

- `hostname`

  Specifies the machine hosting the CM server.

- `port`

  Specifies the port number on that machine.

- `path`

  Specifies a resource path.

Dimensions CM includes the following RESTful resources and sub-collections:

```
deploymentareas
deploymentareas/{areaName}
groups
products
products/{productName}
products/{productName}/baselines
products/{productName}/baselines/{baselineName}
products/{productName}/baselinetemplates
products/{productName}/baselinetemplates/{templateName}
products/{productName}/parts
products/{productName}/parts/{partName}
products/{productName}/parts/{partName}/childparts
products/{productName}/projects
products/{productName}/projects/{projectName}
products/{productName}/projects/{projectName}/baselines
products/{productName}/projects/{projectName}/buildconfigurations
products/{productName}/projects/{projectName}/fileareas
products/{productName}/projects/{projectName}/stages
products/{productName}/roleassignments
roles
users
version
```

NOTE Only the `http GET` method is currently supported.

Path Parameters

Each resource can have one or more path parameters, in curly brackets {}, that you can use to substitute a resource name. In the following resource {productName} is a path parameter:

```
dmrestservices/products/{productName}/projects
```

Resource examples:

- `http://localhost:8080/dmrestservices/deploymentareas`

  Lists deployment areas in a base database.

- `http://localhost:8080/dmrestservices/products`

  Lists all products in a base database.

- `http://localhost:8080/dmrestservices/products/QLARIUS/projects`

  Lists all projects and streams in the product QLARIUS.

Query Parameters

Each resource has the same set of optional query parameters:

- `dbName`

  Specifies the database identifier.

- `dbConnection`

  Specifies the data source name for connecting to a database.

- `server`

  Specifies the Dimensions CM server host name.

This example retrieves the list of design parts for the QLARIUS product and uses query parameters to specify the Dimensions CM base database, data source name, and server to use.

Request

```
GET http://localhost:8080/dmrestservices/products/QLARIUS/
    parts?dbName=cm_typical&dbConnection=dim14&server=dimensions1
```

Response

```
{"designParts": [{
   "uid": 4213928,
   "name": "QLARIUS:QLARIUS.A;1",
   "description": "Qlarius Product"
},
{
   "uid": 4213936,
   "name": "QLARIUS:LIFE_INSURANCE.A;1",
   "description": "Qlarius Life Insurance application"

}]}
```

Authentication Methods

To access any Dimensions CM RESTful web services resource you must provide one of the following authentication credentials:

- Single Sign On (SSO)

  SSO is the primary and preferred mode of authentication for Dimensions RESTful web services. When calling Dimensions CM RESTful web services from SBM it is preferable to use SSO. For detailed instruction about how to configure SSO authentication for a REST Grid widget see the SBM documentation.

- HTTP basic authentication

  Use HTTP basic authentication when calling Dimensions RESTful web services from third party standalone tools that do not support Dimensions SSO. Serena recommends only using this mode of authentication over secured connections, such as SSL.

NOTE You cannot use user names or passwords as query or path parameters.

Example JSON Responses

This section contains examples of JSON responses.

Calling a RESTful Web Service

This example uses a cURL command (`http://curl.haxx.se`) to call a Dimensions CM RESTful web service.

Request

```
curl -k -u myuser:mypassword https://myserver:8443/dmrestservices/
    products?dbName=mydb&dbConnection=myprod&server=myserver
```

Response

```
{"products":[{"uid":4201855,"name":"$GENERIC"},{"uid":4242805,"name":"E
    MPLOYEE"},{"uid":4242076,"name":"PAYROLL"},{"uid":4213926,"name":"Q
    LARIUS"},{"uid":4240798,"name":"TRAVEL"}]}
```

Listing Groups

This example lists the names and descriptions of all the user groups in a base database.

Request

```
GET http://localhost:8080/dmrestservices/groups
```

Response

```
{"groups": [
      {
      "name": "ADMIN",
      "description": "Administration group"
   },
      {
      "name": "BUILDERS",
      "description": "Builders group"
   },
      {
      "name": "DEVELOPERS",
      "description": "Developers group"
   },
      {
      "name": "DEV LEADS",
      "description": "Development team leaders"
   },
      {
      "name": "LEADS",
      "description": "All Team Leaders/Managers"
   }

]}
```

Listing Projects

This example lists all the streams and projects in the QLARIUS product.

Request

```
GET http://localhost:8080/dmrestservices/products/QLARIUS/projects
```

Response

```
{"projects": [
      {
      "uid": 4217404,
      "name": "QLARIUS:JAVA_BRANCHA_STR",
      "type": "STREAM",
      "description": "Java branch A",
      "isStream": true
   },
      {
      "uid": 4217051,
      "name": "QLARIUS:MAINLINE_JAVA_STR",
      "type": "STREAM",
      "description": "Java mainline",
      "isStream": true
   },
      {
      "uid": 4217398,
      "name": "QLARIUS:VS_BRANCHA_STR",
      "type": "STREAM",
      "description": "VS branch A",
      "isStream": true
   }
]}
```

Listing Baselines

This example lists all the baselines in the QLARIUS product that are related to the stream MAINLINE_VS_STR.

Request

```
GET http://localhost:8080/dmrestservices/products/QLARIUS/projects/
    MAINLINE_VS_STR/baselines
```

Response

```
{"baselines": [{
   "uid": 4217387,
   "name": "QLARIUS:MAIN_VS_TIP",
   "type": "BASELINE",
   "description": "Baseline MAIN_VS_TIP.AAAA"
},
{
   "uid": 4217391,
   "name": "QLARIUS:VS_RELEASE_1.0",
   "type": "BASELINE",
   "description": "Baseline for release 1.0"
}]}in the product QLARIUS:
```

WADL Definition File

A Web Application Description Language (WADL) definition file is the RESTful equivalent of the SOAP web services WSDL file. You can access the WADL file at this URL:

`http://localhost:8080/dmrestservices/application.wadl`

NOTE The automatic generation of `application.wadl` is enabled by default. For security reasons you can disable access to it by setting the parameter `jersey.config.server.wadl.disableWadl` to `false` in the file `web.xml` located in: `common\Tomcat\8.0\webapps\dmrestservices\WEB-INF\web.xml`

Traces and Logs

Dimensions CM RESTful web services writes traces and logs to:

```
common\tomcat\8.0\logs\dmrestservices.log
```

You can configure traces and logs for the RESTful web services in the file `log4j.properties` configuration file located at:

```
common\tomcat\8.0\webapps\dimensions\WEB-INF\classes\
```

RESTful Logs

Logs for RESTful web services are controlled by the parameter `log4j.logger.com.serena.dimensions.restservices`:

```
# RESTFul web services core:
# Use the setting below for production.
log4j.logger.com.serena.dimensions.restservices=INFO, LOGFILE
# Use the setting below for high level diagnostics.
# log4j.logger.com.serena.dimensions.restservices=DEBUG, LOGFILE
# Use the setting below for low level diagnostics.
# log4j.logger.com.serena.dimensions.restservices=TRACE, LOGFILE
```

Connection Pool

Logs for the pool of Dimensions CM connections are controlled by the parameter `log4j.logger.com.serena.dimensions.connectionpool2`:

```
# Pool of Dimensions CM connections:
# Use the setting below for production.
log4j.logger.com.serena.dimensions.connectionpool2=INFO, LOGFILE
# Use the setting below for high level diagnostics.
```

Configuring Default Server Parameters

You can configure the following default server parameters in:

`common\Tomcat\8.0\webapps\dmrestservices\WEB-INF\web.xml`:

- To disable SSO authentication set the following parameter to 'true':

```
<param-name>disableSSOAuth</param-name>
<param-value>true</param-value>
```

- To disable HTTP basic authentication set the following parameter to 'true':

```
<param-name>disableHTTPBasicAuth</param-name>
<param-value>true</param-value>
```

- Specify a Dimensions CM user ID:

```
<param-name>username</param-name>
<param-value><userid></param-value>
```

- Specify the password of the CM user:

```
<param-name>password</param-name>
<param-value><password></param-value>
```

- Specify a CM database name:

```
<param-name>dbName</param-name>
<param-value><database name></param-value>
```

- Specify the CM database connection string:

```
<param-name>dbConnection</param-name>
<param-value><database connection string></param-value>
```

- Specify the host name:

```
<param-name>server</param-name>
<param-value><hostname></param-value>
```

IMPORTANT! Values that you set in a `GET` request take precedence over values specified in `web.xml`. The order of preference is:

1. SSO authentication
2. HTTP basic authentication
3. Values specified in `web.xml`

NOTE

- Restart Common Tomcat after changing `web.xml`.
- All context parameters in `web.xml`, except `disableSSOAuth` and `disableHTTPBasicAuth`, are commented out by default. For the changes to take effect check that XML comments are removed where necessary.

Reference

getDeploymentAreas

Relative Path

`deploymentareas`

Parameters

None

Description

Returns all deployment areas in the base database. Each deployment area object contains values for `name`, `type`, and `stage`.

URL Example

```
http://localhost:8080/dmrestservices/deploymentareas
```

getDeploymentAreaDetails

Relative Path

`deploymentareas/{areaName}`

Parameters

`areaName`

The name of the deployment area.

Description

Returns the system attributes of the deployment area specified in the `areaName` parameter.

URL Example

```
http://localhost:8080/dmrestservices/deploymentareas/
    LCL_DEV_JBRNCHA_AREA01
```

getGroups

Relative Path

`groups`

Parameters

None

Description

Returns all user groups in the base database. Each group object contains values for `name` and `description`.

URL Example

```
http://localhost:8080/dmrestservices/groups
```

getProducts

Relative Path

`products`

Parameters

None

Description

Returns all products in the base database. Each product object contains values for `UID` and `name`.

URL Example

```
http://localhost:8080/dmrestservices/products
```

getProductDetails

Relative Path

`products/{productName}`

Parameters

`productName`

The name of the product.

Description

Returns `UID` and `name` values for the product specified in the `productName` parameter.

URL Example

`http://localhost:8080/dmrestservices/products/QLARIUS`

getBaselines

Relative Path

`products/{productName}/baselines`

Parameters

`productName`

The name of the product.

Description

Returns all baselines in the product specified in the `productName` parameter. Each baseline object contains values for `UID`, `name`, `type`, and `description`.

URL Example

`http://localhost:8080/dmrestservices/products/QLARIUS/baselines`

getBaselineDetails

Relative Path

`products/{productName}/baselines/{baselineName}`

Parameters

`productName`

The name of the product.

`baselineName`

The name of the baseline.

Description

Returns the system and user defines attributes for the baseline identified by the parameters `productName` and `baselineName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/baselines/
MAIN_JAVA_TIP
```

getBaselineTemplates

Relative Path

`products/{productName}/baselinetemplates`

Parameters

`productName`

The name of the product.

Description

Returns all baseline templates in the product specified in the `productName` parameter. Each baseline template object contains values for `name` and `scope`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/baselinetemplates
```

getBaselineTemplateDetails

Relative Path

```
products/{productName}/baselinetemplates/{templateName}
```

Parameters

`productName`

The name of the product.

`templateName`

The name of the baseline template.

Description

Returns the system attributes of the baseline template identified by the parameters `productName` and `templateName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/
baselinetemplates/ALL_ITEMS_APPROVED
```

getDesignParts

Relative Path

```
products/{productName}/parts
```

Parameters

`productName`

The name of the product.

Description

Returns all the design parts in the product specified in the `productName` parameter. Each design part object contains values for `UID`, `name`, and `description`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/parts
```

getDesignPartDetails

Relative Path

`products/{productName}/parts/{partName}`

Parameters

`productName`

The name of the product.

partName

The name of the stream or project.

Description

Returns system and user defines attributes for design part identified by the parameters `productName` and `partName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/parts/
QLARIUS.A%3B1
```

getChildDesignParts

Relative Path

`products/{productName}/parts/{partName}/childparts`

Parameters

`productName`

The name of the product.

`partName`

The name of the design part.

Description

Returns the design parts that are children of the design part specified by the parameters `productName` and `partName`. Each design part object contains values for `UID`, `name`, and `description`.

URL Example

```
http://localhost:8080/dmrestservices/products/PAYROLL/parts/
PAYROLL.A%3B1/childparts
```

getProjects

Relative Path

```
products/{productName}/projects
```

Parameters

`productName`

The name of the product.

Description

Returns all streams and projects in the product specified in the `productName` parameter. Each stream object contains values for `UID`, `name`, `type`, `description`, and `stream`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects
```

getProjectDetails

Relative Path

```
products/{productName}/projects/{projectName}
```

Parameters

`productName`

The name of the product.

`projectName`

The name of the stream or project.

Description

Returns the system and user defines attributes for the stream or project identified by the parameters `productName` and `projectName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects/
MAINLINE_VS_STR
```

getProjectBaselines

Relative Path

products/{productName}/projects/{projectName}/baselines

Parameters

`productName`

The name of the product.

`projectName`

The name of the stream or project.

Description

Returns baselines scoped by the stream or project identified by the parameters `productName` and `projectName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects/
MAINLINE_VS_STR/baselines
```

getProjectBuildConfigurations

Relative Path

`products/{productName}/projects/{projectName}/buildconfigurations`

Parameters

`productName`

The name of the product.

`projectName`

The name of the stream or project.

Description

Returns the list of names of build configurations for the stream or project identified by the parameters `productName` and `projectName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects/
MAINLINE_JAVA_STR/buildconfigurations
```

getProjectFileAreas

Relative Path

```
products/{productName}/projects/{projectName}/fileareas
```

Parameters

`productName`

The name of the product.

`projectName`

The name of the stream or project.

Description

Returns the list of deployment areas for the stream or project identified by the parameters `productName` and `projectName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects/
JAVA_BRANCHA_STR/fileareas
```

getProjectStages

Relative Path

```
products/{productName}/projects/{projectName}/stages
```

Parameters

`productName`

The name of the product.

`projectName`

The name of the stream or project.

Description

Returns all deployment stages for the stream or project identified by the parameters `productName` and `projectName`.

URL Example

```
http://localhost:8080/dmrestservices/products/QLARIUS/projects/
MAINLINE_VS_STR/stages
```

getProductRoleAssignments

Relative Path

`products/{productName}/roleassignments`

Parameters

`productName`

The name of the product.

Description

Returns role assignments registered in the product specified in the `productName` parameter. Each role assignment object contains real values for `userName`, `roleName`, `partSpe`, and `roleCapability`.

URL Example

`http://localhost:8080/dmrestservices/products/QLARIUS/roleassignments`

getRoles

Relative Path

`roles`

Parameters

None

Description

Returns all roles in the base database. Each role object contains values for `name` and `description`.

URL Example

`http://localhost:8080/dmrestservices/roles`

getUsers

Relative Path

`users`

Parameters

None

Description

Returns all users in the base database. Each user object contains values for `name` and `fullUserName`.

URL Example

```
http://localhost:8080/dmrestservices/users
```

getVersion

Relative Path

`version`

Parameters

None

Description

Returns the version of the Dimensions CM server.

URL Example

```
http://localhost:8080/dmrestservices/version
```

Part 6

Appendixes

Part 6: Appendixes contains the following chapters:

Known DTK Event Issues	571
Using the Bill of Materials API	575

Appendix A

Known DTK Event Issues

Missing Events	572
Running External Executables	572
Running dmcli from Event Triggers	572

NOTE This appendix discusses a number of known issues that you should keep in mind when using DTK events.

Missing Events

Some operations are currently missing events being fired that you might expect to be fired. The Serena® Dimensions® CM commands that are affected are listed below.

AC – Action Request	No VALIDATE event is fired.
CMB – Create Merged Baseline	No POST_CREATE event is fired.
MWS – Merge project	No POST_CREATE event is fired.

Running External Executables

CAUTION! Do not use this feature unless it is essential.

External executables called from event triggers run as the Dimensions CM proxy user unless you have specified the `-dont_use_proxy` option in the `dmlsnr` configuration file or the `dmlsnr` command line. Specifying this option causes `dmappsrv` processes to run as the authenticated user instead of the Dimensions CM proxy user. This means that external executables will run as the authenticated user; however, it also locks the `dmappsrv` to the user session, which disables `dmappsrv` pooling.

Running dmcli from Event Triggers

If you intend to run `dmcli` from an event trigger, you should not specify connection parameters as part of the command. With no connection parameters, `dmcli` will connect to the same server and database as the session that called the event trigger.

If you want to connect to a different Dimensions CM database, but do so through the same Dimensions CM server as the session that called the event trigger, you must specify the database using the `DMDB` environment symbol.

Suitable command syntax for use with the `system()` C function is:

- UNIX

```
DMDB='dbname@dsn' dmcli 'Dimensions command'
```

- Windows

```
set DMDB=dbname@dsn&dmcli "Dimensions command"
```

NOTE On Windows there must be no whitespace between the DMDB value and the ampersand (&).

Connecting to a remote server is possible, but you must specify a password to dmcli, so this is not recommended.

The legacy command alias pcms exists for dmcli. If your existing event trigger code uses pcms connecting to the same database as the session that called the event trigger, no code changes should be necessary.

The following sections provide example code snippets for running *dmcli* through system().

Windows Example

```
/* start of snippet */
status = snprintf(cmdBuf, sizeof(cmdBuf),
"set DMDB=%s&dmcli \"%s\" 1>%s 2>%s", databaseSpecifier,
dimensionsCommand, stdoutLog, stderrLog);

if (status < 0)
{
/* Command formatting error. */
}
else
{
_flushall();
status = system(cmdBuf);
}
/* end of snippet */
```

where databaseSpecifier is set to <database>@<dsn>

UNIX Example

```
/* start of snippet */
status = snprintf(cmdBuf, sizeof(cmdBuf),
"DMDB=%s dmcli '%s' >%s 2>&1",
databaseSpecifier,dimensionsCommand, logFile);

if (status <= 0 || status >= sizeof(cmdBuf))
{
/* Command formatting error. */
}
else
{

status = system(cmdBuf);
}
/* end of snippet */
```

where databaseSpecifier is set to <database>@<dsn>

Appendix B

Using the Bill of Materials API

Introduction	576
Bill Of Materials API Reference	577

Introduction

This API is an extension of the public Dimensions API, defined in pcms_api.h (see "Writing Dimensions CM DTK Applications" on page 37). The API enables you to create a Bill of Materials (BOM) file on a distributed build platform so that it can communicate to a build server about the targets that were created, and what they were created of. The API does not detect or create dependency information, rather it allows dependency information that is available to be used.

NOTE This is a file based API and you do not need to connect to Dimensions CM.

Errors

- All functions returning an `int` return the usual Dimensions error codes of PCMS_OK, PCMS_FAIL and PCMS_ERROR.
- All functions that return a `_TCHAR*` indicate an error by returning NULL.
- To check if an error has occured, use the function PcmsBuildIsError.
- To proceed after an error (after recovery logic), use the function PcmsBuildResetError.
- To print an error message, use the variable PcmsErrorSrr global.

Persistence

String buffers returned to the caller belong to the API and do not need to be released. If the contents of a string is needed after the lifetime of the session (after PcmsBuildTerm), you will need to make a copy.

Connection

You do not have to connect to Dimensions to use the API functions.

Filenames

Filenames passed to these functions can be relative or absolute. In the BOM the filenames are always converted to absolute paths, relative to the current directory when the API was called. Dimensions node names, UNC paths, and network style addresses are not supported.

Bill Of Materials API Reference

typedef void* PcmsBuildSession;

This is a handle to a session with this API. You can construct several BOM files at the same time by establishing multiple handles. All functions require this to be passed as the first parameter.

PcmsBuildSession PcmsBuildInit (void);

Starts a session and returns a handle.

int PcmsBuildTerm (PcmsBuildSession h);

Terminates the session. This has to be called to free memory allocated during the session.

int PcmsBuildIsError (PcmsBuildSession h);

Determines if the session is in error. If it is in error, a message describing the condition that placed it in a state of error is available in the PcmsErrorStr global variable.

int PcmsBuildResetError (PcmsBuildSession h);

Resets the error state if you wish to carry on processing after a previous error. If you do not do this, then all functions except for PcmsBuildCloseBom and PcmsBuildTerm are unavailable.

_TCHAR * PcmsBuildGetSourceInfo (PcmsBuildSession h, _TCHAR *filename);

Return a string containing some information about the source file. This is the information that is required in the <metadata> part of the BOM. Note that this function is not intended as a general interface to Dimensions metadata.

_TCHAR * PcmsBuildPrepareNewTarget (PcmsBuildSession h, _TCHAR *filename);

Updates a new target so that it will be processed correctly by Dimensions Build. As well as updating a new target's metadata, this function returns some information about this file in exactly the same format as the previous function.

_TCHAR * PcmsBuildGetTempBomfile (PcmsBuildSession h);

Generate a new unique BOM file. The file is initialized with a "null BOM" that is entirely valid to use as a BOM, but as yet has no sources or targets attached. You can use the returned filename later in either a PcmsBuildOpenBom or PcmsBuildAppendBom call.

int PcmsBuildOpenBom (PcmsBuildSession h, _TCHAR *bomname);

Opens an output BOM file. The file is overwritten if it exists.

int PcmsBuildAppendBom (PcmsBuildSession h, _TCHAR *bomname);

Opens an output BOM file. The generated XML will be merged with the data that is already present in the file. The file must have previously been created with this same API or be in exactly the same format. If the existing file is not in the expected format an error is returned.

int PcmsBuildCloseBom (PcmsBuildSession h);

Closes the BOM file that you are currently writing to. Another file can then be opened for writing in the same session. The same file can be reopened later with the PcmsBuildAppendBom function.

int PcmsBuildBomOpenTransaction (PcmsBuildSession h);

Creates a new transaction that models the concept of a build step. Within a transaction, sources and targets are collected together. To move on to a new set of sources and targets, close the transaction and start another one.

int PcmsBuildBomCloseTransaction (PcmsBuildSession h);

Closes the current transaction. XML relating to the targets and sources of the current transaction will now be written to the BOM file.

int PcmsBuildBomAddSource (PcmsBuildSession h, _TCHAR *sourcefile, _TCHAR *info);

Adds a source to the current transaction. These can be major source files, or related files that help the process (for example, header files). In general, they should be files which are known to Dimensions. The info parameter suppies the string that will be placed in the <metadata> key of the BOM for this file. The parameter can be obtained from the PcmsBuildGetSourceInfo function. If info is NULL it is calculated automatically.

int PcmsBuildBomAddTarget (PcmsBuildSession h, _TCHAR *targetfile, _TCHAR *info);

Adds a target to the current transaction. You have to have at least one target to make a valid transaction, however sources are optional. You can add more than one target if the build produced more than one file in this step. The info parameter is as discussed above for PcmsBuildBomAddSource, but obtained from the PcmsBuildPrepareNewTarget function. If info is NULL it is calculated automatically.

int PcmsBuildBomAddListing (PcmsBuildSession h, _TCHAR *listingfile, _TCHAR *info);

Adds a listing file to this build step. This is the same as a target but it is flagged as being listing in the BOM so that the server can treat it differently. These are optional in a BOM. If info is NULL it is calculated automatically.

int PcmsBuildBomSetFootprintMode (PcmsBuildSession h, int footprint);

By default footprinting is off. Use this function to enable footprinting, or disable it again. Calling this function inside a transaction only has a temporary effect on targets within that transaction. Calling this function outside of a transaction, for example at the start, changes the setting globally.

For details about footprinting see the *Build Tools User's Guide*.

int PcmsBuildBomSetReturnCode (PcmsBuildSession h, int returncode);

Sets the step return code for this transaction. This can be used per transaction, or globally to change the default code from zero to another value.

Index

Symbols

)ADJUST template language directive 188
)ATTR template language directive 188
)CALL template language directive 230
)CALLBACK template language directive 189
)CM template language directive 189, 190
)DELETE template language directive 234, 235
)DUMP template language directive 191
)ELSE template language directive 191
)ENDEXPAND template language directive 193
)ENDIF template language directive 192
)ENDR template language directive 193
)ENDSCRIPT template language directive 230
)EXPAND template language directive 193
)IF template language directive 194
)IFDEF template language directive 194
)IFGROUPDEF template language directive 234, 235
)IFGROUPNDEF template language directive 234, 235
)IFNDEF template language directive 195
)IM template language directive 195
)LOGON template language directive 196
)LPAD template language directive 234, 235
)REP template language directive 196
)SAVE template language directive 234, 235
)SCRIPT template language directive 230
)SET template language directive 198
)SET_PATH template language directive 199
)SET_PATH_MVS template language directive 199
)SET_PATH_NT template language directive 200
)SET_PATH_UNIX template language directive 200
)SETL template language directive 198
)SETU template language directive 200
)SLICE template language directive 201
)VECTOR template language directive 201

Numerics

4Gbyte or greater files 38

A

account predefined template symbols 216
actionRequest 347
ALF events
  action request 342
  architecture
    configuring events 339
    events declaration 336
    generation 340
    introduction 336
  build baseline (completed) 341
  build baseline (submitted) 341
  build project (completed) 342
  build project (submitted) 342
  check in item 341
  check out item 341
  configuring 339
  create item 341
  create request 342
  datetime format
    SOAP/XML 343
  deploy baseline 341
  deploy items 341
  deploy items to area 341
  deploy request 342
  deploy request to area 342
  error message logs 343
  generation 340
  introduction 336, 500
  logging 343
  rollback baseline 341
  rollback item 341
  rollback request 342
  structure 500
  supported datetime formats 343
  undo check out 341
  update request 342
allocate memory 76
allocate zero initialized memory 74
API library file names 38
architecture
  client 34
  event 35
  interaction between client and event architecture 35
  scope of the DTK architecture 35
attribute macros 138
attributes
  system-defined 47
  user-defined 47
authentication methods 332

B

baseline ALF events
build submitted and build completed 503
deploy 506
event extension 502
Bill of Materials API 575
build templates
introduction 250
build and deployment job logs 260
build configuration inputs and outputs, using 257
build configurations, using 252
callouts 260
collection templates 254
compilation templates 254
coordination templates 254
default locations 251
expanding wild cards 256
footprinting 259
generating a bill of materials 258
handling error logs 258
handling outputs 258
high level templates 254
options 254
search paths 259
types 254
build templates, distributed
introduction 262
asynchronous templates 266
Bill Of Materials, creating 271
collection templates 264
compilation templates 263
formatted text after preprocessing, submitting 270
handling array variables 268
high level templates 263
Make, using 271
messaging, configuring 266
path or file specification, decomposing 269
search paths, using 265
special characters in Windows, using 267
specific array value, picking 269
STDERR and STDOUT streams, combining 270
templating techniques 267
build templates, MVS
introduction 276
Bill of Materials, creating from the SBEM 283
build step warning messages 282
final return code, computing 282
generated JCL, wrapping into the SBEM 280
source types 284
using outputs 281
using REXX 286
writing plain JCL templates 279
buildBaseline 349
buildProject 352

C

catsearch 242
change Dimensions CM default directory 134
character encoding 332
check results of Dimensions CM command 66
checkInItem 356
checkOutItem 360
client architecture 34
complex symbol references, templating language 217
connect silently to a Dimensions CM database 152
connect to a Dimensions CM database 71, 144
connection functions 142
constants
PCMS_BASELINE 52
PCMS_CHDOC 52
PCMS_CUSTOMERS 52
PCMS_EVENT_POST_OP 161
PCMS_EVENT_PRE_OP 160
PCMS_EVENT_VALIDATE_OP 160
PCMS_ITEM 52
PCMS_PART 52
PCMS_REL_AFF 52
PCMS_REL_BREAKDOWN 52
PCMS_REL_DEP 52
PCMS_REL_DERIVED 52
PCMS_REL_INFO 52
PCMS_REL_IRT 52
PCMS_REL_OWN 52
PCMS_REL_PERMANENT 53
PCMS_REL_PRED 52
PCMS_REL_SUCC 52, 53
PCMS_REL_TOP 52, 53
PCMS_REL_USE 52, 53
PCMS_USER 52
PCMS_WORKSET 52
contacting technical support 29
createDesignPart 367
createDesignPartBaseline 369
createProject 373
createProjectBaseline 377
createRequest 380
createRevisedBaseline 383
createScheduleJob 386

D

DBCS 38
delegateRequest 394
delegateRequestForReplication 396
demoteBaselines 398
demoteItems 401
demoteRequests 404
deployBaseline 407
deployment ALF events

deploy baseline/item/request to area 510
event extension 508
promote/demote items 509
rollback area version 517
rollback baseline/item/request 513
deployment area scripts
attributes 315
debugging 314
example 317
introduction 310
variables 312
deployRequest 409
Dimensions CM Event Callout Interface 158
Dimensions CM Web services
installing 330
Dimensions CM WSDL 323
Dimensions for z/OS predefined template symbols
job sequence numbers 216
user ID and account 216
disconnect from a Dimensions CM database 73, 145
DM_BUILDER_PROGRESS_REPORTING templating language, Dimensions Build optional symbols 215
DM_SP_START_STAGE templating language, Dimensions Build optional symbols 215
DM_TEMPLATE_CATALOG template variable 228
DM_TPLB_LOG template variable 228
DM_TPLB_SEQUENCE_PATH build variable 229
DM_TPLB_SPOOL template variable 228
DM_TPLB_TTLOG template variable 229
dm.cfg template variables
DM_TEMPLATE_CATALOG 228
DM_TPLB_LOG 228
DM_TPLB_SEQUENCE_PATH 229
DM_TPLB_SPOOL 228
DM_TPLB_TTLOG 229
DMALTSERVER templating language, Dimensions Build optional symbols 215
DMBUILDJOB. templating language, Dimensions Build standard symbols 211
DMCERTIFICATE templating language, remote job execution predefined symbols 207
DMCERTIFICATE. templating language, predefined symbols 202
dmclient
examples 181
introduction 178
Java API time format strings 180
Javadoc 181
using 179
DMCOMBIMEOUTERR. templating language, control symbols 207
DMCURRENTDIRECTORY. templating language, predefined symbols 202
DMCYGWIN templating language, predefined symbols 202
DMDAY templating language, predefined symbols 202
DMEXECENV templating language, Dimensions Build optional symbols 212
DMEXPAND templating language, Dimensions Build optional symbols 213
DMFINAL templating language, Dimensions Build optional symbols 214
DMFOLDERSEPARATOR templating language, predefined symbols 203
DMHOUR templating language, predefined symbols 203
DMINPUT templating language, Dimensions Build standard symbols 210
DMJOBID templating language, remote job execution predefined symbols 207
DMJOBID templating language, remote job execution standard symbols 211
DMMAXRC templating language, Dimensions Build optional symbols 214
DMMICROSEC templating language, predefined symbols 203
DMMINUTE templating language, predefined symbols 203
DMMONTH templating language, predefined symbols 203
DMMUSTRUN templating language, Dimensions Build optional symbols 214
DMNOCACHE templating language, Dimensions Build optional symbols 214
DMOSTDERR templating language, predefined symbols 203
DMOSTDOUT templating language, predefined symbols 204
DMOXTMPLT templating language, predefined symbols 204
DMPASS. templating language, Dimensions Build standard symbols 211
DMPASSWORD templating language, predefined symbols 204
DMPATH templating language, Dimensions Build standard symbols 208
DMPATHSEPARATOR templating language, predefined symbols 204
DMPBEMNODE templating language, Dimensions Build standard symbols 211
DMPBEMPORT templating language, Dimensions Build standard symbols 211
DMREBUILDALL templating language, Dimensions Build optional symbols 214
DMRUNMODE templating language, Dimensions Build standard symbols 210
DMSAVESTDERR. templating language, control symbols 206
DMSAVESTDOUT templating language 206

DMSAVESTDOUT templating language, control symbols 206
DMSECOND templating language, predefined symbols 204
DMSEQ9 predefined z/OS symbols 216
DMSEQ99 predefined z/OS symbols 216
DMSEQ999 predefined z/OS symbols 216
DMSEQA predefined z/OS symbols 216
DMSEQAA predefined z/OS symbols 216
DMSEQAAA predefined z/OS symbols 216
DMSEQAAAA predefined z/OS symbols 216
DMSEQNNN predefined z/OS symbols 216
DMSERVER templating language, Dimensions Build standard symbols 211
DMSTEP templating language, Dimensions Build standard symbols 210
DMSTEPMODE templating language, Dimensions Build predefined symbols 214
DMSYSDIR predefined z/OS symbols 216
DMSYSGID predefined z/OS symbols 216
DMSYSSHL predefined z/OS symbols 216
DMSYSUID predefined z/OS symbols 216
DMSYSUSR predefined z/OS symbols 216
DMTARGET templating language, Dimensions Build standard symbols 210
DMTASKNAME templating language, Dimensions Build optional symbols 213
DMTIMEOUT templating language, Dimensions Build optional symbols 213
DMTOKEN templating language, Dimensions Build standard symbols 211
DMTTLOG templating language, Dimensions Build optional symbols 215
DMUNIQUE templating language, predefined symbols 205
DMUSER templating language, predefined symbols 205
DMUSER templating language, standard symbols 211
DMUSERU templating language, predefined symbols 205
DMXFERSOURCE templating language, Dimensions Build standard symbols 211
DMYEAR templating language, predefined symbols 205
Double-Byte Character Set 38
DTK architecture 35
durules predefined call 230

E

ENQs, controlling conflicting 290
event architecture 35
event callout interface 158
events
  calling DTK functions 171
  changing system-attributes 169
  changing user-attributes 170
  designing 166
  determining the event you want 163
  event callout interface 160
  examples 173
  first and second calls 164
  pointers 168
  post-events 161
  pre-events 160
  public function call 159, 168
  shared libraries 158
  summary 165
  types 161
  unsupported function calls 171
  usage warning 174
  using ptrEventInfo 172
  validate 160
  writing 167
  writing a DTK callout 165
examples
  asynchronous remote job 302
  code snippets for running dmcli through system() 573
  deployment area script 317
  remote job template 303
  synchronous REXEC template 301
execute a Dimensions CM command 146
  asynchronously 126
  synchronously 78

F

find Dimensions CM objects and return complete objects 80
find Dimensions CM objects and return uids 122
free a list of values 110
free Dimensions CM object structures 111
free memory 75, 147
function call results 40

G

get attribute
  definition 61
  definition number 85
get attribute's list of values 62
get Dimensions CM object
  attributes 88
  details by specification 108
  details by uid 109
  relationships 114
  reverse relationships 112
get Dimensions CM process model information 68
get input file description 92

get request descriptions 86
get the Dimensions CM command 91
get the last Dimensions CM message 148
get the last Dimensions CM message (dynamic buffer) 149
get user's current project 107
getItems 411
getRequestDescription 415
getRequests 417
getVersion 422

I

idle connection time-out for Web services 330
include file 38
install idle checker 135
installing Dimensions CM Web services 330
is request object in secondary catalog 116
item ALF events
  check in 520
  check out 522
  create 524
  event extension 519
  undo check out 526

J

Java API
  Ant 181
  CruiseControl 181
  time format strings 180
Javadoc 181
job sequence number template symbols, on z/OS 216

K

known DTK event issues
  missing events 572
  running dmcli from event triggers 572
  running external executables 572

L

large file support (greater than 4Gbytes) 38
listProjectItems 426
listProjectRequests 429
listProjects 431
listScheduleJobs 434
look up symbol in structured information return table 101

M

MBCS 38
MDHDSN template inline function 220
MDHEXTRACT template inline function 223
memory allocation 53
moveItemToPart 441
Multi-Byte Character Set 38

O

obtain
  inbox user structures 94
  role section names for a product 98
  user relationship subtypes 103
  user role structures 105
obtain list of loaded DLLs and dates and times for each 93
obtain values associated with a symbol 100
Openmake templates
  introduction 294
  variables 295

P

pcms_api_sql_uk.msb 154
pcms_api.h 38
pcms_api.lib 38
pcms_api.so 38
PCMS_ERROR 39
PCMS_FAIL 39
pcms_len.h 38
PCMS_OK 39
PcmsAttrDefInit 61
PcmsAttrGetLov 62
PcmsAttrValidate 65
PcmsCallbackStruct 40
PcmsChdAttachmentStruct 40
PcmsCheckMessages 66
PcmsClntApiConnect 144
PcmsClntApiDisconnect 145
PcmsClntApiExecCommand 146
PcmsClntApiFree 147
PcmsClntApiGetLastErro 148
PcmsClntApiGetLastErrorEx 149
PcmsClntApiModeBinary 150
PcmsClntApiModeText 151
PcmsClntApiSilentConnect 152
PcmsCntrlPlanGet 68
PcmsConnect 71
PcmsDisconnect 73
PcmsEventStruct 41
PcmsEvntCalloc 74
PcmsEvntCalloc() 53
PcmsEvntFree 75

PcmsEvntFree() 53
PcmsEvntMalloc 76
PcmsEvntMalloc() 53
PcmsEvntRealloc 77
PcmsEvntRealloc() 53
PcmsExecCommand 78
PcmsFullQuery 80
PcmsGetAttrDefNum 85
PcmsGetAttrFile 86
PcmsGetAttrs 88
PcmsGetBaseDbOptions 89
PcmsGetCandidates 90
PcmsGetCommandLine 91
PcmsGetConnectDesc 92
PcmsGetDimensionsVersions 93
PcmsGetPendingUsers 94
PcmsGetRSAttrs 96
PcmsGetRSNames 98
PcmsGetStringSymbolValue 100
PcmsGetSymbolInfo 101
PcmsGetUserRelTypes 103
PcmsGetUserRoles 105
PcmsGetWsetObj 107
PcmsInitSpec 108
PcmsInitUid 109
PcmsLcStruct 41
PcmsLovFree 110
PcmsObjAttrDefStruct 42
PcmsObjAttrStruct 43
PcmsObjFree 111
PcmsObjGetBackRels 112
PcmsObjGetRels 114
PcmsObjInSecondary 116
PcmsObjStruct 44
PcmsPendGet 117
PcmsPendingUserStruct 44
PcmsPendStruct 45
PcmsPendWhoGet 119
PcmsPopulate 121
PcmsQuery 122
PcmsRelStruct 45
PcmsRelTypeStruct 46
PcmsRoleStruct 46
PcmsSendCommand 126
PcmsSetAttrs 128
PcmsSetCallback 130
PcmsSetDbErrorCallback 132
PcmsSetDirectory 134
PcmsSetIdleChecker 135
PcmsSetWsetObj 136
PcmsTypeStruct 46
PcmsUserRoleStruct 47
PcmsWriteAttachments 137
populate an object's attributes values 121
pos predefined call 231
post-events 161
predefined call
  durules 230
  pos 231
  scanarea 232
  substring 233
pre-events 160
project ALF events
  build submitted and build completed 529, 532
  create project 535
  event extension 528
promoteBaselines 443
promoteItems 447
promoteRequests 450

R

re-allocate memory 77
relateItemsToParts 453
relateRequestToRequests 456
remote job execution templates
  asynchronous remote jobs 302
  introduction 300
  remote job script example 303
  synchronous templates 300
  templates 300
request ALF events
  action 536
  create 538
  deploy 541
  event extension 536
  update 542
RESTful API web services 549
  example JSON responses 553
  introduction 550
  reference 558
  WADL defintion file 555
results of function calls 40
retrieve
  attribute numbers in a role section 96
  candidates for delegation 90
  Dimensions CM objects inboxes for a user 117
  users for object 119
retrieve base database options 89
return codes 39
runCommand 463

S

scanarea predefined call 232
schedule job ALF events
  event extension 545
  run 546
Secondary Build Execution Monitor (SBEM) 276
set Dimensions CM API server callback 130
set Dimensions CM object attributes 128
set file transfer mode to ASCII 151

set file transfer mode to binary 150
set server error callback 132
set user's current project 136
simultaneous conflicting ENQs, controlling in data sets 290
SSL 328, 332
Structured Information Return (SIR) 246
submitDeployBaselines 465
submitDeployItems 468
submitDeployRequests 471
submitRollbackBaselines 474
submitRollbackItems 477
submitRollbackRequests 480
substring predefined call 233
system-defined attributes 47

T

technical support
  contacting 29
template processor interface 242
templating language
  complex symbol references 217
  controlsymbols
    DMCOMBINEOUTERR 207
    DMSAVESTDERR. 206
    DMSAVESTDOUT 206
  default template characters 187
  deployment area scripts
    attributes 315
    debugging 314
    example 317
    introduction 310
    variables 312
  Dimensions Build optional symbols
    DM_BUILDER_PROGRESS_REPORTING 215
    DM_SP_START_STAGE 215
    DMALTSERVER 215
    DMEXECENV 212
    DMEXPAND 213
    DMFINAL 214
    DMMAXRC 214
    DMMUSTRUN 214
    DMNOCACHE 214
    DMREBUILDALL 214
    DMTASKNAME 213
    DMTIMEOUT 213
    DMTLOG 215
  Dimensions Build predefined symbols
    DMSTEPMODE 214
  Dimensions Build standard symbols
    DMBUILDJOB 211
    DMINPUT 210
    DMPASS 211
    DMPATH 208
    DMPBEMNODE 211
    DMPBEMPORT 211
    DMRUNMODE 210
    DMSERVER 211
    DMSTEP 210
    DMTARGET 210
    DMTOKEN 211
    DMXFERSOURCE 211
  directives 188
)ADJUST 188
)ATTR 188
)CALL 230
)CALLBACK 189
)CM 189, 190
)DELETE 234, 235
)DUMP 191
)ELSE 191
)ENDEXPAND 193
)ENDIF 192
)ENDR 193
)ENDSCRIPT 230
)EXPAND 193
)IF 194
)IFDEF 194
)IFGROUPDEF 234, 235
)IFGROUPNDEF 234, 235
)IFNDEF 195
)IM 195
)LOAD 234, 235
)LOGON 196
)REP 196
)SAVE 234, 235
)SCRIPT 230
)SET 198
)SET_PATH 199
)SET_PATH_MVS 199
)SET_PATH_NT 200
)SET_PATH_UNIX 200
)SETL 198
)SETU 200
)SLICE 201
)VECTOR 201
  DMSAVESTDOUT 206
  inline functions 220
  introduction 186
  job sequence number symbols, on z/OS 216
  MDHDSN inline function 220
  MDHEXTRACT inline function 223
  Openmake symbols
    ANT_HOME 295
    DM_CURRENT_USERNAME 295
    DM_LICENSE 295
    DM_META_DATASET 295
    DM_META_LOCK_MAJOR 295
    DM_META_LOCK_MINOR 295
    DM_OM_OSPLATFORM 295

DM_OPENMAKE_INSTALL 295
DM_OPENMAKE_SERVER 295
DM_ROOT 295
DM_TPLB_MACHINE 295
DM_TPLB_PUBLIC 295
DMBLDLOGURL 296
DMBLDMAKE_OPTIONS 296
DMBLDPROJ 296
DMCDHLQ 296
DMCDILQ 296
DMCDLPA 296
DMCLEANBUILD 296
DMDAY 296
DMDIRECTORY 296
DMHOUR 296
DMINHLQ 296
DMINST 296
DMMACHINE 296
DMMICROSEC 296
DMMINUTE 296
DMMONTH 296
DMMVSUSR 296
DMOBOMRPT 296
DMOMDATETIME 296
DMOMINSTALL 296
DMOMSERVER 296
DMOPTIONS 296
DMOSPLATFORM 296
DMPBEMNODE 296
DMPBEMPORT 296
DMPUBLIC 296
DMRUNMODE 296
DMSEARCHPATH 296
DMSECOND 296
DMSECONDS 296
DMSEQ9 296
DMSTEP 296
DMSYSUSR 296
DMTARGET 296
DMUNIQUE 296
DMUSER 297
DMYEAR 297
JAVA_HOME 297
Openmake templates
introduction 294
variables 295
overview 186
predefined symbols
DMCERTIFICATE. 202
DMCURRENTDIRECTORY. 202
DMCYGWIN 202
DMDAY 202
DMFOLDERSEPARATOR 203
DMHOUR 203
DMMICROSEC 203
DMMINUTE 203
DMMONTH 203
DMOSTDERR 203
DMOSTDOUT 204
DMOXTMPLT 204
DMPASSWORD 204
DMPATHSEPARATOR 204
DMSECOND 204
DMUNIQUE 205
DMUSER 205
DMUSERU 205
DMYEAR 205
predefined z/OS symbols
DMSEQ9 216
DMSEQ99 216
DMSEQ999 216
DMSEQA 216
DMSEQAA 216
DMSEQAAA 216
DMSEQAAAA 216
DMSEQNNN 216
DMSYSDIR 216
DMSYSGID 216
DMSYSSHL 216
DMSYSUID 216
DMSYSUSR 216
remote job execution predefined symbols
DMCERTIFICATE 207
DMJOBID 207
remote job execution standard symbols
DMJOBID 211
remote job execution templates
asynchronous remote jobs 302
introduction 300
remote job script example 303
synchronous templates 300
templates 300
standard symbols
DMUSER 211
templates
about 186
expanding 217
testing 222

U

unrelateItemsFromParts 483
unrelateRequestFromRequests 486
updateRequest 492
updateRequestDescription 495
updateWorkArea 497
user ID predefined template symbols 216
user written template processing
call out functionality 229
passing data between steps 234
user-defined attributes 47
UTF-8 332

V

validate an attribute value 65
validate events 160
videos 29

W

Web services
actionRequest 347
Apache Axis2 322
architecture 322
authentication 328
buildBaseline 349
buildProject 352
checkInItem 356
checkOutItem 360
createDesignPart 367
createDesignPartBaseline 369
createProject 373
createProjectBaseline 377
createRequest 380
createRevisedBaseline 383
createSceduleJob 386
dateTime format
attribute values 333
general 332
SOAP/XML 332
delegateRequest 394
delegateRequestForReplication 396
demoteBaselines 398
demoteItems 401
demoteRequests 404
deployBaseline 407
deployRequest 409
document/literal style WSDL 322
error handling
batch commands 326
single command 326
getItems 411
getRequestDescription 415
getRequests 417
getVersion 422
licensing 328
list of 322
listProjectItems 426
listProjectRequests 429
listProjects 431
listScheduleJobs 434
logging mechanism 324
moveItemToPart 441
promoteBaselines 443
promoteItems 447
promoteRequests 450
relateItemsToParts 453
relateRequestToRequests 456
runCommand 463
security 328
setting idle connection time-out 330
setting up development environment 328
SOAP 322
submitDeployBaselines 465
submitDeployItems 468
submitDeployRequests 471
submitRollbackBaselines 474
submitRollbackItems 477
submitRollbackRequests 480
system requirements 328
terminology 322
unrelateItemsFromParts 483
unrelateRequestFromRequests 486
updateRequest 492
updateRequestDescription 495
updateWorkArea 497
WSDL 322
WS-I 322
WS-I basic profile 322
wild cards, in build templates 256
windows.h 142
write specified request attachments to disk 137
WSDL 323