MICRO FOCUS | SERENA

Dimensions CM

Command-Line Reference

Copyright © 2017 Serena Software, Inc., a Micro Focus company. 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., 2345 NW Amberbrook Drive, Suite 200, Hillsboro, OR 97006.

Product version: 14.4

Publication date: December 2017

Table of Contents

Chapter 1 **Using the Dimensions Command-Line Interface 11**

About the Command-Line Interface 12
z/OS Limitations 12
Typographical Conventions 13
Command Syntax 13
Command Syntax Diagram 13
Using Quoted Strings in Dimensions Commands 15
The Escape Character 15
Using Comments in Dimensions Command Files 17
Multivalue and Multiline Attributes 17
Compound Fields in Dimensions Commands 17
Error Handling with Multiple Commands 18
Running Commands from a Dimensions Client 19
HTTP/S Network Protocol 19
Connecting from Windows Dimensions Clients 19
Connection Processes for UNIX Dimensions Clients 22
Examples of Client Commands 24
Important Considerations for Item Commands 24
Selecting Item Revisions 24
Updating the Content of an Item Without Changing the Item Revision 25
Assigning Default Project and Working Location 25
Assigning the Default Project 26
Assigning the Working Location 26
Requirements for Request Attributes 26
Using Certificates 27
Example 27
Operating System Differences 28
Spaces in File Names 28
Windows UNC Paths 28
Specifying a Project File Name in z/OS Item Operations 28
Command-Line Logging and Usage Analysis 28
Audit Trail of Commands 29
Logging All Commands Run by All Users 29
Logging Users Who Connect to Dimensions 30
Invoking Help at the Dimensions Command-Line 31

Chapter 2 **Command Reference 33**

ABL – Action Baseline or Items 34
AC – Action Request 36
ACDI – Action Request Items 38
ACDWS – Add Request Items to Project 39
ACF – Assign Data Formats to Request Types 41

ADF – Assign Data Formats to Item Types 42
AGRPU – Assign Groups to a User 43
AI – Action Item 44
AIWS – Add Item Revision to Project 46
ANNOTATE - Display File Annotations 49
APNO – Allocate Part Numbers 51
AUDIT – Audit Area 52
AUGRP – Assign Users to a Group 54
AUPG - Start Upgrade Process 55
AUR – Assign User Roles 56
AUTH – Authorize Access to Node 60
AWS – Action Project 61
BC – Browse or Print Request 62
BI – Browse Item 64
BLD – Build 66
BLDB – Build Baseline 72
CA – Create Area 76
CAR – Create Archive 79
CBA – Create Build Area 80
CBDB – Register a Base Database Entry 81
CBL – Create Baseline 82
CBP – Copy Build Project 88
CC – Create Request 90
CCO – Create a New Contact 93
CCS – Create Credential Set 94
CCST – Create a New Codeset 95
CCU – Create Customer 96
CFS – Create a File System 97
CGRP – Create Group 98
CHMOD – Change File Permissions 99
CI – Create Item 100
CINS – Register a Database Instance Entry 105
CIP - Create Installation Package 106
CIU – Cancel Item Update 108
CLCA – Create Library Cache Area 110
CLEAN – Clean Deployment Area 112
CMB – Create Merged Baseline 113
CMD – Execute Dimensions Command File 116
CMP – Compare Structures or Baselines 117
CNC – Create a Network Node Connection 118
CNDO – Create a Node Object 119
CNN – Create a Network Node 120
CNSJ – Cancel Schedule Job Execution 121
CNWO – Create a Network Object 122
COS – Create an Operating System 123
CP – Create Design Part 124
CPV – Create Design Part Variant 125
CRB – Create Revised Baseline 126

CRSD – Create a Resident Software Definition 134
CS – Create a Stream 135
CSJ – Create Schedule Job 138
CUSR – Register User 139
CVS – Create Variant Structure 140
CWSD – Create Project Directory 142
DAR – Delete Archive 143
DBC – Delete Build Configuration 144
DBDB – Unregister an Existing Base Database Entry 145
DBL – Delete Baseline 146
DBPROJ – Create a Dimensions Build Project 147
DBT – Deliver Build Targets 148
DCH – Delete Request 149
DCO – Delete an Existing Contact 150
DCS – Delete Credential Set 151
DCST – Delete an Existing Codeset 152
DDF – Define Data Formats 153
DELIVER – Deliver to a Stream 155
DFS – Delete an Existing File System 160
DGRP – Delete Group 161
DI – Delete Item 162
DINS – Unregister an Existing Database Instance Entry 164
DIR – Define Item Relations 165
DLCA - Download to Library Cache Area 166
DLGC – Delegate Request 168
DLGI – Delegate Item 170
DLGS – Delegate Personal Stream 172
DMBL – Demote Baseline 173
DMI – Demote Item 176
DMRQ – Demote Request 178
DNC – Delete an Existing Network Node Connection 180
DNDO – Delete an Existing Network Node Object 181
DNN – Delete an Existing Network Node 182
DNP – Define New Product 183
DNWO – Delete an Existing Network Object 185
DOS – Delete an Existing Operating System 186
DOWNLOAD – Download Project or Baseline 187
DPB – Deploy Baseline 192
DPI – Deploy Item 194
DPL – Define Product Libraries 197
DPR – Deploy Request 200
DPROJ – Define a Dimensions Project 202
DPRP – Define Preservation Rules Policy 203
DPV – Delete Design Part Variant 207
DREL – Delete Release 208
DRSD – Delete an Existing Resident Software Definition 209
DS – Delete Stream 210
DSJ – Delete Schedule Job 211

DUR – Define User Roles. 212
DUSR – Unregister User 213
DVB – Define Version Branch 214
DWP – Delete Whole Product. 215
DWS – Define New Project 216
DWSD – Delete Project Directory 220
ECDI – Extract (Check Out) Request Items 222
ECFG – Extract (Check Out) Build Configuration 224
EI – Extract (Check Out) Item for Update 225
ESJ – Edit Schedule Job 230
EXIT – End Dimensions Execution 231
EXPORT – Export Build Configuration 232
FBI – Fetch (Get) Baseline Items 235
FCDI – Fetch (Get) Request Items 238
FI – Fetch (Get) Item 240
FIF – Find Item File 245
FRC – Forward a Release to a Customer 247
FWI – Fetch (Get) Project Items 248
GENCERT - Generate Certificates 251
GREP – Search and Replace 252
HELP – Help 255
HIDE - Hide Unused Streams and Projects 256
IMPORT – Import Build Configuration 257
LA – List Areas 258
LAST - List Area Structure. 260
LAVC – List Deployment Area Versions 261
LBA – List Build Areas 263
LBDB – List Existing Base Database Entries. 264
LBPROJ – List Dimensions Build Projects. 265
LCK – Lock Project 266
LCO – List Existing Contacts 267
LCS – List Credential Set 268
LCST – List Existing Codesets 269
LFS – List Existing File Systems 270
LGRP – List Groups 271
LII – List Item Build Relationships 272
LINS – List Existing Database Instance Entries 273
LLCA – List Library Cache Areas. 274
LMNR – List Mail Notification Rules. 275
LNC – List Existing Network Node Connections 276
LNDO – List Existing Network Node Objects 277
LNN – List Existing Network Nodes. 278
LNWO – List Existing Network Objects 279
LOG - Lists Stream or Project Changeset History 280
LOS – List Existing Operating Systems 283
LPRIV – List Privileges 284
LPROJ – List Dimensions Projects and Build Projects 285
LPRP – List Preservation Rules Policies 286

LPRT – List Existing Network Protocols 287
LPSP – List Per-Stage Project Properties 288
LRC - List Request Changes 289
LRSD – List Existing Resident Software Definitions 290
LSAR – List Archives 291
LSBL – List Baselines 292
LSJ – List Scheduled Jobs 293
LSTG – List Stages 295
LUPG - List Upgrade History 296
LWC – List Project Conflicts 297
LWS – List Projects 298
LWSD – List Project Directories 300
MCPC – Move Request To Primary (Main) Catalog 302
MCSC – Move Request To Secondary Catalog 303
MDR – Move Design Part Relationship 304
MERGE - Merge into Stream Work Area 305
MI – Merge Item Revisions 310
MIP – Move Item to Another Part 313
MIT – Move (Change) Item Type 315
MVC – Move Request 317
MWS – Merge Projects 319
MWSD – Move Project/Stream Directory 321
PA – Populate Areas 322
PBA – Populate Build Area 323
PEND – Update Users' Pending Request Lists 324
PEND – Update Users' Pending Item Lists 326
PEND – Update Users' Pending Baseline Lists 328
PEND – Update Users' Pending Project Lists 330
PLCA – Purge Library Cache Area 331
PMBL – Promote Baseline 332
Parameters and Qualifiers 332
PMI – Promote Item 334
PMRQ – Promote Request 336
PRIV – Manage Privileges 338
QUIT – Quit 340
RA – Remove Area 341
RABC – Relate Area to Build Configuration 342
RAI – Remove Archived Item 343
RAMA – Remove Archived Material Selected by Archive 344
RAMP – Remove Archived Material Selected by Product 345
RAT – Read Archive Tape 346
RAWS – Relate Area to Project 347
RBA – Remove Build Area 349
RBBL – Relate Baseline to Baseline 350
RBCD – Relate Baselines to Requests 351
RBPROJ – Delete a Dimensions Build Project 352
RBWS – Relate Baseline to Project 353
RCCD – Relate Requests to Request 354

RCDI – Return Request ID 355
RCDWS – Remove Request Items from Project 357
RCFG – Return (Check In) Build Configuration 358
RCI – Report Current Items 359
RCP – Report Current Parts 361
RCSJ – Relate Command to Schedule Job 363
RCU – Remove Customer 364
RDEL – Delete Jobs from the Job Queue 365
RDS – Report Design Structure 367
REL – Release 370
RENAME – Rename a Product, Baseline, Design Part, Project, or Item 373
REQC – Request Dimensions Request 375
REXEC – Execute a Job on a Network Node 376
RI – Return (Check In) Item 378
RICD – Relate Item to Requests 381
RII – Relate Item to Item 383
RIP – Relate Item to Part 385
RIR – Remove Item Relation Definition 386
RIWS – Remove Item Revision from Project 387
RLCA – Remove Library Cache Area 389
RLIST – Lists Jobs in the Job Queue 390
RMDF – Remove Data Formats 393
RMVB – Remove Version Branch 394
ROA – Retrieve Offline Archive 395
RP – Relate Design Part 396
RPCD – Relate Part to Requests 397
RPCP – Report Product Control Plan (Process Model) 398
RPNO – Report Part Numbers 399
RPROJ – Remove a Dimensions Project 400
RPT – Report Requests 401
RPT – Baseline Detail Report 405
RRCD – Relate Requirement to Request 407
RREG – Reassign User Registration 409
RSJ – Run Schedule Job 410
RSTAT – Update Job Status 411
RUR – Run User-Defined Report 412
RVDA – Remove VDA Records 414
RWCD – Relate Project to Request 415
RWS – Remove Project 416
RWWS – Relate Project to Project 417
SAVE – Save to Persistent Symbol Table 418
SCWS – Set Current Project 419
SDF – Set Data Format Flags 423
SDPBL – Submit Deploy Baseline 425
SDPI – Submit Deploy Item 426
SDPRQ – Submit Deploy Request 427
SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment 428

SF - Set Favorites 431
SHELVE - Shelve Changes to a Personal Stream 432
SHOW - Show Hidden Streams and Projects 436
SI – Suspend Item 437
SPSP – Set Per-Stage Preservation Policy 439
SPV – Suspend Design Part Variant 440
SRAV – Submit Rollback Area Version 441
SSPM – Display Values in Symbol Tables 442
SUB – Subscribe to Notification Rule 443
SVBF – Set Version Branch Flags 444
SWF – Set Project File Name 446
SWS – Set Project/Stream Attributes 448
SWSP – Set Project Permissions 451
TBI – Transfer Baseline In 452
TBO – Transfer Baseline Out 453
UA – Update Area 454
UBA – Update Build Area 457
UBDB – Update an Existing Base Database Entry 458
UBLA – Update Baseline Attributes 459
UBPROJ – Update a Dimensions Build Project 460
UC – Update Request 461
UCM – Update Code Metrics 465
UCO – Update an Existing Contact 467
UCS – Update Credential Set 468
UCSJ – Unrelate Command from Schedule Job 469
UCST – Update an Existing Codeset 470
UCU – Update Customer 471
UFS – Edit an Existing File System 473
UGRP – Update Group 474
UI – Revise Item 475
UIA – Update Item Attributes 479
UINS – Update an Existing Database Instance Entry 483
ULCA – Update Library Cache Area 484
ULCK – Unlock Project 486
UNC – Update an Existing Network Node Connection 487
UNDO - Undo a Delivery 488
UNN – Update an Existing Network Node 490
UNWO – Update an Existing Network Object 491
UOS – Update an Existing Operating System 492
UP – Update Design Part PCS 493
UPA – Update Part Attributes 494
UPDATE – Update Work Area 496
UPLOAD – Upload Local File or Directory 505
UPNO – Update Part Numbers 510
UPROD - Update a Dimensions Product 511
UPROJ – Update a Dimensions Project 512
UREG – Register User 513
URP – Unrelate Design Part 514

URSD – Update an Existing Resident Software Definition 515
USUB – Unsubscribe from Notification Rule 516
UUA – Update User Attributes 517
UWA – Update Project or Stream Attributes 518
VLSJ – View Log of Schedule Job Execution 521
WRC – Withdraw a Release from a Customer 522
XABC -Remove Area from Build Configuration 523
XAWS – Unrelate Area from Project 524
XBBL – Unrelate Baseline from Baseline 525
XBCD – Unrelate Baselines from Requests 526
XBWS – Unrelate Baseline from Project 527
XCCD – Unrelate Requests from Request 528
XICD – Unrelate Item from Requests 529
XII – Unrelate Item from Item 531
XIP – Unrelate Item from Part 532
XPCD – Unrelate Part from Requests 534
XRCD – Unrelate Requirement from Request 535
XREG – Unregister User 537
XWCD – Unrelate Project from Request 538
XWWS – Unrelate Project from Project 539

Chapter 3 **Standalone Dimensions Utilities 541**

Introduction 542
General Information 543
Case Translation 543
Wild Card Characters 543
Execution Authority: Change-Manager or Tool-Manager 543
Metadata Utility 544
Overview 544
Syntax 544
Actioning Requests by Date or Attribute Value 551
Sending Reminders of Pending Lists 552
Automatic Job Triggering: Using crontab 553
Encryption 554
Syntax 554
Examples 555
PRCS–RCS-like Front End to Dimensions 556
Subcommands 557
Dimensions Options 561
Description 562
Revisions 563
Dimensions Named Branches 564
Environment 564
prcs Dimensions Files 564
Constraints 564
PSCCS–SCCS-like Front End to Dimensions 565
Subcommands 566
Dimensions Options 570

Description 571
Revisions 572
Dimensions Named Branches 572
psccs Dimensions Files 572
Constraints 572

Chapter 4 **The Developer Command-Line Interface 575**
Introduction 576
What Can I Do with the Developer Command-Line? 576
How Do I Use the Developer Command-Line Interface? 576
How do I Invoke the Developer Command-Line Interface? 577
How can I Display Help Information? 577
How do I Connect to the Database? 577
Working with DM – Some Typical Development Scenarios 578
Creating and Deleting Streams 579
Importing Your Code Into the Stream 579
Obtaining a Working Copy of the Code for Modification 579
Making Changes and Committing them Back to the Repository 580
Handling Conflicts 581
Using Requests to Control Change Sets 584
What are the Commands I Can Perform? 586
Managing Streams 586
Working with Streams 586
Alphabetic List of Commands 587
Command List 589
add – Schedule a file or directory to be added to the repository 589
annotate – Add a comment to a file 591
cat – Display the contents of a file 592
commit – Commit content to a stream 593
createstream – Create a new stream in the repository 595
delete – Schedule deletions 597
deletestream – Delete a stream 598
deliver – Deliver content to a stream 599
diff – Display the code differences between a stream and a local work area 602
export – Exports a non-versioned copy of a stream to a work area . . 604
get – Get the contents of a stream to a local work area 606
getinfo – Get current stream and work area details 608
import – Import uncontrolled content into a stream 609
list – List the contents of a stream 611
listbaselines – List the baselines in the repository 612
liststreams – List the streams in the repository 614
lockfile – Lock a file in the repository 616
lockstream – Lock a stream 617
log – Display the repository history for a file 618
logout – Clear the Dimensions login credentials 619
move – Move (rename) files 620
revert – Revert local changes made to a work area 621

status – Report on changes to a local work area 623
switchstream – Switch the working stream. 625
unlockfile – Unlock a file in the repository 626
unlockstream – Unlock a stream 627
update – Update a local work area 628
Configuring the Developer Command-Line 631

Index. **633**

Chapter 1

Using the Dimensions Command-Line Interface

About the Command-Line Interface	12
Command Syntax	13
Running Commands from a Dimensions Client	19
Important Considerations for Item Commands	24
Assigning Default Project and Working Location	25
Requirements for Request Attributes	26
Using Certificates	27
Operating System Differences	28
Command-Line Logging and Usage Analysis	28
Invoking Help at the Dimensions Command-Line	31

About the Command-Line Interface

The Dimensions® CM command-line interface (available for the majority, but not all, Dimensions functions) is an efficient alternative to Dimensions GUI-based clients, **provided that you are familiar with Dimensions and the product**. Command mode is particularly suited for situations where you wish to perform unattended bulk batch operations.

IMPORTANT! Command mode can be used in any of several ways, on either a Dimensions server or a Dimensions client. In all cases you must first log in to Dimensions—see "Running Commands from a Dimensions Client" on page 19 for details of Dimensions Client command-line operation/connection.

CAUTION! Use of shell scripts or other forms of scripting external to Dimensions to perform refactoring operations on code or other files under source control without invoking the command-line interface causes unexpected behavior and is not supported.

z/OS Limitations

NOTE The term z/OS in this manual covers both the z/OS 1.1 (or later) and OS/390 V2R10 (or later) operating systems.

Several of the mechanisms detailed below are not supported by Dimensions for z/OS. Refer to the *Serena Dimensions for z/OS Users and Administrator's Guide* for further details.

- A single command may be preceded by DMCLI and entered at the operating system prompt.

NOTE Supported from USS for z/OS with some restrictions for credentials. Supported from MVS batch via DIM390B.

- A single command may be entered at the `Dimensions>` prompt that results from typing DMCLI at the operating system prompt.

NOTE Supported from USS for z/OS with some restrictions for credentials.

- A command may be placed on a line or several consecutive lines of a text file (see below), which is specified as the parameter in an CMD command. The file may contain any number of commands to be processed sequentially in a single batch job. However, the interactive functions (BI, and some uses of BC and UC) cannot be included.
- A single command may be entered at the Execute Command window from the Dimensions desktop client's Run interface.

Typographical Conventions

The following typographical conventions are used in this manual. These typographical conventions are used to assist you when using the documentation; they are not meant to contradict or change any standard use of typographical conventions in the various product components or the host operating system.

italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
bold	Emphasizes important information and field names.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospace italics`	Indicates names that are placeholders for values you specify; for example, *`fileName`*.
`monospace bold`	Indicates the results of an executed command.
vertical bar `|`	Separates menus and their associated commands. For example, select File | Copy means to select Copy from the File menu. Also, indicates mutually exclusive choices in a command syntax line.
angle brackets `<>`	Indicates names that are placeholders for values that you specify; for example, `<file-name>`.
square brackets `[]`	Indicates optional items. For example, in the following statement: `SELECT [DISTINCT]`, DISTINCT is an optional keyword.
`. . .`	Indicates command arguments that can have more than one value.

Command Syntax

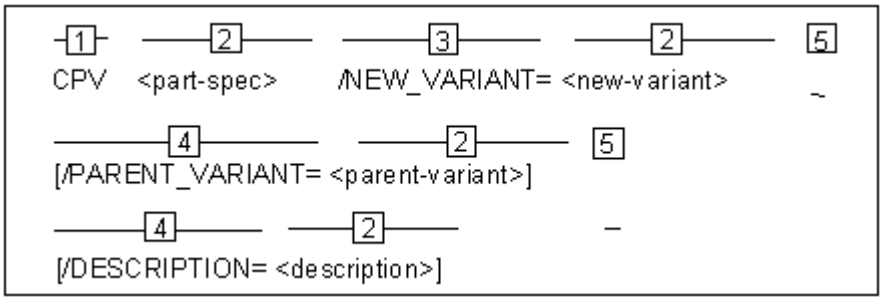
The following topics provide essential information on the format and usage of Dimensions command-line commands.

Command Syntax Diagram

A command consists of a Dimensions command, followed by parameters and qualifiers. The following is an example of the CPV (create design part variant) command:

```
CPV SOMEPROD:"RELEASE MANAGEMENT".AAAA /NEW_VAR=IBM -
/DESC="Release Support - IBM Version"
```

The basis for coding each command is the **syntax diagram**, and there is a different one for each mnemonic. This is the syntax diagram for CPV:



The meaning of each part of the diagram is as follows:

1 The *command* identifies the Dimensions command to be performed.

2 A parameter indicates where a variable value is to be substituted. Parameters that end in `-spec` denote compound fields, and the syntax for coding all the components of these is specified below in "Compound Fields in Dimensions Commands" on page 17.

An *ellipsis (…)* indicates a list of any number of parameters, separated by commas and enclosed in parentheses, for example:

```
/CHANGE_DOC_IDS=(<request1>,<request2>,…)
```

If there is only one parameter in the list, the parentheses are not necessary, for example:

```
/CHANGE=PROD_DR_25.
```

After each comma separating the parameters in the list, one or more spaces are optional before the next parameter. Along with the spaces, if required, a continuation character can be included and a new line begun.

3 A *required qualifier* is coded as shown. It always begins with a forward slash (**/**) and usually ends with an equal sign (**=**), the latter indicating that a substituted parameter variable must follow. (A qualifier, which does not end with **=**, is complete in itself.)

4 An *optional qualifier* is enclosed in square brackets ([ ]), and may be omitted in certain circumstances (as detailed in this reference). The square brackets themselves are never included in the Dimensions command.

All qualifiers (required and optional) may be abbreviated provided that no ambiguity is caused. Options are shown on consecutive lines that have the same indentation, with an underscored **or** at the start of the lower line. Only one of the lines so designated may be chosen.

5 A *continuation indicator* is shown as hyphen (-). This is the character normally used at the end of a line to indicate that a command is being continued on another line.

There must be at least one space between the last command character and the hyphen (or backslash), but there must not be any spaces between the hyphen and the end of the line.

Exceptions:

- *UNIX systems* (if the command is being entered at the UNIX system prompt): a backslash (\ ) must be used instead of a hyphen to indicate continuation.
- *Windows system* (if the command is being entered at the operating system prompt): there is no continuation available, the command must be entered on a single line and is limited to 256 characters maximum.

Using Quoted Strings in Dimensions Commands

If you want to include spaces or any non-standard characters in a parameter variable, you must use enclose the part of the command that contains the non-standard in double-quotation characters (" "), for example:

"RELEASE MANAGEMENT"

There are additional conventions that you must follow if you want to enter a Dimensions command that includes a quoted string at the operating system prompt.

Windows System Prompt

The syntax is identical to that used at the Dimensions prompt (except that no continuation line is available).

IMPORTANT! When you execute a command using the `-cmd` parameter of `dmcli`, each double quotation mark used for wrapping strings that contain spaces must be escaped with a backslash (\).

UNIX System Prompt

The double-quoted string must itself be enclosed in single-quotation characters (' '), for example

```
'PROD:"QUERY RELEASE".AAAA-SRC;2'
```

NOTE This alternative syntax is not shown in the remainder of this reference. **You must understand implicitly that it is required whenever the command is used in this way**.

The Escape Character

NOTE The escape character discussed here does not refer to the Esc key on your keyboard.

An escape character must precede any character in a parameter that should not be interpreted as part of the syntax of the command. Such characters may include:

'at'	@	Double quotation	"
Comma	,	Single quotation	'
Left parentheses	(	Forward slash	/
Right parentheses	)	Backslash	\
Newline	@n		

The default escape character is **@**, but the setting of the Dimensions symbol DM_ESCAPE_CHAR may be used to specify any alternative as the escape character. The Windows command-line interface escape character is a backslash (****) and this **cannot** be changed.

For example, in UNIX, to set the attribute TITLE to:

```
The "at" symbol (@)
```

the command would be:

```
UC PROD_DC_17 -
/ATTR=(TITLE="The @"at@" symbol @(@@@)")
```

The same command submitted using *dmcli* from the Windows operating system prompt would be:

```
dmcli -cmd "UC PROD_DC_17 -
/ATTR=(TITLE=\"The @\"at@\" symbol @(@@@)\")"
```

A Perl script on Windows might have this:

```
my $command_string = "dmcli -cmd \"UC PROD_DC_17
     /ATTR=(TITLE=\\\"The \@\\\"at\@\\\" symbol \@(\@\@\@)\\\")\"";
```

!

Note that in a Windows batch file, you cannot include double quotation marks in a variable definition if the variable will be used in a command where quotation marks need to be escaped. For example, the following does not work:

```
set FILE1="C:\<dir-name-with-spaces>\<file-name>"
...
call dmcli ... %FILE1%
```

Instead, do it like this:

```
set FILE1=C:\<dir-name-with-spaces>\<file-name>
...
call dmcli ... \"%FILE1%\"
```

An example how to use the "@n" syntax for escaping newlines is given in the discussion of multiline attributes below.

Using Comments in Dimensions Command Files

As discussed on page 12, a number of Dimensions commands can be batched together in a text file that is used as input to the Dimensions CMD command (see "CMD – Execute Dimensions Command File" on page 116).

To aid readability of this text file, you can enter comment lines. You do this by putting an exclamation point as the first non-blank character of a line where a command could start—this means that you *cannot* place the exclamation point in the middle of a set of continuation lines for a single Dimensions command.

Multivalue and Multiline Attributes

A multivalue attribute—such as *OPS* with valid values *Sun*, *HP*, *DEC* and *IBM*—would be handled in command mode as follows:

```
/ATTR=(OPS=["Sun","HP","DEC","IBM"])
```

A multiline attribute – such as *DOC* with value

```
Hello world
    First line
    Second line
```

would be handled in command mode as follows:

```
/ATTR=(DOC="Hello world@nFirst line@nSecond line")
```

Compound Fields in Dimensions Commands

A compound field is a parameter that is defined as a set of multiple values or fields, in a specific syntax format. There are five compound fields defined in the syntax diagram (see "Command Syntax Diagram" on page 13). They are:

<project-spec>, <part-spec>, <item-spec>, <baseline-spec> and <release-spec>

See the following for details on the syntax of each field.

<project-spec>

Identifies a specific project and has the following syntax:

```
<product-id>:<project-id>
```

<part-spec>

Identifies a specific design part and has the following syntax:

```
<product-id>:<part-id>.<variant>;<pcs>
```

<item-spec>

Identifies a specific item and has the following syntax:

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

The <revision> field can optionally have the syntax:

```
<branch-id>#<version>
```

where <branch-id> identifies the development branch to which this item revision belongs, and <version> identifies its revision within this branch. For example:

```
PROD:"QUERY RELEASE".AAAA;maint#3
```

If the field is **not** of the above form (i.e. it does not contain the **#** character), then the entire field is the revision number, and the item revision is not in a named branch.

<baseline-spec>

Identifies a specific baseline and has the following syntax:

```
<product-id>:<baseline-id>
```

<release-spec>

Identifies a specific release and has the following syntax:

```
<product-id>:<release-id>
```

Examples

An example of <part-spec>, omitting <variant> and <pcs>:

```
PROD:"RELEASE MANAGEMENT"
```

An example of <item-spec>, omitting <item-id> and <variant>:

```
PROD:-SRC;1
```

Special Considerations

In certain circumstances, it is possible to omit some fields when coding compound parameters. When doing so, follow these rules:

- The <product-id> and the colon (:) following it can never be omitted.
- Apart from <item-id>, which can be optional, the second field (<part-id>, <baseline-id> or <release-id>) is also always required.
- If the <item-id> field is omitted, no other punctuation is omitted with it.
- If any other field is omitted, the immediately preceding punctuation character is also omitted, e.g. dot (**.**) when omitting <variant>; hyphen (–) when omitting <item-type>; and semicolon (;) when omitting <pcs> or <revision>.

Error Handling with Multiple Commands

You can optionally include the capability to stop processing a sequence of commands in a script by including a -stop parameter in the commands. When you include this parameter, the command-line interface will return an error and stop running if any of the commands in the sequence fails. For example, if a script includes three commands and the second command fails, then the script will stop running at the second command and the third command will not be processed.

Running Commands from a Dimensions Client

You can run Dimensions command-line commands from a Dimensions client or server. .

To run the Dimensions command-line from a Dimensions client, you must establish a network connection.

HTTP/S Network Protocol

For security reasons you may be required to login using the HTTP/S network protocol instead of the default Standard Dimensions Protocol (SDP). The Dimensions CM HTTP Connector allows a connection to a server using HTTP/S, for details see the *System Administration Guide*.

Connecting from Windows Dimensions Clients

There are a few methods that you can use to connect to a Dimensions server from a Windows client, in order to get started using the command-line interface.

Connecting Using the -con Command-Line Parameter

If you assign (or have already assigned) a "Previous Connections" name to the connection details in a remote login dialog box (for example from the desktop client or Windows Explorer integration), then you can log in directly using that connection, by entering the following command. This is the recommended connection method.

```
dmcli -con <connect-name> -user <user-id> -pass <password>
```

Syntax

```
dmcli
   [-con <connect-name>]
   -user <user-id>
   -pass <pswd>
   -card
   -host <server>
   -dbname <db-id>
   -dsn <dsn-name>
   -param <param-file>
   -file <cmd-file>
   -cmd <dm-cmd>
   -help
   -version
```

`-con <connectname>`

Specifies either:

- An existing connection string associated with stored connection parameters (created either by an earlier invocation of this command or by use of the "Previous Connections" field in the remote login dialog box). In this case, you would normally only use the `-con` and `-pass` parameters.
- A connection string to be created at this invocation of DMCLI. In this case, you either specify the connection parameters required on the same command line

  ```
  dmcli -con <new-connect-name> -user <user-name> …
  ```

  or enter

```
dmcli -con <new-connect-name>
```

After which you are prompted for the relevant connection details. This connection is then saved to file and will be used the next time `–con <new-connect-name>` is specified, when you will be prompted only for your password.

NOTE Unless `-con` is specified, none of the other DMCLI parameters will be stored for future use.

`-user <user-id>` specifies the operating system user name of your account on the server.

`-pass <pswd>` specifies the password of your operating system user name account on the server.

`-card` specifies that you are using Smart Card authentication (instead of your user id and password). See "Connecting Using Smart Card Authentication" on page 21.

`-host <server>` specifies the name of the server and optionally the port number to connect to:

- SDP protocol: `<server name[<:port>]`
- HTTP protocol: `http://<server name[<:port>]`
- HTTPS protocol: `https://<server name[<:port>]`

`-dbname <db-id>` specifies the database identifier.

`-dsn <dsn-name>` specifies the data source name for connecting to your remote database.

`-param <param-file>` specifies a file containing the above parameters. The file must be specified using the full directory path contained within double quotation characters ("). This file has a format similar to the following example:

```
-user dmsys
-pass xxx
-host server1
-dbname intermediate
-dsn PC50
```

`-file <cmd-file>` specifies a file containing several Dimensions commands to be executed. The file must be specified using the full directory path contained within double quotation characters (").

`-cmd <dm-cmd>` specifies a single Dimensions command to be executed.

`-help` displays command-line help.

`-version` displays the Dimensions release version.

Further Detail The parameter file or connection parameters can be followed by a Dimensions command using the `-cmd` option or in a command file using the `-file` option. If no command or command file is specified, then commands are read and executed from standard input until an EOF (CTRL+Z) character is detected, or the pseudo-command `exit` is encountered.

NOTE

- If the command you are running contains double quotation marks in the qualifiers, you must wrap the entire command in double quotations, and 'escape' the double quotation marks in the command. For example:

```
dmcli -user dmsys -pass dmsys -host myhost -dbname intermediate -dsn
mydsn -cmd "CI \"PAYROLL:LICENSE2 DAT TXT.A-SRC;1\" /
USER_FILENAME=G:\license.dat.txt /FILENAME=license2-dat-01.txt /
PART=PAYROLL:PAYROLL.A;1 /WS_FILENAME=license2.dat.txt /
DESCRIPTION=\"test test\" /FORMAT=TXT /
ATTRIBUTES=(COMPLEXITY=lowish) /COMMENT=\"This is a test\" /
CHANGE_DOC=(\"PAYROLL_CR_21\") /KEEP"
```

- Dimensions commands requiring quotation characters within the double quotations characters referred to above will require:
  - For Windows: three double quotation characters before the quoted string and three double quotation characters after the quoted string. For example:

```
dmcli -con _tabuilder -cmd "EI """TA_DESKTOP:TEST TXT.BASE-
    SOURCE_INT""" /USER_FILENAME="""c:\temp\test.txt"""
    /WORKSET=TA_DESKTOP:INTERNAL /NOOVERWRITE"
```

  - For UNIX: single quotation characters before the command containing the quoted string. For example:

```
dmcli -con _tabuilder -cmd 'EI "TA_DESKTOP:TEST TXT.BASE-
    SOURCE_INT"' /USER_FILENAME='"/usr/temp/test.txt"'
    /WORKSET=TA_DESKTOP:INTERNAL /NOOVERWRITE"
```

Connecting Using Smart Card Authentication

If your client has been configured to allow Smart Card authentication, you can use this feature to log in by specifying the -card parameter. For example, entering the command:

```
dmcli -dbname cm_typical-dsn dim12 -card
```

will result in you being prompted for your Smart Card PIN (if you have not previously entered it). On entering your PIN, you will be presented with dialog box showing a list of certificates and asked to choose one of them. The log in details will then be passed to the server and you will be connected.

Note when running a script, you will need to supply your username and password.

Connecting Using the Remote Login Dialog Box

To connect to the Dimensions server using the Remote Login dialog box, enter the following command at a Windows command prompt:

```
dmcli
```

The Remote Login dialiog box appears, and you can enter your connection information. Once the connection is established, the following prompt appears:

```
Dimensions>
```

You can now enter commands on the Dimensions client.

NOTE The Remote Login dialog box will also appear when other connection methods fail to establish a connection.

Connecting from a Server Using the DMDB Environment Variable

When running Dimensions operations from the command-line interface you will normally be required to set the DMDB variable, unless you access the command line through the Dimensions GUI login dialog in which case it will be set for you. The DMDB variable has to be set to the value:

```
<base_database_id>@<db_connection_string>
```

For example, in Dimensions for Windows:

```
set DMDB=intermediate@dim12
```

NOTE If you have installed the server on a Widows 64-bit machine and subsequently installed the client on that machine, you may need to set the path in order to access the correct 64-bit version of dmcli, as the client install will be referencing the 32-bit version. To do this, for example, perform the following command:

```
C:\>set PATH=C:\Program Files\Micro Focus\Dimensions
    14.4\CM\prog;%PATH%
```

Connection Processes for UNIX Dimensions Clients

Connecting Using the Remote Login Dialog Box

If you have an X Windows-based GUI environment installed on your UNIX Dimensions client (for example, Motif) and have the the X Windows environment `DISPLAY` set appropriately, then entering:

```
dmcli
```

at the operating system prompt in a terminal window launches the remote login dialog box. Complete this dialog box in one of the following ways:

- Complete the fields.

NOTE The user name and host name are initially inherited from the client operating system. These must be replaced when the Dimensions server and client are not physically located on the same machine.

  You can store the field values by assigning a name in the Previous Connections field. The stores the values into the `.dimensions.rc` file. You can then use the stored login information from the dialog box, or from the command-line `dmcli -con` command.

  Previous connection details can be deleted by use of the X button to the right of the Previous Connections field. This will also delete the information from the `.dimensions.rc` file.

- Loading values from the file `.dimensions.rc` in your home directory.

This file is created following a successful log in attempt. All of the fields in the login dialog box – with the exception of the password – are saved to the file under the heading of a Connection Name. By default the most recently used connection is loaded by the dialog box.

An example `.dimensions.rc file` is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<DimensionsConnections>
   <conname id="test1">
          <user>devlop</user>
          <dmdb>devlop</dmdb>
          <dsn>dev8</dsn>
          <host>aix4</host>
          <auto>yes</auto>
          <dflt>yes</dflt>
   </conname>
</DimensionsConnections>
```

Once successful connection has been established, the login dialog box will be dismissed and a

```
Dimensions>
```

prompt will be displayed in the terminal window. You can now enter commands on the Dimensions client.

NOTE All the UNIX connection mechanisms described below will also default to the GUI login dialog mechanism if they fail to successfully establish a connection.

Connecting from a Server Using the DMDB Environment Variable

On a Dimensions server *only*, you can get a local connection from the command line by setting the UNIX environment variable `DMDB`. Dimensions will search for any existing occurrences of the `PCMSDB` environment for backward compatibility with any scripts you have.

The syntax for `DMDB` is:

```
<base_db_name>@<connection_string>
```

for example,

```
intermediate@dim9
```

Connecting Using the -con Command-Line Parameter

This connection mechanism is functionally exactly the same as "Connecting Using the -con Command-Line Parameter" on page 19. Refer to that description for details.

Connecting Using the Command-Line

To support UNIX Dimensions client connections from non-Motif supporting terminals there is a command-line login interface to the saved `.dimensions.rc` connection file. The determining factor in these cases is the value of the X-Windows DISPLAY environment variable, if it is set the GUI login dialog box is used.

This connection mechanism is functionally exactly the same as "Connecting Using the -con Command-Line Parameter" on page 19. Refer to that description for details.

Examples of Client Commands

```
dmcli -param "c:\connection.txt" -cmd SCWS
```

connects to a Dimensions server using the parameters specified in c:\connection.txt and executes a SCWS command.

```
dmcli -user pcms -pass XXXX -host server1 -dbname intermediate -dsn dim9
    -cmd "FI FS:CABIN REPORT.A-SRC;1 /USER_FILENAME=c:\report.c"
```

connects to the Dimensions server node server1 as user pcms and executes a Get (Fetch) Item command of a text file.

```
dmcli
```

Without any parameters an interactive login box is invoked and you can enter your connection information. Commands can then be typed at the Dimensions client command prompt.

Important Considerations for Item Commands

The following topics describe key issues to consider when you work with items and item revisions using the command-line interface.

Selecting Item Revisions

When you run a command on an item, you can often specify a specific revision to act on. When you do not specify a revision, the command defaults to the *latest revision in the user's current project*. Note that this may not necessarily be the latest revision of the item in the database, since only the revisions in the user's *project* are considered. *Latest revision* means that version file content *has most recently been created or updated* – which may not necessarily be the highest numbered revision. For items which had previously been checked out, the time of creation/update is regarded as the time of the check in (`RI` command), *not* the earlier time of the check out (`EI` command). This does not take into account the branch names or whether the branches are locked or owned remotely (via replication).

Criteria other than the above are *not* used in selecting a revision by default. Consider the following example:

- Revision 2 of an item is the latest revision.
- Revision 1 item has been related to request A_B_1 as *Affected*.
- Now in order to create Revision 3 as *In Response To* request A_B_1, the check out (EI command) *must specify Revision 1* in the `<item-spec>`. This ensures that Revision 3 starts off as identical to Revision 1. Otherwise, by default, Revision 3 starts off as identical to Revision 2, even thought Revision 2 was *not* cited as *Affected*.

Incomplete Item Specification

An incomplete item specification is permitted.

- If you do not provide an item revision, the latest revision is used (for update commands a new revision is created).
- To omit other item specification fields, use a command's `/FILENAME` qualifier to specify the full relative path of the item in a stream or project. Use the UNIX format including the filename extension. For MVS items, a relative path of `q1.q2...ext(fn)` has the form `q1/q2/.../ext/fn.ext`.

  Rules:

  - `<product-id>` and its following ':' are required.
  - You can omit one or more of: `<item-id>`, `<variant>`, `<item-type>`, and `<revision>`
  - If you specify any of these fields you must also specify the character that precedes it. For example, if you specify `<variant>` it must be preceded by a '.'
  - You can specify all the punctuation but none of the values.

Examples:

```
EI ACCTS:.-; /FILENAME="CCOBOL/ACCT01.CCOBOL"

EI ACCTS: /FILENAME="CCOBOL/ACCT02.CCOBOL"

FI ACCTS:;2 /FILENAME="ASM/P000002.ASM"

RI ACCTS: /FILENAME="ASM/P000002.ASM"
```

Updating the Content of an Item Without Changing the Item Revision

If the PRODUCT-MANAGER has set up the process model to allow you to update the *file content* of an item revision at the initial lifecycle state *without* having to change the item revision, keep the following in mind.

A particular item revision may have been imported into several projects either manually or as a result of project replication (it should be remembered that a project is basically a logical group of item revisions). In such situations, if you edit the content of an item revision in one project *without* changing its revision, then in *all* projects that reference that item revision the content of the associated item file will be updated. Conversely, if you edit the content of an item revision in one project *and* change its revision, the changes in content will **only** be reflected in that project and any project replicated from it.

Assigning Default Project and Working Location

Every user must have a default project assignment. This default project supplies the default value for any command that requires a project specification.

Assigning the Default Project

The default project is assigned as follows:

- When the Tool Manager initially registers a Dimensions user, the user is automatically assigned to the default global project called `$GENERIC:$GLOBAL`. This project assignment enables the user to reference any item revision in any product in the base database to which they are connected.
- Subsequently, the user can then use the command SCWS (Set Current Project) or the `/WORKSET` qualifier found on certain commands to reference a specific project, such as one created from a baseline representing some development activity (this will then enable the user to reference item revisions that are pertinent to that development activity only).

  SCWS command qualifiers can be used to reassign (or not) the default project as described below:

  - `/DEFAULT` to specify that the current project assigned by SCWS will remain the default for all future sessions until respecified. An example of such a command is:

    ```
    SCWS PROD_X:MAINT /DEFAULT
    ```

  - `/NODEFAULT` to specify that the current project assigned by SCWS is for the duration of the present process/session only, and that the current project will revert to its former default setting once the session is exited. If neither the `/DEFAULT` nor `/NODEFAULT` qualifier is specified, then SCWS behaves as if `/DEFAULT` was specified. An example of such a command is:

    ```
    SCWS PROD_X:MAINT
    ```

  The /WORKSET qualifier found on certain commands is used in most cases to specify the project to be used for the *duration of the command* concerned.

Assigning the Working Location

Each project, when opened, must have a (mandatory) top level "working location" assigned to it. This "working location" defines a point in the directory hierarchy structure below which (or relative to which) the project file name is placed e.g. in UNIX `<dir>/<ws_filename>`. This is assigned as follows:

- By, where applicable, the `/DIRECTORY` command qualifier.
- The user's current working directory if `/DIRECTORY` is not specified or is not applicable.

The project file name as used in commands such as get or build consists of the relative directory path from the working location `<directory-spec>` and the file name from `<ws_filename>` concatenated together.

Requirements for Request Attributes

Review the following guidelines for request attributes and ensure that they are all followed.

Required Attributes for the RPT Command

You must follow these guidelines in order to successfully generate a request report using the RPT command. See "RPT – Report Requests" on page 401.

- The request attribute 1 must always be defined in the process model with variable name TITLE. It must be declared as single-valued. Its length must be less than or equal to 80 characters.
- Request attributes 2 and 3 must be defined in the process model and they must be defined as single-valued. These attributes appear in the report when users are e-mailed as the result of requests being actioned to new states.

Block Table Requirements

Attributes used to define a block (table) must satisfy the following conditions.

- They must all be multiple-valued.
- They must all be declared as visible.

Using Certificates

To access the Dimensions server command-line utility using a certificate, use the `-cert` option. This is most useful in conjunction with the REXEC (see "REXEC – Execute a Job on a Network Node" on page 376) or RSTAT (see "RSTAT – Update Job Status" on page 411) command. For detailed information on certificates as well as the RSTAT and REXEC commands, see the *Developer's Reference*.

Example

The following sample code is from a batch file in the `templates` directory:

```
echo RSTAT %DMJOBID. /STATUS=SUCCEEDED /RC=0 > c:\temp\dmcli.in
echo quit >> c:\temp\dmcli.in
dmcli -cert %DMCERTIFICATE. -file c:\temp\dmcli.in >
    c:\temp\%DMJOBID..log
```

This sample code generates the following batch file:

```
echo RSTAT R-4195789 /STATUS=SUCCEEDED /RC=0 > c:\temp\dmcli.in
echo quit >> c:\temp\dmcli.in
dmcli -cert
    81C4AC3983A8AB3CD847B6BA185BF8C6853B7102BB0ADA1B1BF420FE96EFB90E2F9
    4F008AB99435FCE153EA40EE8C6F7C159E58BC61E01725EE6E7C491A88FE78C8DCE
    58F7A2824FCE00EBF0A2169C073873DD825430316B341A8A5C7F0B38EBAD677FF81
    53F7E5F < c:\temp\dmcli.in
```

Operating System Differences

Keep in mind the following key differences between supported Dimensions operating systems, when working with the command-line interface.

Spaces in File Names

UNIX and Windows operating systems support the use of spaces in file names.

Dimensions may allow the creation of files with leading and trailing spaces but some tools may not be able to access these files.

Windows UNC Paths

Dimensions allows the use of UNC (Universal Naming Convention) paths for work areas. If a user's working location is set as a UNC path and the user opens the project on UNIX, the directory path will not be recognized.

Specifying a Project File Name in z/OS Item Operations

When running Dimensions item operations from a z/OS platform, you need to specify a 'backslash' character (\) in place of any parenthesis within a project file name. For example, if the project file name is `TEST.COBOL(STAFF)`, to get (fetch) the item using the project file name you would need a command such as:

```
                    fi cv3prod:.-src /filename="TEST/COBOL/STAFF.COBOL"
    /user_file="cvuser3.test.cobol(staff)"
```

Command-Line Logging and Usage Analysis

The Dimensions server provides command-line logging and usage analysis. You can use this to perform command-line auditing and general usage analysis.

This logging functionality is available in three forms:

- All users can view a summary of all database commands that a server has processed for viewing via a Dimensions published view. See "Audit Trail of Commands" on page 29 for more details.

- All users can log the full details of the commands that a server has processed to a file. This includes client machine details, user details, and full commands. See "Logging All Commands Run by All Users" on page 29 for more details.
- Each user also has the ability to log all their command details to a file, with the *SET* command. See "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428 for more details.

Audit Trail of Commands

The Dimensions server can summarize all of the database commands processed by the server. This summary information is available through the published view PCMS_COMMAND_STATISTICS, which contains the following information:

- The Dimensions command

NOTE Only the command name, for example, `CBL`, `CRB`, `CC` is logged—not command qualifiers or parameters.

- The user who ran the command.
- The last date the command was run.
- How many times the user has *successfully* run this command.
- How many times the user has *unsuccessfully* run this command.

You can enable this audit trail by setting the `DM_AUDIT_CMD_USAGE` option to true in the dm.cfg configuration file, or in the operating system. Then, to display the logged data, you can connect to the Dimensions database via a valid report user and run a SQL query such as:

```
SELECT * FROM pcms_command_statistics
```

using an SQL tool like sqlplus.

For more information on Dimensions published views and the PCMS_COMMAND_STATISTICS view, please see the *Reports Guide*. For details on modifying the dm.cfg configuration file, see the *System Administration Guide*.

Logging All Commands Run by All Users

The Dimensions server can log all commands run by all users, including parameters and session inforrmation. This log stores all commands except for commands that have a `/PASSWORD` qualifier in them, such as `AUDIT`, for security reasons).

To enable this logging, set the following parameter in the dm.cfg file:

```
DM_INTERNAL_AUDIT_CMD_FILE <logFile>
```

Where `<LogFile>` is the absolute path to the logging file. This file will be created and owned by the user running the Dimensions pooled servers. For more information on the dm.cfg file, see the *System Administration Guide*.

This log file will contain information such as:

```
** dmappsrv log "Wed Dec 31 22:36:21 2003" (GMT)
    (DMDB=INTERMEDIATE_TESTDB@hotaruchan) pid=3328 user="reichanadmin"
    node="AYANAMI" execution time = 0(s)
   'lwsd' (SUCCESS)
** dmappsrv log "Wed Dec 31 22:36:24 2003" (GMT)
    (DMDB=INTERMEDIATE_TESTDB@hotaruchan) pid=3328 user="reichanadmin"
    node="AYANAMI" execution time = 0(s)
   'lwsd' (SUCCESS)
```

An example of log output for a `get` operation on an item:

```
** dmappsrv log "Thu Mar 04 14:46:09 2010" (GMT)
    (DMDB=QLARIUS_CM@dim2009) pid=126240 user="dmsys" node="STAL-VC-
    2009" execution time=4(s)
'FI "QLARIUS:A57.A-SRC;1.0" /
    USER_FILENAME="C:\DOCUME~1\dmsys\LOCALS~1\Temp\pt1ebd41.txt" /
    EXPAND /NOOVERWRITE /WORKSET="QLARIUS:V" /NOMETADATA' (SUCCESS)
```

NOTE All log times are reported in GMT format to allow meaningful comparisons across time zones.

Logging Users Who Connect to Dimensions

In secure environments where you wish to track all users attempting to connect to a Dimensions server, Dimensions allows you to log all connection information for all clients. This log file is defined by the following parameter in the dm.cfg file:

```
DM_USER_AUDIT_LOG_FILE <file-name>
```

Where `<file-name>` is the absolute path on the server to the log file. This log file contains details such as when connect attempts were made, from which clients, by which user, and if that connection was successful or not.

All connection attempts are split into two types:

- The first type is an operating system user check. This verifies that the user attempting to log in actually exits on that server.
- The second type is a check to verify that the user specified is registered against Dimensions.

For example:

```
** dmpool connect "Thu Oct 14 19:40:53 2004" (GMT) pid=3332
    user="reichanadmin" node="AYANAMI"
   User attempted to login to Dimensions - OS user check (SUCCESS)
** DMAPPSRV connect "Thu Oct 14 19:40:54 2004" (GMT)
    (DMDB=ENTRY_LEVEL_TESTDB@hotaruchan) pid=3396 user="reichanadmin"
    node="AYANAMI"
   User attempted to login to Dimensions - database check (SUCCESS)
```

NOTE All log times are reported in GMT format to allow meaningful comparisons across time zones.

Invoking Help at the Dimensions Command-Line

When working at the Dimensions command-line interface, you can invoke text based help for any Dimensions command. When you invoke help, the following information is returned:

- The full name of the command.
- The complete syntax of the command.

To invoke help for a command:

At the Dimensions command line, type:

```
help <Dimensions function mnemonic>
```

For example:

```
Dimensions>help abl
ABL - Action Baseline or Items
        <baseline-spec>
        [/ITEM_FILTER=<item-spec>]
        [/STATUS=<status>]
Operation completed
```

Chapter 2

Command Reference

This chapter contains an alphabetic listing of commands.

NOTE For detailed information on roles, groups, and privileges, see the *Dimensions CM User's Guide* and the *Dimensions CM Process Configuration Guide*.

ABL – Action Baseline or Items

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
[/ITEM_FILTER=<item-spec>]
[/STATUS=<status>]
[/COMMENT=<text>]
```

Example

```
ABL PROD:"R M VERSION 2 FOR HP" -/STATUS="UNDER TEST"
```

Parameters and qualifiers

- `<baseline-spec>`

  Comprises:

  `<product-id>:<baseline-id>`

- `/ITEM_FILTER=<item-spec>`

  Comprises:

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

  Wildcard characters _ (underscore) for "any one" and % (percent) for "zero or more" characters may be used to identify what subset of the item revisions in the baseline are to be actioned to a new *item* lifecycle state.

  If omitted, it is `<baseline-spec>` itself that is actioned. See Description on page 34 for details.

- `/STATUS=<status>`

  Specifies the new status to be given *either* to the baseline itself *or* to every item revision in the subset identified above. Unless you hold the `PRODUCT-MANAGER` role, this status must be reachable from the current status (of each object to be actioned) by a single lifecycle transition.

  If omitted, the new status is the next normal lifecycle state for each object. See Description on page 34 for details.

- `/COMMENT=<text>`

  This is an optional user-defined action-comment.

  Dimensions enters a default comment if this is omitted.

Description

This command actions to a new lifecycle state **either** the specified baseline (if the `<item-spec>` filter is omitted) **or** each of the item revisions in the baseline that match the specified filter. (To action both the baseline itself and (a subset of) the item revisions in it, two instances of the ABL command must be used.)

If `<status>` is omitted, the object(s) to be actioned must each be at a normal lifecycle state, and each will be advanced one state further along the normal lifecycle. Thus it is possible in a single ABL operation to advance all the matching item revisions each by one approval level (i.e. each moves along its normal lifecycle by one transition), even when the start and end states of these transitions are not the same for all the item revisions

actioned. This is particularly convenient when the matching item revisions are of more than one item type, which means that they would probably be following different lifecycles.

Constraints

A user can action a baseline or item if they have one of the following privileges:

ID	Short Description
BASELINE_ACTION_NEXTSTATE	Baseline-Action to Next State
BASELINE_ACTION_ANYSTATE	Baseline-Action to Any State
ITEM_ACTION_NEXTSTATE	Item-Action to Next State
ITEM_ACTION_ANYSTATE	Item-Action to Any State

AC – Action Request

```
<request-id>
[/STATUS=<status>]
[/ACTION_CHECK] or [/CLOSURE_CHECK]
[/COMMENT=<text>]
[/DESCRIPTION=<desc-file>]
```

Examples

```
AC PROD_DR_25 /STATUS="CRB APPROVED"
AC PROD_DR_26 /STATUS="CRB APPROVED" /ACTION_CHECK
AC PROD_DR_27 /ACTION_CHECK
AC PROD_DR_28 /CLOSURE_CHECK
AC PROD_HELD_350
```

Parameters and qualifiers

- `<request-id>`

  The identity of the request to be actioned or checked.

- `/STATUS=<status>`

  Specifies the new status to be given to the request (provided `ACTION_CHECK` is omitted). Unless the user has the CHANGE-MANAGER role (see note below) the new status must be reached by a single lifecycle transition from the current status.

  If this parameter is omitted, Dimensions actions the request to its next state in the normal lifecycle. If the request is held Dimensions saves it, which places the request at its initial lifecycle state.

NOTE Saving a request changes the value of `<request-id>`, but you can use `$LAST` to refer to the request in subsequent commands in a CMD file. See the note on the CC command-mode command on page 90.

CAUTION! You cannot omit `<status>` if the current status is a state not in the normal lifecycle.

- `/ACTION_CHECK`

  Checks if the specified request can be actioned to the state specified by the `/STATUS` qualifier, or the next normal state if `/STATUS` is not specified (and conform to the current rules). Cannot be used with `/CLOSURE_CHECK`.

  Mandatory attributes must be set to action to the next normal state or to the state defined by `/STATUS`.

- `/CLOSURE_CHECK`

  Checks if the specified request can be actioned to the final state in its normal lifecycle, and conform to the current rules. Cannot be used with `/STATUS` or `/ACTION_CHECK`.

- `/COMMENT=<text>`

  A description of the action.

- `/DESCRIPTION=<file-desc>`

  Specifies a file containing a description of the action when the description is too long to specify on the command line. Use instead of `/COMMENT`.

NOTES

- Users holding the Action to any State privilege may action any request to any valid state in its lifecycle, including re-opening a request that has been closed or rejected (has reached a final state).
- If you are actioning requests that are related to Dimensions RM requirements there are special considerations. See "Dimensions CM Requests and Dimensions RM Requirements" in the "Miscellaneous Functions" chapter of the *User's Guide* and *Dimensions CM-Dimensions RM ALM Integration Guide*.

Constraints

Unless you have the appropriate management privileges to action a request, you must have a role required to action the request to a new state. To select any lifecycle state from any stage of the lifecycle you need the CHANGE-MANAGER role or have the role on the lifecycle transition if that transition exists.

Requests that were created in a held state are not considered to have been "created" by process models where optional sensitive states or attributes have been set up ("electronic signatures"). Entering a request into the system by actioning it out of the held state is considered the "authorization point" for these process models. Also applies to held requests that are updated at the held state (using the command UC) before being actioned.

ACDI – Action Request Items

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[/LOGFILE=<log-file>]
[/STATUS=<status>]
[/WORKSET=<project>]
```

Example

```
ACDI PAYROLL_CR1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/STATUS`

  Specifies the status to which items and requests are to be actioned. If this is not specified, the next default state is used.

- `/WORKSET`

  Specifies the project/stream to be processed by this command.

Description

This command actions to a specified state all the items and requests that are related to a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

If any failure occurs, all actions are rolled back.

ACDWS – Add Request Items to Project

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[/DIRECTORY=<project-directory-filter>]
[/[NO]RECURSIVE]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/LOGFILE=<log-file>]
/WORKSET=<project>
[/[NO]KEEP_STAGE]
```

Example

```
ACDWS PAYROLL_CR1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items processed.

- `/RECURSIVE`

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which this change to the project is to be related `In Response To`.

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/WORKSET`

  Specifies the project to be processed by this command.

- `/KEEP_STAGE`

  Specify this optional qualifier to control the stages of the items that you add.

  - Use /NOKEEP_STAGE to reset the stages of the items to the initial stage.
  - Use /KEEP_STAGE to keep the stages of the items from the source project.

  This qualifier can only be used when the project uses the manual deployment model.

  Default (when the qualifier is not specified): /KEEP_STAGE

Description

This command adds to the project specified by the /WORKSET qualifier all the items that are related to a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

This command is not available for streams.

ACF – Assign Data Formats to Request Types

```
<product-id>
/TYPE=<request-type>
/FORMAT=<format>
[/EXTENSION=<file-extension>]
```

Example

```
ACF PAYROLL /TYPE=CR /FORMAT=HTM /EXTENSION="html"
```

Parameters and qualifiers

- `<product-id>`

  Specifies the product within which the assignment is to be made.

- `/TYPE=<request-type>`

  Specifies the request type within the specified product to which the format assignment is to be made.

- `/FORMAT=<format>`

  Specifies a valid data format to be assigned to the specified request type. This will then become the valid data format assigned when creating a request of type `<request-type>`.

- `/EXTENSION=<file-extension>`

  Optionally specifies a file extension to be assigned to the specified request type.

Description

This command assigns a data format to a particular request type. The data format must have been previously defined using the Define Data Format (DDF) command (see page 153) or the Administration Console (see the *Process Configuration Guide*).

Once assigned, the format and file are used by the Dimensions client applications (web client, desktop client, and IDE) to correctly choose an application/viewer to display the request.

Optionally, you can also specify a file name extension to be assigned to the specified request type.

This function is available *only* in Command Mode.

Constraints

Only users with the appropriate management privileges can run this command.

ADF – Assign Data Formats to Item Types

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<product-id>
/ITEM_TYPE=<item-type>
[/FORMAT_LIST=(format1,format2,format3,...)]
```

Example

```
ADF PROD /ITEM_TYPE=DAT /FORMAT_LIST=(C,TXT,CPP)
```

Parameters and qualifiers

- `<product-id>`

  Specifies the product within which the assignment is to be made.

- `/ITEM_TYPE=<item-type>`

  Specifies the item type within the specified product to which the format assignments are to be made.

- `/FORMAT_LIST=<format1,format2,format3,...)`

  Specifies the list of valid data formats to be assigned to the specified item type. This will then become the valid list of data formats from which users must select when creating an item.

  If `/FORMAT_LIST=.` (dot) is specified, then any existing assignments are cleared.

Description

This command assigns data formats to particular item types. The data formats must have been previously defined using the Define Data Format (DDF) command (see page 153) or the Administration Console (see the *Process Configuration Guide*). Once these formats are assigned to an item type, the choice of one these formats is compulsory when creating items of that type; whereas, if none has been assigned, then any format can be used at the time of item creation, even one not defined by a user with the role of TOOL-MANAGER.

IMPORTANT! This command overwrites the existing list of formats.

Constraints

Only users with the appropriate management privileges can run this command.

This function is available *only* in Command Mode.

AGRPU – Assign Groups to a User

```
<user-name>
[/GROUPS=(<group-name>,...)]
[/ADD] or [/REMOVE]
```

Example

```
AGRPU <user-name> /GROUPS=(<group-name>,...) /ADD
```

Parameters and qualifiers

- `<user-name>`

  is the user name for the user to whom you are adding groups or from whom you are removing groups.

- `<group-name>,...`

  is the list of groups to be added or removed.

Description

Use this command to assign groups to a user or to remove group assignments from a user.

Constraints

Only users with the appropriate management privileges can run this command.

AI – Action Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/STATUS=<status>]
[/COMMENT=<text>]
[/WORKSET=<project-spec>]
```

Example

```
AI PROD:"QUERY RELEASE".AAAA;2 /FILENAME=query.c -/STATUS="UNDER TEST"
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if `<file-name>` is specified.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project/stream. The project file name for the same item may *differ* between projects or streams; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/STATUS=<status>`

  Specifies the new status to be given to the revision.

  Except as noted below, it must be reachable from the current status by a single lifecycle transition.

  It can be omitted, **only if** the current status is a state in the normal lifecycle, in which case the item will be actioned to its next normal lifecycle state.

- `/COMMENT=<text>`

  An optional user-defined action-comment.

  Dimensions enters a default comment if this is omitted.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project/stream to be used for this command: failing this, the user's current project/stream will be taken.

The project/stream is used to select the revision to action if the revision is not actually specified. If the revision is specified, the project or stream is ignored (as Dimensions assumes reference to the explicit revision).

Constraints

Unless you have the appropriate management privileges, you can run this command only if the current item revision is in your pending list.

NOTE To simplify the transfer of an existing product to Dimensions, a user with role of PRODUCT-MANAGER can action any item revision to any valid state in its lifecycle. This includes permission to re-action a revision which has reached a final state, thereby re-opening it to further ordinary actioning.

Such a user can also action items when no appropriate roles have yet been allocated. This is to facilitate quicker migration of files.

AIWS – Add Item Revision to Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for items that belong to a stream.

```
<item-spec>
[/FILENAME=<file-name>]
/WORKSET=<project-spec>
/WS_FILENAME
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/[NO]KEEP_STAGE]
[/USER_ITEMLIST="item list path"]
```

Example

```
AIWS PROD_X:"HELLO WORLD".AAAA-SRC;2.6 /WORKSET=PROD_X:"WS MAINT"
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists
`<revision>`	defaults to the latest revision in your current project.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

NOTE When you add an item to a project where it already exists, the filename will be the name of the item in the project. For example, if you add the item 'foo.c' to a project, and the same item exists in that project with the name 'boo.c', the imported item will be named 'boo.c'.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This command will add the item specified to the given project. The specified item and project must exist. If the specified item is already in the project a warning will be given.

  Item revisions may be removed from a project using the RIWS command.

- /WS_FILENAME

  Specifies the workset filename for an item to be added to a project. For example, the following command adds a file called *file.c* with the new filename *foo.c* to a project called *test*:

  ```
  AIWS "FOO C" /WORKSET=test /FILENAME=file.c /WS_FILENAME="foo.c"
  ```

  When the target project already contains a revision of the item that is going to be added:

  - If you do not specify /WS_FILENAME a warning is displayed if the item paths across the source and target projects do not match.
  - If you specify /WS_FILENAME the specified path is used as the new path of the item in the target project. Therefore the item is effectively renamed if its existing path differs from the /WS_FILENAME value.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

<requestN>	identifies a request to which this change to the project is to be related In Response To.

  Specify this optional qualifier if you want the change (i.e. item addition) to the project to be recorded against the specified request(s). If path control has been enabled, this qualifier is mandatory. If path control is not enabled, then the request(s) will be ignored.

- /KEEP_STAGE

  Specify this optional qualifier to control the stages of the item revisions that you add.

  - Use /NOKEEP_STAGE to reset the stages of the item revisions to the initial stage.
  - Use /KEEP_STAGE to keep the stages of the item revisions from the source project.

  This qualifier can only be used when the project uses the manual deployment model.

  Default (when the qualifier is not specified): /KEEP_STAGE

- /USER_ITEMLIST="item list path"

  Specify this qualifier to export or remove multiple items and to submit a single deployment job for all of the specified items. For example:

  ```
  /USER_ITEMLIST="C:\itemlist.txt"
  ```

  The item list file has the following format:

  ```
  "item spec1" "ws_filename1"

  "item spec2" "ws_filename2"

  "item specN" "ws_filenameN"
  ```

  Notes:

  - You can omit the "ws_filename" column.
  - You do not need to specify <itemSpec>.
  - /FILENAME and /WS_FILENAME are ignored.

Description

The item selected by the `<item-spec> [/filename=<file-name>]` parameters from the current project will be added to the project specified by the `/WORKSET=<project-spec>` qualifier. The baseline project file name is not used.

NOTE Whenever a new revision is added to a project, its stage is reset to DEVELOPMENT, and associated deployment areas and library cache areas are updated.

Constraints

- Users must have been granted the privileges:
  - `deploy item to next stage`
  - `deploy item to any stage`

  This constraint can be relaxed using the Set Project Permissions (SWSP) command, see page 451.
- Cannot be used for streams.

ANNOTATE - Display File Annotations

```
<ITEM_SPEC>
[/WORKSET=<WORKSET>]
[/LINES=<RANGE>]
[/ROOT_PROJECT]
[/FILENAME]
[/USER_FILENAME]
```

Description

Displays an annotated listing of a source file with this information:

- The line number.
- The revision number of the change.
- The person who made the change.

Example

```
ANNOTATE "QLARIUS:AUTOQUOTE JAVA.A-SRC;java_s1_1#1"
   /WORKSET="QLARIUS:JAVA_BRANCHA_STR" /LINES=1-14
```

```
1    java_s1_0#1  ASMITH  /*
2    java_s1_1#1  PRANDALL * Automotive Insurance Quotation Application
3    java_s1_1#1  PRANDALL * This is the main code for the GUI for AutoQuote application
4    java_s1_0#1  ASMITH   *
5    java_s1_0#1  ASMITH   */
6    java_s1_0#2  PRAYMOND package qlarius.interfaces;
7    java_s1_0#1  ASMITH
8    java_s1_0#3  LLEWELL import javax.swing.JFrame;
9    java_s1_0#1  ASMITH
10   java_s1_0#1  ASMITH  // @author Serena
11   java_s1_0#4  LTHOMAS private javax.swing.JPanel jContentPane = null;
12   java_s1_0#1  ASMITH  private javax.swing.JMenuBar jJMenuBar = null;
13   java_s1_0#1  ASMITH  private javax.swing.JMenu fileMenu = null;
14   java_s1_0#1  ASMITH  private javax.swing.JMenu editMenu = null;
```

Parameters and Qualifiers

- “<ITEM_SPEC>”

  Item specification (binary files not supported).

- /WORKSET=< WORKSET>

  Specifies the project or stream where the item is located.

- /LINES=<RANGE>

  Specifies a line range to be displayed in the output. For example:

  /LINES=10-20: displays lines 10 to 20.

  /LINES=10-*: displays lines 10 to the end of the file.

  /LINES=*-20: displays lines 1 to 20.

  /LINES=15: displays the 15th line.

- /ROOT_PROJECT=<project-spec>

  where <project-spec> is comprised of:

  <product-id>:<project-id>

  Specifies the root project. Use when the current project set via the SCWS command, or the project specified by the /WORKSET qualifier, occurs in more than one project tree.

- /FILENAME=<file-name>

  Specifies the name of the project file. If /ROOT_PROJECT is used to specify the root project, /FILENAME is interpreted in the scope of that project.

  The project file name identifies the relative path (directory and file name) from the working location of the file to be used when the item is fetched from the current project. The project file name for the same item may differ between projects. For example:

  src/hello.c

  hello.c

  src/build/hello.c

  May be omitted if <item-id> is specified.

- /USER_FILENAME

  Specifies a file where the annotated information is saved instead displaying the output in the console.

APNO – Allocate Part Numbers

```
<part-spec>
[/GENERIC_NO=<standard-no> [/NOCHECK]]
[/LOCAL_NO=<local-no>]
[/DESCRIPTION=<description>]
```

Example

```
APNO PROD:"RELEASE MANAGEMENT" /GENER="SQLS 1234"
```

Parameters and qualifiers

- `<part-spec>`

  Specifies the design part to be numbered. It comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored. A part-number always applies to all PCSs.

- `/GENERIC_NO=<standard-no>`

  Specifies a standard part number to be allocated.

  It may be omitted provided `<local-no>` is specified.

  - `/NOCHECK`

  Specifies that the standard part number need not be in a range of numbers allocated to the product.

- `/LOCAL_NO=<local-no>`

  Specifies a local part number to be allocated.

  It may be omitted provided `<standard-no>` is specified.

- `/DESCRIPTION=<description>`

  Specifies a new description to be given to the design part.

Constraints

Only users with the appropriate management privileges can run this command.

Each part category that is to use part numbers has to be enabled by the Process Modeler.

AUDIT – Audit Area

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
/STAGE=<stage-spec>
/USER_FILENAME=<file-spec>
[/AREA_LIST=<areaList>]
[/FILTER=<area-filter-name>]
[/[NO]FIX]
```

Examples

Audit the UNIT TEST area "ACME_2.1-WINDOWS-UT", which is assigned to project ACME:ACME_2.1, repairing the area and generating a report file:

```
AUDIT "ACME:ACME_2.1"
   /STAGE="UNIT TEST" -
   /AREA_LIST="ACME_2.1-WINDOWS-UT" -
   /USER_FILENAME="c:\my_audit_report.txt" -
   /FIX
```

Audit the contents of the DEVELOPMENT deployment area ACCT1.0-ZOS-DEV, which is assigned to project EXEDLL:EXEDLL 2.0, and generate a report file:

```
AUDIT "EXEDLL:EXEDLL 2.0"
   /STAGE="DEVELOPMENT" -
   /USER_FILENAME="D:\temp\audit.log" -
   /AREA_LIST="ACCT1.0-ZOS-DEV"
```

Parameters and qualifiers

- `<project-spec>`

  Comprises `<product id>:<project id>` and specifies a project or stream associated with the area to be audited.

- `/STAGE=<stage-spec>`

  Specifies the ID of a deployment stage to be audited.

- `/USER_FILENAME=<file-spec>`

  Specifies a file where the audit output is to be saved.

- `[/AREA_LIST=<areaList>]`

  Specifies the IDs of deployment areas to be audited. If not specified, all areas for the project/stream and stage pair are audited.

- `[/FILTER=<area-filter-name>]`

  Specifies the set of inclusion/exclusion rules that determine whether a file is to be excluded from a fix during an audit of the area when you run the AUDIT command.

CAUTION! Audit and area filters can easily be confused. See *Correct Use of Area and Audit Filters* in the *Area Definitions* chapter in the *Process Configuration Guide.*

- `[/FIX]`

  Repair an area (synchronize it with all items corresponding to the specified stage). Repairing an area ensures that it contains all item revisions from the project/stream

that are at the specified deployment stage. Any other files present in the area will not be affected by this feature.

NOTES

- AUDIT works with z/OS mainframes by utilizing Dimensions metadata stored on the mainframe.
- It is possible to audit one area or *all* areas associated with the specified project (or stream) and stage (by not specifying an area list on the command line).

AUGRP – Assign Users to a Group

```
<group-name>
[/USERS=(<user-name>,...)]
[/ADD] or [/REMOVE]
```

Example

```
AUGRP <group-name> /USERS=(<user-name>,...) /ADD
```

Parameters and qualifiers

- `<group-name>`

  is the name of the group to which you are adding users or from which you are removing users.

- `<user-name>,...`

  is the list of users to be added or removed.

Description

Use this command to assign users to a group or to remove user assignments from a group.

Constraints

Only users with the appropriate management privileges can run this command.

AUPG - Start Upgrade Process

```
/NETWORK_NODE=<node name>
```

Description

Starts an upgrade process on a Dimensions CM agent system.

Examples

- Start the upgrade process on the node ST6123:

  ```
  AUPG /NETWORK_NODE=ST6123
  ```

- Start the upgrade process on all registered Dimensions CM agent nodes whose host name matches the pattern "ST-WIN-0*":

  ```
  AUPG /NETWORK_NODE=ST-WIN-0*
  ```

Qualifiers

```
/NETWORK_NODE=<node name>
```

Specifies a network node where an upgrade process will be run.

AUR – Assign User Roles

```
<user-name> / <group name>
/ROLE=<role>
[/TYPE=<assignment-type>]
/PART=<product-id>:<part-id>.<variant>
[/CAPABILITY=<capability>]
[/ADD] or [/DELETE]
[/WORKSET=<project-spec>]
[/REPORT]
```

Example

```
AUR SMITH
   /ROLE=DEVELOPER -
   /CAPABILITY=P -
   /PART=PROD:"RELEASE MANAGEMENT" -
   /ADD
```

NOTE You can use the AUR command to allocate roles to groups as well as users.

Parameters and qualifiers

- `<user-name> / <group name>`

  This is the login user name of the Dimensions user to whom the role is (to be) assigned. Can also be the name of a group.

- `/ROLE=<role>`

  Specifies a role to be defined or assigned to a user.

- `/TYPE=<assignment-type>`

  Specifies the type of assignment. It is either C (denoting role candidate definition) or R (denoting actual user role assignment). If omitted, R is assumed.

- `/PART=<product-id>:<part-id>.<variant>`

  Specifies a design part over which this role assignment is applicable.

NOTE If the variant field is left blank, the role assignment applies to **all** variants of the design part, excluding those that have an explicit role.

- `/CAPABILITY=<capability>`

  Specifies that this role assignment is one of the following:

  - L for Leader
  - P for Primary
  - S for Secondary (default).

  Leader (**L**) role function: It is sometimes useful to have more than one user with a particular role with respect to a request or item-spec e.g. so that they can add comments (called Action Descriptions in requests). However, it may also be appropriate to restrict the number of users in this group who can actually action the object to the next stage in its lifecycle. The way to implement this is via the Leader function. When a Leader function is defined in a group of users who have the same role for a given object, only the Leader can update the associated attributes **and** action on the object. All other users with the role may add only

Action Descriptions or user comments. The Leader function applies whether rules are used or not. If Leader role function is assigned to a user, then Primary role function (described below) cannot be assigned to the same user i.e. Leader role and Primary role functions are mutually exclusive.

The **Primary** (**P**) user for a role in the lifecycle of an object is the user regarded in the project/stream as having the main responsibility for that role on the request or item-spec. There cannot be more than one Primary user defined for a role (as applicable to any particular design part or segment of the product structure). If Primary role function is assigned to a user, then Leader role function (described below) cannot be assigned to the same user i.e. Primary role and Leader role functions are mutually exclusive

Secondary (**S**) users are intended to act as deputies for the Primary. They have exactly the same privileges as the Primary: they can add action comments and also, unless the Leader capability is in use, update the object's attributes and action them.

- `/ADD`

  Specifies that this user role assignment is to be added. This is the default.

- `/DELETE`

  Specifies that this user role assignment is an existing one to be revoked.

  If omitted, this assignment is a new one to be granted.

- `/WORKSET=<project-spec> comprises:`

  `<product-id>:<project-id>`

  This is optional. If specified, the role assignment will apply to that particular project/stream. If unspecified, the role assignment will apply to all projects/streams, unless the role is `WORKSET-MANAGER` (in which case it is necessary to assign a specific project/stream for that role-assignment to be effective).

- `REPORT`

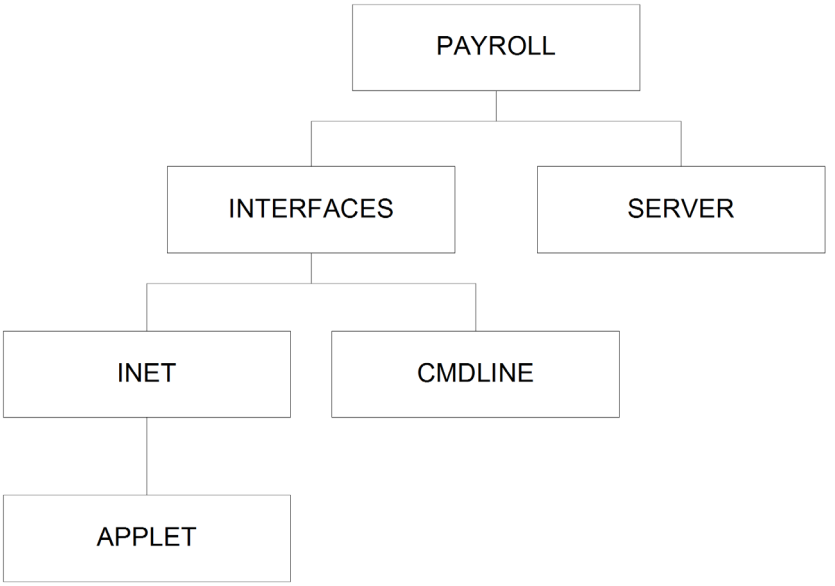
  This is optional. If specified, an on-screen report will be generated detailing what the result of such a proposed role assignment would be, without actually performing the role assignment—a "what if report".

NOTE This option is provided to aid administrators in understanding the impact of making a role assignment change.

Example

Consider the following design part structure and current user role assignments:

- User JOHN has role DEVELOPER on the top part `PAYROLL:PAYROLL`.
- User JILL also has the role DEVELOPER on the top part `PAYROLL:PAYROLL`.
- User CHRIS has the role DEVELOPER on the `PAYROLL:APPLET` design part.



Given the above scenario, entering the following command

```
AUR USER2 /ROLE=DEVELOPER /PART=PAYROLL:INTERFACES /REPORT
```

would generate a report, like the following, detailing how activating this role assignment would override the DEVELOPER role of various existing users who have been assigned that role at various other levels in the design part structure

```
Warning: You are removing a role from other user(s)...
   User JOHN was assigned the role on Design Part PAYROLL:PAYROLL.A;1
   By making this role assignment the above user will no longer have
        the role on part PAYROLL:INTERFACES.A;1 or any of its
        descendants

   User JILL was assigned the role on Design Part PAYROLL:PAYROLL.A;1
   By making this role assignment the above user will no longer have
        the role on part PAYROLL:INTERFACES.A;1 or any of its
        descendants

Warning: You are assigning the role on the following Design Parts...
   The role assignment will be effective on Design Part
        PAYROLL:INTERFACES.A;1
   The role assignment will also be effective on Design Part
        PAYROLL:INET.A;1
   The role assignment will also be effective on Design Part
        PAYROLL:CMDLINE.A;1

Warning: The role assignment will not take effect on Design Part
     PAYROLL:APPLET.A;1 or any of its descendants
   The following user(s) already hold the role:
      CHRIS

Operation completed
```

Constraints

Only users with the appropriate management privileges can run this command.

AUTH – Authorize Access to Node

```
[/NETWORK_NODE=<node-name>
[/USER=<userid or credential-set-name>
[/PASSWORD=<password>
[/NEW_PASSWORD=<new-password>]]]]
```

Examples

```
AUTH /NETWORK_NODE=MYNODE
```

requests a list of authenticated users on the node MYNODE

```
AUTH /NETWORK_NODE=MYNODE /USER=MICKEY /PASSWORD=MOUSE
```

requests access to user files on node MYNODE for the user MICKEY, with password MOUSE.

Parameters and qualifiers

- /NETWORK_NODE=<node-name>

  Specifies the name of the node where your user files are stored.

- /USER=<userid or credential-set-name>

  The User ID or credential set for the specified node. For more information about credential sets see the *Dimensions CM System Administration Guide*.

- /PASSWORD=<password>

  The password associated with the User Id that you specified. Not required if you specify a credential set name in the /USER parameter.

- /NEW_PASSWORD=<new-password>

  This is the string that you want to change your password to.

Description

The AUTH command enables you to perform tertiary node access to items located on a remote node. All communication across the network of this sensitive information is encrypted.

NOTE In a pure LDAP environment, this command requires a local operating-system account.

You can use AUTH to:

- Obtain information about current authenticated users as follows:
  - To obtain a list of all authenticated users for each of the nodes currently available, enter the command with *no* parameters.
  - To obtain a list of authenticated users for one node, enter the command with the parameter /NETWORK_NODE.
- Change the current user on a node. Enter the command with the parameters /NETWORK_NODE and /USER. The user must already have been authenticated.
- Request access for a specified user on a node. Enter the command with the parameters /NETWORK_NODE, /USER and /PASSWORD. You can also change the password at the same time, by specifying the parameter /NEW_PASSWORD.

AWS – Action Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
[/ATTRIBUTES=(<attr>,...)]
[/STATUS=<status>]
[/COMMENT=<text>]
```

Example

```
AWS <project-spec> /ATTRIBUTES=(<attr>,...)
```

Parameters and qualifiers

- `<attr>,...` is a list of attributes.
- <text> is an optional comment.
- If you omit `/STATUS`, AWS uses the next normal lifecycle state.

Description

This command is made necessary by the fact that projects have a user-defined lifecycle.

Constraints

Unless you have the appropriate management privileges, you must, for each object to be actioned, have been assigned a role authorized to perform its transition (whether `<status>` is specified or not).

BC – Browse or Print Request

```
<request-id>
/FILENAME=<user-filename>
[/[NO]PRINT]
[/ACTION_NO=<number>]
[/ATTACHMENTS=([FILENAME=<file-id>, USER_FILE=<user-file>],
               [FILENAME=<file-id>, USER_FILE=<user-file>],...)]
[/CONTENT_ENCODING=<file-encoding>]
[/TEMPLATE=<template-name>]
[/REV=<revision of template>]
```

Examples

```
BC PROD_DR_25 /FILENAME=D:\temp\dr_25.txt /ACTION=2

BC PAYROLL_CR_21 /FILENAME=D:\temp\patroll_cr_21.txt
/ATTACHMENTS=([FILENAME=Figure2.jpg,USER_FILE=c:\temp\attachment1.jpg],
              [FILENAME=Figure4.jpg,USER_FILE=c:\temp\attachment2.jpg])
```

NOTE You must specify the /FILENAME qualifier.

Parameters and qualifiers

- `<request-id>`

  This is the identity of the request to be browsed or gotten and/or printed.

- `/FILENAME=<user-filename>`

  Specifies the name of the file which will be created in the user area, and into which the request will be gotten. This command does not support any interactive browsing. This qualifier is mandatory.

- `/PRINT`

  Specifies that the request is to be printed. If this qualifier is used, interactive browsing is not invoked.

- `/ACTION_NO=<number>`

  Specifies that the request is to be browsed or gotten and/or printed in its state as it was prior to the action given by `<number>`.

  If this is not specified, the current state of the request is shown.

- `/ATTACHMENTS=([FILENAME=<file-id>, USER_FILE=<user-file>])`

  Specifies an attachment (<file-id>) to be retrieved from a given request (<request-id>) to the file that you specify in the <user-file> parameter. The /FILENAME qualifier is optional for non-interactive use if you use the /ATTACHMENTS qualifier (however <request-id> is still required).

  The FILENAME parameter in the /ATTACHMENTS qualifier is the attachment associated with the request, and the /FILENAME qualifier is the user file where the expanded browse template is saved.

Additional information:

- A request cannot have two attachments with the same <file-id>.
- When you add or delete attachments, or update their descriptions, the action is recorded in the request's history.
- A new request substitution variable called %chdoc_attachments% has been added, which enables you to view details of any attachment linked to a request that you browse.
- When you delete a held request, its attached files and history records are also deleted.

- `/CONTENT_ENCODING=<file-encoding>`

  Specifies the content encoding. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- `/TEMPLATE=<template-name>`

  Specifies the name of a browse template to be used.

- `/REV=<revision of template>`

  Specifies a revision of a browse template.

Constraints

This command can be run by any user with a role (any role will suffice) on the product owning the request selected.

BI – Browse Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions and cannot be run from Dimensions for z/OS.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/BASELINE=<baseline-spec>]
[/WORKSET=<project-spec>]
<tool-name>
```

Example

```
BI PROD:"QUERY RELEASE".AAAA-SRC C:\utils\textpad;
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored if `<baseline-spec>` is specified; otherwise, if omitted, the latest revision is used (see About the Command-Line Interface on page 12).

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project or stream. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/BASELINE=<baseline-spec>`

  Specifies a release-baseline which contains the particular revision of `<item-spec>` to be browsed. It comprises:

  ```
  <product-id>:<baseline-id>
  ```

  If this qualifier is omitted, the specified or default `<revision>` (as described above) is browsed.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project/stream to be used for this command: if not specified, the user's current project/stream will be taken.

- `<tool-name>`

  Specifies the program to use to view the item. For example, you might set this to the path to notepad.exe on your system.

Constraints

This command can be run by a user who has a role on the design part owning the item or a role on one of the ancestor nodes of that design part. For more information see the section 'About Role Responsibilities' in the *Process Configuration Guide*.

BLD – Build

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
[/AREA=<area-name>]
[/TYPE=DEPLOYMENT | WORK]
[/STAGE=<stage-name>]
[/[NO]AUDIT]
[/[NO]WAIT]
[/BUILD_CLEAN]
[/BUILD_CONFIG = <build-configuration-name>]
[/BUILD_OPTIONS = (<opt1>=<value1>,<opt2>=<value2>,...)]
[/[NO]CAPTURE]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/TARGETS=<targets-list>]
[/USER_FILENAME = <file-name>]
[/SRC_CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/SRC_FILES=(<item-filename1>,<item_filename2>,...)]
[/SRC_ITEMS=(<item-spec1>,<item-spec2>,...)]
[/SRC_FILELIST=[<node>::]<filespec>]
[/SRC_ITEMLIST=[<node>::]<filespec>]
[/TARGETS_LIST=[<node>::]<filespec>]
[/[NO]CANCEL_TRAVERSE]
[/POPULATE_SCOPE=NONE|ALL|REQUESTED]
[/[NO]TOUCH]
[/[NO]LOCK_SEARCH_PATH]
[/TARGET_PROJECT=[product:]projectname]
[/USER=userid]
[/PASSWORD=password]
[/DEPENDENCY_ANALYSIS_FLAGS=(list)]
```

Example

```
BLD "ACME_2.1" /AREA="ACME_2.1-WINDOWS-UT"
   /TARGETS=("foo.exe", "win32\bar.exe")
   /NOAUDIT
   /NOBATCH
   /CAPTURE
   /CHANGE_DOC_IDS=(PVCS_CR_1234)
   /USER_FILENAME=D:\temp\bld.log
```

Parameters and qualifiers

- `<project-spec>`

  Specifies the name of a project or stream.

- `/AREA=<area-name>`

  Specifies the Dimensions area to be used for the build. If this is not specified, all areas associated with the build configuration or configurations are used.

  Cannot be used together with /TYPE.

  If the parent product uses the Deployment Automation (DA) deployment model, this qualifier can only be used with work areas.

- `/TYPE=DEPLOYMENT | WORK`

  Builds all areas of the specified build type.

  Note: Cannot be used together with /AREA.

- `/STAGE=<stage-name>`

  Applicable only to deployment areas. If the area type is DEPLOYMENT, this qualifier specifies the stage for which the targets are to be built.

  If the parent product uses DA, this qualifier is not supported.

- `/AUDIT`

  Specifies that an audit is to be run before the build.

  Default: no audit

- `/WAIT`

  Specifies that the command will wait for the build to finish.

  Specify /NOWAIT for the build to be run in batch mode (the command does not wait for the build to finish).

  Default: wait.

- `/BUILD_CLEAN`

  Specifies that the area is to be cleaned of targets before the build process begins.

  Default: no clean.

- `/BUILD_CONFIG=<build-configuration-name>`

  Specifies the build configuration. If this is not specified, all build configurations associated with the Dimensions project/stream are used.

- `/BUILD_OPTIONS=(<opt1>=<value1>,<opt2>=<value2>,...)`

  Specifies build options, for details see *Dimensions Build User-Defined Optional Symbols* in *The Templating Language and Processor* chapter of the *Developer's Reference*.

- `CAPTURE`

  Specifies whether built targets are to be collected. It is not possible to collect default targets when building at the DEVELOPMENT stage.

  Default: no capture.

- `/CHANGE_DOC_IDS=(<request1>,<request2>, ...)`

`<requestN>`	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.

- `/TARGETS=<targets-list>`

  Specifies the list of targets to be built. If this is omitted, all targets for the project/stream are built. If both sources and targets are specified, the command will use a union of these two specifications to determine which targets are to be built.

  Note: This specifies not target names (which is a general string in the build configuration) but the would-be project file names, or wild-card specifications in the

distributed format (not the platform format). So, for MVS files, examples would be XML/*.XML and not XML(*).

- /USER_FILENAME=<file-name>

  This optional qualifier specifies a file that will be created in the user area to store the build results. If this qualifier is not specified, the build result information is returned to the command client as message text. For example:

```
Dimensions>BLD  "REPX:REPX" /BUILD_CONFIG="win 32 buildme;6" /
    AREA="WS_3" /NOCAPTURE /NOBATCH /NOAUDIT
Current status of build job 691 :       SUCCESS
Open this link in a browser for more details
<link>

Operation completed
```

- /SRC_CHANGE_DOC_IDS=(<request1>,<request2>,...)

  Specifies requests, which are related to sources, that will be part of a build request. Causes the build to be driven by the sources, though you can also specify targets with the /TARGETS or /TARGETS_LIST qualifiers.

- /SRC_FILES=(<item-filename1>,<item_filename2>,...)

  Specifies filenames that will be a part of a build request. Causes the build to be driven by the sources, though you can still specify targets with the /TARGETS or /TARGETS_LIST qualifiers. If both sources and targets are included the command will use a union of these two specifications to determine which targets are to be built.

  Note: the source files must be specified in platform format. So, for MVS files examples would be names like PLI(*).

- /SRC_ITEMS=(<item-spec1>,<item-spec2>,...)

  Specifies filenames, by item specification, that will be a part of a build request. Causes the build to be driven by the sources, though you can still specify targets with the /TARGETS or /TARGETS_LIST qualifiers.

- /SRC_FILELIST=[<node>::]<filespec>

  Specifies a file containing a list of source files that will be a part of a build request. You can use the node:: syntax to specify a Dimensions tertiary node. This causes the build to be driven by the sources, though you can also specify targets with the /TARGETS or /TARGETS_LIST qualifiers.

  This is an alternative to specifying the /SRC_FILES, /SRC_ITEMS, /SRC_CHANGE_DOC_IDS, or /SRC_ITEMLIST qualifiers.

  File names are assumed to be one per line. Leading and trailing spaces are removed. There must be no blank lines.

- /SRC_ITEMLIST=[<node>::]<filespec>

  Specifies a file containing a list of source files by item specification that will be a part of a build request. You can use the node:: syntax to specify a Dimensions tertiary node. This causes the build to be driven by the sources, though you can also specify targets with the /TARGETS or /TARGETS_LIST qualifiers.

  This is an alternative to specifying the /SRC_FILES, /SRC_ITEMS, /SRC_CHANGE_DOC_IDS, or /SRC_FILELIST qualifiers.

Item specifications are assumed to be one per line. Leading and trailing spaces are removed. There must be no blank lines.

Item specifications are either relative to the project, or the current working location will be removed to obtain the project filename. This must match the item specification.

- `/TARGETS_LIST=[<node>::]<filespec>`

  Specifies a file containing a list of targets to be built. You can use the `node::` syntax to specify a Dimensions tertiary node. This is an alternative to specifying the /TARGETS qualifier. If both this and the /TARGETS qualifier are omitted, all targets for the project/stream are built.

  Note: This specifies not target names (which are general strings in the build configuration) but the would-be project file names, or wild-card specifications in the distributed format (not the platform format). So, for MVS files, examples would be XML/*.XML and not XML(*).

- `/[NO]CANCEL_TRAVERSE`

  Specifies whether to traverse, or include, the child requests of the requests specified in /SRC_CHANGE_DOC_IDS.

  Default: yes (include all child request)

- `/POPULATE_SCOPE=NONE|ALL|REQUESTED`

  (Work areas only) Specifies whether the work area is populated before submitting the build. Set NONE to not populate the work area. Set ALL to populate the work area with all items.

  Default: no

- `/[NO]TOUCH`

  (Can only be used with /POPULATE) Specifies whether to apply the system date/time to each file being populated in a work area.

  Default: no

- `/[NO]LOCK_SEARCH_PATH`

  Requires that the search path be locked for all work and deployment areas, in addition to the build area, starting with the stage specified by the DM_SP_START_STAGE variable on the logical node.

  Default: no

- `/TARGET_PROJECT`

  Specifies a different project (within the same product) for the collected / built objects to be collected to.

- /USER=userid

  Defines the user ID required to launch a build in a deployment area. User credentials are not required for work areas.

- /PASSWORD=password

  Defines the password required to launch a build in a deployment area. User credentials are not required for work areas.

- /DEPENDENCY_ANALYSIS_FLAGS=(list)

  Controls the selection of build targets where (list) can contain:

  - [NO]FINAL: Return intermediate and final targets. Default: NOFINAL
  - [NO]SOFT: Return predicted targets. Default: SOFT
  - [NO]DEPS: Run dependency analysis. Default: DEPS
  - [NO]ALLTARGETS: Return all targets but do not preselect them. Default: ALLTARGETS
  - [NO]TARGETS: Controls the processing of all target information from the get dependencies and get targets logic. If it is turned off no target information is returned. Default: TARGETS
  - USESELECTED (force non-selected targets to be included in a build) or USEALL (use all targets when selecting targets for a build). Default: USESELECTED
  - [NO]SIDEFFECTS: Request side effect targets from dependency analysis. Default: NOSIDEFFECTS
  - [NO]CONFIG: Analyze the build configuration. Default: CONFIG
  - [NO]FOREIGN: Include foreign targets. Default: FOREIGN
  - LIST: Display all the values, and their defaults, that the list can contain.

  Important: This qualifier cannot be used with /TARGETS or /TARGETS_LIST.

  Target analysis is a general term that is applied to two interrelated pieces of processing. The results of the target analysis are passed to the Java build processing to guide its logic.

Desired outcome	TARGETS flag	CONFIG flag	DEPS flag
Maximum target analysis. Bill of Materials processing identifies likely built objects based on: ■ Previous builds. ■ Full analysis of the build configuration. Built object names and rules are returned to the user or function.	TARGETS	CONFIG	DEPS
BOM processing identifies the build object item names based on made-of records only.	NOTARGETS	CONFIG or NOCONFIG	DEPS

Desired outcome	TARGETS flag	CONFIG flag	DEPS flag
BOM processing identifies the likely target objects based on previous builds and gives the related rule names.	TARGETS	NOCONFIG	DEPS
No dependency analysis is performed. Configuration analysis is run and built objects and related rules are returned. If the source implies an applicable rule, the rule is returned without the built object name(s).	TARGETS	CONFIG	NODEPS
No useful work is done by the dependency analysis. The Java build performs its analysis however the results may be unpredictable.	NOTARGETS	NOCONFIG	NODEPS

Description

The BLD command:

- Builds targets that are defined for the specified project/stream and that are available at the specified build stage.
- Optionally, collects built targets and creates relationships between them and the sources used to build them.

  By default, the collected targets will be:

  - Created in the development project from which the build request was initiated. The default product-specific Dimensions upload rules will be used to derive Dimensions data format and item type information. For more information about upload rules, refer to the *Process Configuration Guide*.
  - Created at the initial lifecycle state, which would normally be associated with the DEVELOPMENT stage. If the lifecycle of the item type of a built target is mapped to build stages (such as UT, ST, and REL), then Dimensions will also promote the built target to the corresponding build stage, and will place the resulting item revision into the corresponding project-stage project.

The sources to be built can be specified using one of the following qualifiers: /SRC_FILES, /SRC_ITEMS, /SRC_CHANGE_DOC_IDS, /SRC_FILELIST or /SRC_ITEMLIST. The targets to be built are specified using either the /TARGETS or /TARGETS_LIST qualifiers. If both sources and targets are specified, the command will use a union of these two specifications to determine which targets are to be built.

For more information about using Dimensions Build see the *Dimensions CM Build Tools User's Guide.*

BLDB – Build Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
/AREA=<area-name>
[/[NO]WAIT]
[/BUILD_CLEAN]
[/BUILD_CONFIG = <build-configuration-name>]
[/BUILD_OPTIONS = (<opt1>=<value1>,<opt2>=<value2>,...)]
[/[NO]CAPTURE]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/TARGETS=<targets-list>]
[/USER_FILENAME = <file-name>]
[/TARGETS_LIST=[<node>::]<filespec>]
[/[NO]TOUCH]
[/[NO]LOCK_SEARCH_PATH]
[/TARGET_PROJECT=[product:]projectname]
[/USER=userid]
[/PASSWORD=password]
```

Example

```
BLDB "PAYROLL:ACME_2.1_TSTBLN"
   /TARGETS=("aix/foo", "aix/libfoo.so")
   /BUILD_OPTIONS=(DMBLDMAKE_OPTIONS="-s -ov")
   /CAPTURE
   /CHANGE_DOC_IDS=(PVCS_CR_1234)
   /USER_FILENAME=D:\temp\bld.log
```

Parameters and qualifiers

- `<baseline-spec>`

  Specifies the baseline to be built.

- `/AREA=<area-name>`

  Specifies the Dimensions work area to be used for the build.

- `/WAIT`

  Specifies that the command will wait for the build to finish.

  Specify /NOWAIT for the build to be run in batch mode (the command does not wait for the build to finish).

  Default: wait.

- `/BUILD_CLEAN`

  Specifies that the area is to be cleaned of targets before the build process begins.

  Default: no clean.

- `/BUILD_CONFIG=<build-configuration-name>`

  Specifies the build configuration. If this is not specified, all build configurations associated with the Dimensions project/stream are used.

- /BUILD_OPTIONS=(<opt1>=<value1>,<opt2>=<value2>,...)

  Specifies any build options.

- CAPTURE

  Specifies whether built targets are to be collected. It is not possible to collect default targets when building at the DEVELOPMENT stage.

  Default: no capture.

- /CHANGE_DOC_IDS=(<request1>,<request2>, ...)

  <requestN> 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.

- /TARGETS=<targets-list>

  Specifies the list of targets to be built. If this is omitted, all targets for the project are built.

  Note: This specifies not target names (which is a general string in the build configuration) but the would-be project file names, or wild-card specifications in the distributed format (not the platform format). So, for MVS files, examples would be XML/*.XML and not XML(*).

- /USER_FILENAME=<file-name>

  This optional qualifier specifies a file that will be created in the user area to store the build results. If this qualifier is not specified, the build result information is returned to the command client as message text. For example:

```
Dimensions>BLD  "REPX:REPX" /BUILD_CONFIG="win 32 buildme;6" /
    AREA="WS_3" /NOCAPTURE /NOBATCH /NOAUDIT
Current status of build job 691 :       SUCCESS
Open this link in a browser for more details
<link>

Operation completed
```

- /TARGETS_LIST=[<node>::]<filespec>

  Specifies a file containing a list of targets to be built. You can use the node:: syntax to specify a Dimensions tertiary node. This is an alternative to specifying the /TARGETS qualifier. If neither this or the /TARGETS qualifiers are specified, all targets for the project/stream are built.

  Note: This specifies not target names (which are general strings in the build configuration) but the would-be project file names, or wild-card specifications in the distributed format (not the platform format). So, for MVS files, examples would be XML/*.XML and not XML(*).

- /[NO]TOUCH

  Specifies whether to apply the system date/time to the files that are transferred prior to build.

  Default: no

- /[NO]LOCK_SEARCH_PATH

  Requires that the search path be locked for all work and deployment areas, in addition to the build area, starting with the stage specified by the DM_SP_START_STAGE variable on the logical node.

  Default: no

- /TARGET_PROJECT

  Allows you to specify a different project (within the same product) for collected / built objects to be collected to.

- /USER=userid

  Defines the user ID required to launch a build in a deployment area. User credentials are not required for work areas.

- /PASSWORD=password

  Defines the password required to launch a build in a deployment area. User credentials are not required for work areas.

- /DEPENDENCY_ANALYSIS_FLAGS=(list)

  Controls the selection of build targets where (list) can contain:

  - [NO]FINAL: Return intermediate and final targets. Default: NOFINAL
  - [NO]SOFT: Return predicted targets. Default: SOFT
  - [NO]DEPS: Run dependency analysis. Default: DEPS
  - [NO]ALLTARGETS: Return all targets but do not preselect them. Default: ALLTARGETS
  - [NO]TARGETS: Controls the processing of all target information from the get dependencies and get targets logic. If it is turned off no target information is returned. Default: TARGETS
  - USESELECTED (force non-selected targets to be included in a build) or USEALL (use all targets when selecting targets for a build). Default: USESELECTED
  - [NO]SIDEFFECTS: Request side effect targets from dependency analysis. Default: NOSIDEFFECTS
  - [NO]CONFIG: Analyze the build configuration. Default: CONFIG
  - [NO]FOREIGN: Include foreign targets. Default: FOREIGN
  - LIST: Display all the values, and their defaults, that the list can contain.

  Important: This qualifier cannot be used with /TARGETS or /TARGETS_LIST.

Target analysis is a general term that is applied to two interrelated pieces of processing. The results of the target analysis are passed to the Java build processing to guide its logic.

Desired outcome	TARGETS flag	CONFIG flag	DEPS flag
Maximum target analysis. Bill of Materials processing identifies likely built objects based on: ■ Previous builds. ■ Full analysis of the build configuration. Built object names and rules are returned to the user or function.	TARGETS	CONFIG	DEPS
BOM processing identifies the build object item names based on made-of records only.	NOTARGETS	CONFIG or NOCONFIG	DEPS
BOM processing identifies the likely target objects based on previous builds and gives the related rule names.	TARGETS	NOCONFIG	DEPS
No dependency analysis is performed. Configuration analysis is run and built objects and related rules are returned. If the source implies an applicable rule, the rule is returned without the built object name(s).	TARGETS	CONFIG	NODEPS
No useful work is done by the dependency analysis. The Java build performs its analysis however the results may be unpredictable.	NOTARGETS	NOCONFIG	NODEPS

Description

Builds targets from baselines, and optionally collects built targets and creates relationships between the collected targets and sources used to build these targets.

By default, the collected targets will be created in the project from which the build request was issued. The default product-specific Dimensions upload rules will be used to derive Dimensions data format and item type information. For more information about upload rules, refer to the *Process Configuration Guide*. The collected targets will be created at the initial lifecycle state.

The targets to be built are specified using either the /TARGETS or /TARGETS_LIST qualifiers. If both sources and targets are specified, the command will use a union of these two specifications to determine which targets are to be built.

For more information about using Dimensions Build see the *Dimensions CM Build Tools User's Guide.*

CA – Create Area

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
[/DESCRIPTION=<area-description>]
/NETWORK_NODE=<node-name>
/DIRECTORY=<HLQ/directory>
[/TYPE=<area-type>]
[/STAGE=<stage-name>]
[/USER=<user-name or credential set> [/PASSWORD=<password>]]
[/LIBRARY_CACHE_AREA=<area-name>]
[/[NO]FETCH_EXPANDED]
[/TRANSFER_SCRIPTS=<script-set>]
[/SCRIPT_PARAMETERS=(<name1>=<value1>,<name2>=<value2>,...)]
[/OWNER=<user-name> or <group-name>]
[/USER_LIST=(<user-or-group>,<another-user-or-group>,...)]
[/STATUS=ONLINE or OFFLINE]
[/FILTER=<area-filter>]
```

Example

```
CA <area-name> /NETWORK_NODE=<host-machine> /DIRECTORY=<area-directory>
    /TYPE=WORK USER_LIST=(<user1>,<user2>,<user3>)
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area. Area names must be unique within the base database.

- `/DESCRIPTION=<description>`

  Optional. Specifies a description for the new area.

- `/NETWORK_NODE=<node-name>`

  Specifies the machine hosting the area.

- `/DIRECTORY=<HLQ/directory>`

  Specifies the directory, or PDS (partitioned data set), where the area is located.

NOTE You cannot assign the same location (the same network node and directory) to more than one area. An error will occur if this location has already been assigned.

HLQ is a high-level qualifier; for example, MERVK.WORK. It is a common prefix for all data sets in the area such as MERVK.WORK.C, MERVK.WORK.CBL, and so forth.

CAUTION! When specifying the `/DIRECTORY` parameter on Windows and UNIX platforms, you need to specify an absolute path value; otherwise, the area will be created as a sub-directory of the directory specified by `%DM_TMP%` (Windows) and `$DM_TMP` (UNIX) in the dm.cfg file. By default, `%DM_TMP%` and `$DM_TMP` are set to `%TMP%\` and `/tmp` respectively.

- `/TYPE=<area-type>`

  Specifies the type of the area: WORK or DEPLOYMENT.

  Below is a mapping between Dimensions 9 and Dimensions 10 area types:

Dimensions 9	Dimensions 10
Development Area (working location)	Work Area
Managed Development Area	Deployment Area associated with stage DEVELOPMENT
Build Area associated with stage <XXX>	Deployment Area associated with stage <XXX>

- `/STAGE=<stage-name>`

  Applicable only to deployment areas. If the area type is DEPLOYMENT, this qualifier specifies the stage with which the new deployment area is associated.

- `/USER=<user-name or credential-set-name> [/PASSWORD=<password>]`

  Login information for the operating system user account or credential set that will own files transferred into the area. You do not need to specify a password if you use a credential set.

  For more information about credential sets see the *System Administration Guide*.

- `/LIBRARY_CACHE_AREA=<area-name>`

  Specifies a library cache area defined with the CLCA (Create Library Cache Area) command. During fetch operations (FI, FWI, FBI, FCDI, EI, EWI, EBI, ECDI, DOWNLOAD), Dimensions checks whether the library cache area associated with the current project/stream already contains a copy of of the requested file. If so, Dimensions copies the file from the library cache to the user file area instead of from the library itself, which improves performance when the connection between the item library node and the user's network is slow.

- `/[NO]FETCH_EXPANDED`

  Specifies whether item header substitution variables will be expanded when item files are fetched to the area. Default is /FETCH_EXPANDED.

- `/TRANSFER_SCRIPTS=<script-set>`

  Applicable only to the DEPLOYMENT area type. Specifies the transfer script set. The script set contains a comma-separated list of the names of pre/post/fail transfer scripts in the following format:

  `(<pre-script>,<post-script>,<fail-script>)`

  If one of the scripts is undefined, CA uses $NONE as a placeholder. The pre-script is executed before an item is transferred into an area, the post script is executed after an item is successfully transferred into an area, and the fail script is executed after a failed transfer of all items into an area.

- /SCRIPT_PARAMETERS

  Specifies a list of keyword and values, and arrays of values, to be passed as script parameters. The keyword and values must be comma separated. For example:

  ```
  /SCRIPT_PARAMETERS="NAME=value","NAME=value"

  /SCRIPT_PARAMETERS="ARRAY=[value1,value2]"
  ```

  Names in lowercase are converted to uppercase during execution. Names in templates must be written in uppercase, for example: %NAME1. %NAME2. For details see the "Templating Language and Processor" chapter of the *Developer's Reference*.

- /OWNER=<user-name> or <group-name>

  Optional. Specifies the user or group that is to become the owner of the new area. If /OWNER is not specified, the user who created the area is set as its owner. The owner of the area has the right to manage the definition of the area.

- /USER_LIST=(<user-or-group>,<another-user-or-group>,...)

  Applicable only to the WORK area type. Specifies the list of users and/or groups that are granted the right to use this work area. If the user list of an area is empty, any user can specify the area as the working location and can get or check out items into the area and check in items from the area. If the user list of an area is not empty, only the listed user and members of the listed groups can use the work area as a target of get, check-out, and check-in operations.

- /STATUS=ONLINE or OFFLINE

  Applicable only to the DEPLOYMENT area type. Specifies the status of the area. If the area's status is ONLINE, the area may participate in file transfer operations. If the area's status is OFFLINE, the area is automatically excluded from any file transfer operations. If this qualifier is not specified, an area with status ONLINE is created.

- /FILTER=<area-filter>

  Applicable only to the DEPLOYMENT area type. Specifies the name of the area filter to be used when deploying files into this area.

Description

The CA command creates an area definition.

Constraints

To create a work area, you must have the Create Work Areas privilege. To create a deployment area, you must have the Create Deployment Areas privilege.

In a pure LDAP environment this command requires a local operating-system account.

CAR – Create Archive

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<archive-id>
/BASELINE=<baseline-spec>
/DEVICE=<device-id> or /DEVICE=NONE
/TAPE=<tape no.>
/VOLUME=<volume-id>
[/FORCE]
[/DESCRIPTION=<description>]
[/DIRECTORY=<directory>]
[/[NO]REPORT]
```

Example

```
CAR AA12AB /BASELINE="PRODX:BL12AB" -
   /DEVICE="/dev/rmt0h" /TAPE="aa100" /VOLUME="bb100" -
   /DIRECTORY="/usr/smith/work"-
   /DESCRIPTION="Archive of 12AB - sources"
```

See the *System Administration Guide* for details.

CBA – Create Build Area

NOTE This command is no longer available; use CA (Create Area) instead.

See the CA command.

CBDB – Register a Base Database Entry

```
/BDB_NAME=<base_db_name>
/PCMS_VER=<dimensions_version>
/CAP_REPLICATE=<project_replication_y/n>
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
[/DESCRIPTION=<base_db_description>]
[/PCMS_ROOT_DIR=<dimensions_root_diectory>]
[/CO_NAME=<contact_name>]
[/SITE_NO=<site_no>]
```

Example

```
CBDB /BDB_NAME=SERENA-PCMS
   /NN_NAME=MACHINE.COMPANY.COM
   /DB_SERVICE=PCMSUDB
   /CAP_REPLICATE=N
   /PCMS_ROOT_DIR="DIMENSIONS ROOT DIRECTORY "
   /PCMS_VER="DIMENSIONS 9.1"
   /DESCRIPTION="DETAILS OF BASE DATABASE FOR NODE MACHINE.COMPANY.COM"
```

This command enables you to register base database entries in an installation's network administration tables. See the *System Administration Guide* for details.

CBL – Create Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
[/PART=<part-spec>]
[/TEMPLATE_ID=<template-id>]
[/TYPE=<baseline-type>]
[/SCOPE=WORKSET | PART]
[/WORKSET=<project-spec>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/LEVEL=<integer>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/[NO]CANCEL_TRAVERSE]
[/[NO]INCLUDE_INFO]
[/[NO]INCLUDE_CLOSED]
[/BASELINE]
[/SCOPE_TO_WS]
[/REQUIREMENT_IDS=(<requirement_spec1>{container_name+project_name+
    dbname+rm_browser_url},<requirement_spec2>{container_name+
    project_name+dbname+rm_browser_url},...)]
[/DESCRIPTION=<description>]
[/USER_FILTER=<filter-file-spec>]
[/[NO]STRICT]
```

Examples

```
CBL PROD:"R M VERSION 2 FOR HP"
   /TEMPLATE=SOURCE_DOC
   /PART=PROD:"RELEASE MANAGEMENT".AAAB
```

```
CBL "REPX:RB1" /PART="REPX:REPX.A;1" /TEMPLATE_ID="CHDOC_TEMPLATE1"
/WORKSET="REPX:REPX" /LEVEL="0" /TYPE="BASELINE" /SCOPE="PART"
/REQUIREMENT_IDS=("Marketing_Requirements.MRKT_000020;79{Ephoto Hot
    List+RMDEMO6+RM10+http://mars/rtmBrowser/}",
    "Marketing_Requirements.MRKT_000002;54{Ephoto Hot
    List+RM10+RMDEMO7+http://mars/rtmBrowser/}")
```

Common baseline types

- Release Baseline

  Only includes a single revision of each item and is suitable for deployment.

- Design Baseline

  Used for backup and replication and may include many or all revisions of items. Is a snapshot of the current stage of development and does not include checked-out revisions.

Parameters and qualifiers

- `<baseline-spec>`

  which comprises:

  `<product-id>:<baseline-id>`

`<baseline-id>`	Specifies the identity of the new baseline.

- `/PART=<part-spec>`

  which comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	May be omitted if only one exists.
`<pcs>`	Is ignored (the current PCS is always used).

- `/TEMPLATE=<template-id>`

  The identity of the item or request baseline-template. You must specify a template to create a release baseline or use `/SCOPE=WORKSET`.

  Default: design baseline

- `/TYPE=<baseline-type>`

  Specifies the type of baseline to be created.

  Default: BASELINE

- `/SCOPE=<scope-name>`

  where:

  - `/SCOPE=WORKSET`: creates a project baseline.

    You can restrict the selection of item revisions to be included in the baseline by specifying a file filter using the `/USER_FILTER` qualifier.

  - `/SCOPE=PART`: creates a design part scoped baseline. A project baseline is always owned by the product that owns the project.

- `/WORKSET=<project-spec>`

  where `<project-spec>` is:

  `<product-id>:<project-id>`

  If you use `/WORKSET` only the items in the specified project or stream are considered for baselining. If you do not use this qualifier only the items in the user's current project or stream are considered. See the LCK command (page 266) about locking a project when baselining.

  If you use `/SCOPE=WORKSET` the `/WORKSET` qualifier specifies the project or stream to be baselined and not the project or stream used to constrain the part-based revision selection.

IMPORTANT! Items that have an In Response To relationship to a change request are included even if they are not in the project or stream specified by `/WORKSET`.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

  <attrN> 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>.

  <valueN> The substitution value to be given to this attribute.

- /LEVEL=<integer>

  Restricts the baseline to a specified number of levels in the design tree structure.

  Default: 0 (all levels below the design part selected by /PART).

  For example, '1' specifies that only items related to the selected design part are processed.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  <requestN> Identifies a request to which the new baseline will have an InResponseTo relationship.
  If the template specified is a request template, the request identification is overridden to specify the list of parent requests used for the CBL command.

- /[NO]CANCEL_TRAVERSE

  Halts the traversal of dependent requests. By default, requests that are related as dependent are processed by the CBL command.

  If you specify /SCOPE=WORKSET this qualifier controls the baselining of child collections.

  Default: /NOCANCEL_TRAVERSE

- /[NO]INCLUDE_INFO

  Includes items related to requests via an Info relationship. By default, only items that are related to requests via an InResponseTo relationship are included.

  If you specify/SCOPE=WORKSET 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.

  Default: /NOINCLUDE_INFO

- /[NO]INCLUDE_CLOSED

  Includes closed requests when processing requests for request baselines. This qualifier overrides the default setting for baselines based on baseline templates that use the SUP status code (specified state or next existing state upward). The SUP status code normally excludes any closed requests. For information about status codes and baseline templates see "Request Baseline Templates" on page 86.

  Default (do not include requests): /NOINCLUDE_CLOSED

- /BASELINE

  If you are creating a new baseline via a request baseline template, this qualifier creates a reference to which change documents identified by a template can be applied to revise the baseline. This causes the CBL command to behave like a request template driven CRB command. However, the processing is different and the results may not be the same.

- `/SCOPE_TO_WS`

  Limits the scope of the CBL command to the current project or the project specified by `/WORKSET`. This is the default for request type baselines. Use `/NOSCOPE_WS` to remove the limitation.

- `[/REQUIREMENT_IDS=(<requirement_spec1>{container_name+project_name+dbname+rm_browser_url},<requirement_spec2>{container_name+project_name+rm_browser_url},...)]`

  Specifies a comma separate list of Dimensions RM requirements. Each requirement is comprised of:

  `<requirement_spec>{container_name+project_name+dbname+rm_browser_url}`

  where:

<requirement_spec>	Comprises: `<class_name>.<puid>;objId` For example: `Marketing_Requirements.MRTK_000020;4`.
container_name	The name of the originating Dimensions RM container (baseline, collection, document, or snapshot) for the requirement. Multiple "versions" of a requirement cannot be related to a single Dimensions CM request. For example: `Ephoto Hot List`.
`project_name`	Specifies the Dimensions RM project name, for example: `RMDEMO6`
`dbname`	Specifies the Dimensions RM database name, for example: `RM10`
`rm_browser_url`	Specifies the RM Browser URL, for example: `http://mars/rtmBrowser/`.

  Example of a requirement_id:

  ```
  "Marketing_Requirements.MRKT_000020;79{Ephoto Hot
      List+RMDEMO6+RM10+http://mars/rtmBrowser/}"
  ```

IMPORTANT! You can only specify Dimensions RM requirements if you have:

- Installed the Dimensions RM integration.
- Associated Dimensions CM projects and streams with Dimensions RM containers.
- Have associated Dimensions CM products with Dimensions RM projects.

See the "Miscellaneous Functions" chapter in the Dimensions CM *User's Guide*, the *Dimensions CM-Dimensions RM ALM Integration Guide*, and the Dimensions RM documentation for details.

- `/DESCRIPTION=<description>`

  Describes the baseline.

- `/USER_FILTER=<filter-file-spec>`

  Specifies the name of a local file containing a filter definition. The baseline will only contain item revisions that satisfy the criteria specified in the filter. The format of the filter and an example are described on page 502.

- /STRICT

  If any item does not have a common ancestor, the baseline is not created.

  Default: /NOSTRICT

  Example:

```
CBL "QLARIUS:BAS02"
/STRICT
/PART="QLARIUS:QLARIUS.A;1"
/TEMPLATE_ID="JM"
/WORKSET="QLARIUS:DEF225963"
/LEVEL="0"
/TYPE="BASELINE"
/CHANGE_DOC_IDS=("QLARIUS_CR_43","QLARIUS_CR_42","QLARIUS_CR_41",
    "QLARIUS_CR_40","QLARIUS_CR_39",)
/SCOPE="PART"
PCM5200001E Error: Unable to find a common descendent amongst
    revisions of item "QLARIUS:JM2013 SRCBAT-195916602X636X0.A-SRC"
    selected for update. The item will not be updated.
```

Request Baseline Templates

Request baseline templates enable you to specify rules for selecting requests to be used as input for creating baselines. The templates comprise one or more rules that are made up from the following:

- Request type
- Request status
- Baseline status code, which is comprised of one of the following keys:
  - EQS – specified state only.
  - SUP – specified state or next existing state upward.

NOTE Baseline collective codes such as *MADE_OF and *LATEST, which are used in item baseline templates, are not applicable to requests baselines.

A request baseline template consists only of request template rules and you cannot add rules using item types. Conversely, you cannot add request baseline template rules to an item baseline template. To enforce this separation, the same template identifier cannot be used to create an item baseline template and a request baseline template.

Using Request Baseline Templates with CBL

When you create a baseline using the CBL command, and specify a request baseline template and a set of starting parent requests, all the requests that meet the following conditions are processed:

- Are related to the parent requests
- Match the template rules

The baseline template rules are processed the same way as item templates: requests are selected based on the type, status, and the baseline status code that was specified. For example, if a template has a rule that considers:

- all requests of the type PR
- at status ACCEPTED
- with the baseline status code EQS

then all requests that satisfy these conditions are included in the baseline.

After this list of requests has been determined, items that are related to requests with an InResponseTo or an Info relationship are also included into the baseline. However, because the baseline that is being created is a release baseline, only one revision of each item is included. If there are multiple revisions of the same item, only the latest item revision is selected using that item's pedigree. If item revisions are in conflict and no common successor items are found, CBL fails with an appropriate error message.

When the baseline has been created, the requests that were used to create it are related with an InResponseTo relationship the new baseline.

Constraints

- Baselines do not include items that are in an off-normal state.
- To create a baseline you must have the appropriate management privileges for the baseline's top design part that are required to action it from its initial lifecycle state to a new state. However, if a user with the `PRODUCT-MANAGER` role has assigned the top design part `$ORIGINATOR` role to the first transition in the lifecycle for this baseline type, any Dimensions user can create a baseline of this type provided they have any role on the top design part on which the baseline is being created.
- To enforce a consistent model of behavior for item baselines, the following additional constraints apply to request baselines:
  - Requests that are either at a closed or off-norm lifecycle state are not processed by the SUP baseline code.
  - Only requests in the primary catalog are processed.
  - Only requests related through a dependency relationship, DEPEND, are included in traversal scans.
  - If a request related through a dependency relationship does not fulfill the criteria specified in the request template rules, that request is ignored including every other request that is a child of the request. For example, multiple levels of requests are related together in a chain through dependent relationships:

    CR_1 → CR_5 → CR_7 → CR_9 → CR_12

    If CR_7 fails to match a template rule, then CR_7, CR_9, and CR_12 are ignored in any further processing.
  - Only requests that are owned by the product on which the baseline is being created are processed. Any requests owned by other products are ignored, including any child requests.
  - Only parts or items that are owned, or have usage relationships to the parent part specified in the CBL command, are included in the final baseline.
  - If a request refers to affected parts and/or items that may be out of scope, these parts and items are ignored. Out of scope parts and items are not owned by, or related to, the parent part or any of its children.

CBP – Copy Build Project

```
/SOURCE_PROJECT=<project-spec>
[/TARGET_PROJECT=<project-spec>]
/SOURCE_CONFIG=<build-configuration-name>
[/TARGET_CONFIG=<build-configuration-name>]
[/[NO]COPY_CONFIGS]
[/[NO]FORCE]
```

Examples

Copy all build configurations and relationships from PROJA to PROJB:

```
CBP /SOURCE_PROJECT=QLARIUS:PROJA
   /TARGET_PROJECT=QLARIUS:PROJB
CBP /SOURCE_PROJECT=QLARIUS:PROJA
   /TARGET_PROJECT=QLARIUS:PROJB
   /SOURCE_CONFIG="
CBP /SOURCE_PROJECT=QLARIUS:PROJA
   /SOURCE_CONFIG="
   /TARGET_CONFIG="
```

Parameters and qualifiers

- `/SOURCE_PROJECT=<project-spec>`

  Specifies the project/stream from which to copy the build information.

  comprises:

  `<product-id>:<project-id>`

- `/TARGET_PROJECT=<project-spec>`

  Optionally, specifies the project/stream to which the build information is to be copied.

  comprises:

  `<product-id>:<project-id>`

  If this is omitted, it is assumed that the target project/stream is the same as the source project/stream

- `/SOURCE_CONFIG=<build-configuration-name>`

  Specifies the name of the build configuration to be copied. If this is not specified, all build configurations associated with the Dimensions project/stream are copied.

- `/TARGET_CONFIG=<build-configuration-name>`

  Specifies the name of the target build configuration to which the configuration specified by /SOURCE_CONFIG is to be copied.

  This can only be specified if /TARGET_PROJECT is the same project as /SOURCE_PROJECT and /SOURCE_CONFIG has been specified.

- `/[NO]COPY_CONFIGS`

  Specifies whether build configurations will be copied from from /SOURCE_PROJECT to /TARGET_PROJECT.

  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.

- `/[NO]FORCE`

This option specifies whether the CBP will abort when the first error is encountered, or whether the process will continue to produce as many diagnostic messages as possible. /NOFORCE 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 /NOFORCE.

Description

This command copies build information from one existing stream or project to another or copies build configurations within the same stream or project.

CC – Create Request

```
<product-id>
<request-type>
[/WORKSET=<project-spec>]
[/BASED_ON=<request-id>]
[/DESCRIPTION=<desc-file>]
[/AFFECTED_PARTS=(<part-spec>,<part-spec>,...)]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/RELATIONSHIP=<rel_name>]
[/[NO]HOLD]
[/ATTACHMENTS=([USER_FILE=<user-file>, FILENAME=<file-id>,
    DESCRIPTION=<description-text>], ...)]
[/BASELINE_LIST=(<baseline1>,<baseline2>,...)]
[/[NO]EXCLUSIVE_LOCK]
```

Example

```
CC PROD DR
   /DESC=qrel_subdir.desc
   /AFFECT=PROD:"RELEASE MANAGEMENT".AAAA
   /ATTRIB=(TITLE="QREL Subdir problem",SEVERITY=3)Sub
   /ATTACHMENTS=([USER_FILE=C:\Attachments\Figure1.jpg,
                  FILENAME=Figure1.jpg,DESCRIPTION="first page"])
```

Parameters and qualifiers

- `<product-id>`

  Specifies the product that will own the new request.

- `<request-type>`

  Specifies the type of request to be created.

- `/WORKSET=<project-spec>`

  Comprises `<product-id>:<project-id>` and specifies a project or stream with which the request will be associated. When a request is created with an owning project or stream, the Stage ID for the request is set to the initial stage in the Global Stage Lifecycle. If you do not specify `/WORKSET` no project or stream is associated with the request.

- `/BASED_ON=<request-id>`

  Bases ('primes') the creation of the new request on the attributes of the request given by `<request-id>`. If omitted, the new request's data is derived from the other parameters specified in this command.

- `/DESCRIPTION=<desc-file>`

  Specifies a plain text file containing a detailed description of the request. Binary formats, such as Microsoft Word, are not supported.

- `/AFFECTED_PARTS<part-spec>`

  comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	May be omitted if only one exists.
`<pcs>`	Is ignored (the current PCS is always used).

  Specifies one or more design parts to be related to the new request.

If /AFFECTED_PARTS is omitted, the product's top design part is related by default.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

<attrN>	The variable name defined for one of the 220 user-defined attributes for requests, which has also been declared as usable for this <product-id> and <request-type>.
<valueN>	The substitution value to be given to this attribute.

- /RELATIONSHIP=<rel_name>

  Specifies the relationship type between the new request and the base request. The relationship is a bi-directional link with the new request as the child and the base request as the parent. Valid only if the qualifier /BASED_ON is used.

- /HOLD

  Specifies that the new request is to be placed on the Held List before being entered into the system.

  Default: /NOHOLD (the new request is saved and entered into the system).

NOTE

The <request-id> generated by CC may be referenced as $LAST by a subsequent command in a CMD command script.

- /ATTACHMENTS=([USER_FILE=<user-file>,FILENAME=<file-id>, DESCRIPTION=<description-text>], ...)

  Specifies a file, or files, to be attached to the request when it is created.

USER_FILE=<user-file>	The name of the user file from where the attachment is to be loaded.
FILENAME=<file-id>	The name of the file to be attached.
DESCRIPTION=<description-text>	A description of the attachment.

- /BASELINE_LIST=(<baseline1>,...)

  Identifies one or more existing release baselines that the request will be related to (as Affected) when the request enters the system. Has the following properties:

  - When a CC command is run without /AFFECTED_PARTS, the request is automatically related to the current open PCS of the part owning the baseline. When multiple baselines are specified, the owning parts are also related as if multiple affected parts had been specified.
  - When a CC command is run with /AFFECTED_PARTS and /BASELINE_LIST specified, the parts affected are a merged list of the parts listed in the /AFFECTED_PARTS qualifier and the parts that own the baselines.

- /EXCLUSIVE_LOCK

  Locks the new request against any request (issue) replication "requests" from users located on other replication sites. A locked request is available for users to work on normally if they are located on the owning replication site. Users on the owning site where that locked request was created can continue to use the new request normally. Default: /NOEXCLUSIVE_LOCK (the new request can be requested from any authorized replication site).

Constraints

This command can be run by all users. However, Dimensions can be configured so that users must have a role on the product before they can create requests.

Requests that were created in a held state are not considered to have been "created" by process models where optional sensitive states or attributes have been set up ("electronic signatures"). Entering a request into the system by actioning it out of the held state is considered the "authorization point" for these process models. Also applies to held requests that are updated at the held state (using the command UC) before being actioned.

CCO – Create a New Contact

```
/CO_NAME=<contact_name>
[/TITLE=<job_title_of_contact>]
[/EMAIL=<e-mail_address_of_contact>]
[/ADDRESS=<postal_address_of_contact>]
[/CONTACT_TYPE=<additional_information>]
[/CONTACT_ID=<identity_of_contact>]
```

Example

```
CCO /CO_NAME="SERVER MANAGER" -
   /TITLE="SENIOR SERVER MANAGER"-
   /EMAIL=SERVER.MANAGER@@COMPANY.COM -
   /ADDRESS="ABBEY VIEW, ST ALBANS" -
   /CONTACT_TYPE=M
   /CONTACT_ID="John Brown"

CCO /CO_NAME="MAINFRAME MANAGER" -
   /TITLE="MAINFRAME ARCHITECT" -
   /EMAIL=MAINFRAME.MANAGER@COMPANY.COM -
   /ADDRESS="ABBEY VIEW, ST ALBANS" -
   /CONTACT_TYPE=M
   /CONTACT_ID="Janet Green"

CCO /CO_NAME="DB ADMIN" -
   /TITLE="DBA" -
   /EMAIL=DBA.MANAGER@COMPANY.COM -
   /ADDRESS="ABBEY VIEW, ST ALBANS" -
   /CONTACT_TYPE=M
   /CONTACT_ID="Fred Bowyer"
```

This command enables you to add a new contact to an installation. See the *System Administration Guide* for details.

CCS – Create Credential Set

```
CCS <credential-spec>
/USER =<userid>
/PASSWORD=<password>
```

This command enables you to create a new credential set. See the *System Administration Guide* for details.

CCST – Create a New Codeset

```
/CDST_NUMBER=<codeset_number>
/DESCRIPTION=<codeset-description>
```

Example

```
CCST /CDST_NUMBER="2000" -
   /DESCRIPTION="Description - EBCDIC Ireland (Euro)"
```

This command enables you to add a new codeset to an installation. See the *System Administration Guide* for details.

CCU – Create Customer

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<name>
/LOCATION=<location>
/PROJECT=<project-spec>
[/COMMENT=<comment>]
[/CONTACT=<contact-details]
```

Example

```
CCU "Brown Finances"
   /LOCATION="Manchester"
   /PROJECT="PAYROLL"
   /CONTACT="Mrs E Green"
```

Parameters and qualifiers

- `<name>`

  Specifies a name for the customer.

- `/LOCATION=<location>`

  Specifies the customer's physical location.

- `/PROJECT=<project-spec>`

  Specifies the project name.

- `/COMMENT=<comment>`

  for optionally adding more about the customer.

- `/CONTACT=<contact-details>`

  for optionally adding customer contact details.

Description

The Dimensions product allows you to maintain a list of customers and a record of which Dimensions releases have been sent to each customer.

The CCU command enables you to add customers to the list when you are ready to forward a release to that customer.

Constraints

The combination of customer name, location, and project-spec must be unique in the Dimensions database.

CFS – Create a File System

```
/FS_NAME=<file_system_name>
[/DESCRIPTION=<description>]
```

Example `CFS /FS_NAME=NTFS /DESCRIPTION="NT FILE SYSTEM"`

This command enables you to define specific file systems definitions for each registered installation operating system. See the *System Administration Guide* for details.

CGRP – Create Group

```
<group-name>
[/DESCRIPTION=<description>]
```

Example

```
CGRP "Contractors" /DESCRIPTION="Contract employees"
```

Parameters and qualifiers

- `<group-name>`

  The name of the group.

- `<description>`

  An optional description for the new group.

Description

The CGRP command creates a group.

Constraints

Only users with the appropriate management privileges can run this command.

CHMOD – Change File Permissions

```
<permissions>
[<filename>]
```

Example

```
CHMOD 777 src/build/hello.c
```

Parameters and qualifiers

- `<permissions>`

  This is the UNIX file permissions to be applied to the file.

- `<file-name>`

  Specifies the name of the file whose permissions are to be changed in the repository.

  The 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.

Description

This command allows you to change the permissions of an item file in the Dimensions repository. This changes the permissions the file will have when it is fetched to the work area. These permissions will be recreated when fetching files from the desktop client, and approximately recreated when using the Web client and Eclipse plug-in.

Constraints

Only users with the appropriate management privileges can run this command.

CI – Create Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for creating items in a stream.

```
<item-spec>
/PART=<part-spec>
[/ROOT_PROJECT=<project-spec>]
/FILENAME=<file-name>
[/WS_FILENAME=<ws_filename>]
[/COMMENT=<comment text>]
[/FORMAT=<format>]
[/USER_FILENAME=<user-filename>]
[/[NO]KEEP]
[/DESCRIPTION=<description>]
[/EXTRA_VARIANTS=(<var1>,<var2>,...)]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/STATUS=<status>]
[/WORKSET=<project-spec>]
[/CODEPAGE=<code-page>| DEFAULT]
[/CONTENT_ENCODING=<file-encoding>]
[/NOMETADATA]
```

Example

```
CI PROD:"QUERY RELEASE"-SRC
   /FILENAME=qr.c
   /USER_FILE=qr.c
   /WS_FILENAME="src/qr.c"
   /PART=PROD:"RELEASE MANAGEMENT".AAAA
   /EXTRA=AAAB
   /COMMENT="created for CRB 66"
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	identifies the new item within the product.
`<variant>`	if omitted, the default specified when the product was defined is used.
`<revision>`	defaults to 1, if omitted

NOTE If auto-id generation is enabled for this item type (see the related document *Process Configuration Guide*), an item identifier is generated automatically when an item is created. In this case Dimensions will generate a unique identifier. Note that the identifier will be automatically generated even if the user enters a value.

- /PART=<part-spec>

  Identifies the design part that will own the item.

  It comprises just the design part ID, or the following specification:
  <product-id>:<part-id>.<variant>;<pcs>

<product-id>	must be the same as that for <item-spec>
<variant>	may be omitted if only one exists.
<pcs>	is ignored. The item is always OWNED by the current PCS.

  If several design parts with the same name and different variants exist in the product, the full design part specification should be provided.

- /ROOT_PROJECT=<project-spec>

  Comprises:

  <product-id>:<project-id>

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- /FILENAME=<file-name>

  Specifies the name of the file which is to contain the **item** in the item library. It should (but need not necessarily) be of the form: <name>.<type> (see <format> below for the use of <type>). <file-name> **may** include subdirectory name(s) but must **not** specify an absolute path; thus:

  - **UNIX** ⇒ <file-name> **must not** begin with **/** (forward slash).
  - **Windows** ⇒ <file-name> **must not** begin with **** (backslash) or <drive:>.

  If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

- /WS_FILENAME=<ws_filename>

  Identifies the relative path (directory plus file name) of the file to be used when the item created here is subsequently checked out or gotten as an item from the current project.

<ws_filename>	may include sub-directory name(s), but must not specify an absolute path, as above.

- /COMMENT=<comment text>

  Comment text to explain the reason for the creation of this item revision. The comment text can be up to 1978 characters long, and can be made available within the item header.

- /FORMAT=<format>

  A user with the role of TOOL-MANAGER may have defined (DDF) and assigned (ADF) a list of valid data formats to particular item types. Uses for such a list include:

- Validation on item creation, specifying file types for the Dimensions desktop client application and specifying MIME types for the Dimensions web client.
- Dimensions Make. The format field allows items of the same item type to be distinguished on the basis of language (for program sources) or of execution platform (for executable program files) and so on. In this way different build processes can be defined for items of the same type but different format. For example, an item of type SRC and format C will be compiled using a C compiler, whereas an item of the same type but format PAS will be complied using a Pascal compiler.

If a list of valid file formats have been assigned to an item type, then the use of one of those formats is compulsory when creating items of that type; whereas, if a list of format types has not been assigned, then any format can be used at the time of item creation, even one not defined by a user with the role of TOOL-MANAGER.

If <type> is specified in <file-name> then <format> will default to that and does not need to be explicitly specified (but if <file-name> does **not** include <type>, then <format> **cannot** be omitted).

- /USER_FILENAME=<user-filename>

  Specifies the name of the file which holds the item in the user-area.

  If omitted, then either a skeletal document (from a format-template) or a null file is created.

- /KEEP

  Specifies that the <user-filename> which is normally deleted once the item has been placed under Dimensions control, is to be left intact.

- /DESCRIPTION=<description>

  This is optional text that will be displayed by Dimensions applications and reports.

  Dimensions supplies default text if this is omitted.

- /EXTRA_VARIANTS=<varN>

  Identifies another variant of the OWNER design part which also USES the item.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

  <attrN> is 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 value to be given to this attribute.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  <requestN> identifies a request to which the new item is to be related In Response To.

- /STATUS=<status>

  Specifies the state of the created item.

NOTES

If you specify a state that has been set to 'sensitive' in your process model, the sensitive state is ignored and the item is created at the initial state of the lifecycle. For example, assume that you have three states, *A, B*, and *C*, and *B* has been set to sensitive. If you try to create a new item at status B, it is created instead at the initial state, *A*.

The only equivalent to this parameter in interactive mode is CI followed by AI. The status, if specified, must be one which would be valid if AI had been used separately. If omitted, the initial state (in the lifecycle defined for <item-type>) is assigned.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  and is optional. If specified, the new item will be placed in that project. If unspecified, the new item will be placed in the user's current project.

- /CODEPAGE=<code-page>|DEFAULT

  Specifies the *code page* to be associated with the item. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
  /CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

  For further details concerning code pages and logical nodes see the *Network User's Guide*.

  You are advised to let the parameter default to the code page for the item type or the platform.

  The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.

- /CONTENT_ENCODING=<file-encoding>

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

Note on Areas

The CI command results in a new revision being created in the project. If DEVELOPMENT deployment areas are in use, CI automatically updates the corresponding areas.

Constraints

Generally, this command can be run by users who have one of the roles required to action the item from the initial lifecycle state to a new state. However, if a user with the role of PRODUCT-MANAGER has assigned $ORIGINATOR role to the first transition in the lifecycle for this item type, any Dimensions user can create an item of this type provided they have a role (any will suffice) on the design part owning the item.

A user with the role PRODUCT-MANAGER can also create items where no appropriate roles have yet ben allocated. This is to facilitate quicker migration of files.

CINS – Register a Database Instance Entry

```
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
[/DB_TRANSPORT=<db_spefic_transport>]
[/DB_NAME=<db_name>]
[/DB_TWO_TASK=<Oracle_specific_remote_connection_string>]
[/DB_HOME_DIR=<home_directory_of_db_instance>]
[/DB_ACTIVE=<reseved_(actively_used_y/n)>]
[/CO_NAME=<contact_name>]
```

Example

```
CINS /NN_NAME=MACHINE.COMPANY.COM
   /DB_SERVICE=PCMSUDB
   /DB_HOME_DIR=.
   /DB_NAME=PAYROLL
```

This command enables you to register database instances in an installation's network administration tables. See the *System Administration Guide* for details.

CIP - Create Installation Package

```
/DIRECTORY=<directory path>
/ATTRIBUTE=(AU_VERSION="<value>", AU_PRODUCT="<description>",
    AU_TYPE=DmCmClient | DmCmAgent, AU_PLATFORM=win32 | win64 |
    redhat-amd64 | linux-ia64 | linux-x86 | linux-s390x)
/PRODUCT=<product-id>
[/LOG=<filename>]
```

Description

Creates a baseline containing files used to upgrade a Dimensions CM client or agent.

Examples

```
CIP /directory="/home/lthomas/Documents/WORK/AU/cm_w64_client" -
/ATTR=(AU_PLATFORM=WIN64, AU_VERSIon=14.4, AU_TYPE=DmCmClient,
    AU_PRODUCT="Dimensions CM Client for Windows 64-bit") -
/product=cau /log=cip.log

CIP /directory="/home/lthomas/Documents/WORK/AU/cm_w32_client" -
/ATTR=(AU_PLATFORM=WIN32, AU_VERSION=14.4, AU_TYPE=DmCmClient,
    AU_PRODUCT="Dimensions CM Client for Windows 32-bit") -
/product=cau /log=cip.log
```

Parameters and Qualifiers

- `/DIRECTORY=<directory path>`

  Specifies the folder whose contents will be used to create the installation package.

- `/ATTRIBUTE`

  Specifies the type of installation package, the CM version, and the platform. The attributes are mandatory and are single value string fields with a maximum length of 240 characters.

  NOTE: Attribute definitions are created automatically if they do not exist.

  - `AU_VERSION="<value>"`

    Specifies the version of Dimensions CM in the installation package, for example: `"14.4"`

  - `AU_PRODUCT="<description>"`

    Describes the installation package, for example: `"Dimensions CM Client for Windows 64-bit"`

  - `AU_TYPE=DmCmClient | DmCmAgent`

    Specifies the CM installation type (client or agent).

- AU_PLATFORM=<platform>

  where <platform> specifies the target platform of the installation package:

win32	Windows 32-bit
win64	Windows 64-bit
redhat-amd64	Linux 64-bit
linux-ia64	Linux Itanium
linux-s390x	Linux zSeries

  To determine the highest available update for a system, combine AU_VERSION with AU_TYPE and AU_PLATFORM.

- /PRODUCT=<product-id>

  Specifies the product that will own the installation package.

- /LOG=<filename>

  Specifies the log path and filename.

Prerequisites

The baseline type used to create the installation package must exist and have a valid lifecycle associated with it. By default the CIP command uses the type "BASELINE". Add or modify the following dm.cfg variable to customize the baseline type name that is used:

```
DM_CIP_TYPE_NAME <baseline type name>
```

CIU – Cancel Item Update

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for items that belong to a stream.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/WORKSET=<project-spec>]
[/[NO]KEEP]
[/NOMETADATA]
```

Example

```
CIU PROD:"QUERY RELEASE".AAAA-SRC;1
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if you have checked out only one revision of this item.

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  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. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  A project is normally specified on item check out, so it would not normally need to be specified on cancel item update.

If not specified, the user's default project will be used.

- `/KEEP or /NOKEEP`

  /KEEP prevents CIU from deleting the extracted (checked out) file. This is the default. Specify /NOKEEP to delete the extracted file.

- `/NOMETADATA`

  This parameter disables creation and usage of metadata files in the local work area.

Constraints

This command can be run only by the user who checked out the item with the EI command, or by a user with the appropriate management privileges.

CLCA – Create Library Cache Area

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
[/DESCRIPTION=<area-description>]
/NETWORK_NODE=<node-name>
/DIRECTORY=<HLQ/directory>
[/USER=<user-name or credential-set-name> [/PASSWORD=<password>]]
[/OWNER=<user-name> or <group-name>]
[/STATUS=ONLINE or OFFLINE]
```

Example

```
CLCA <area-name> /NETWORK_NODE=<host-machine> /DIRECTORY=<area-
    directory>
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area. Area names must be unique within the base database.

- `/DESCRIPTION=<description>`

  Optional. Specifies a description for the new area.

- `/NETWORK_NODE=<node-name>`

  Specifies the machine hosting the area.

- `/DIRECTORY=<HLQ/directory>`

  Specifies the directory, or PDS (partitioned data set), where the area is located.

  If you specify a UNIX directory, for example, `/home/dmsys/lib_cache`, you must enclose the path in double-quotes (`"/home/dmsys/lib_cache"`) for dmcli to parse the `/DIRECTORY` qualifier correctly.

  HLQ is a high-level qualifier; for example, MERVK.WORK. It is a common prefix for all data sets in the area such as MERVK.WORK.C, MERVK.WORK.CBL, and so forth.

- `/USER=<user-name or credential-set-name> [/PASSWORD=<password>]`

  Login information for the operating system user account or credential set that will own files transferred into the area. For more information about credential sets see the *System Administration Guide*.

- `/OWNER=<user-name> or <group-name>`

  Optional. Specifies the user or group that is to become the owner of the new area. If /OWNER is not specified, the user who created the area is set as its owner.

- `/STATUS=ONLINE or OFFLINE`

  Specifies the status of the library cache area. If the area's status is ONLINE, the area may participate in file transfer operations. If the area's status is OFFLINE, the area is automatically excluded from any file transfer operations. If this qualifier is not specified, an area with status ONLINE is created.

Description

The CLCA command creates a library cache area definition.

A library cache area is a named location on a network node, which must have a dimensions listener running.

The purpose of a library cache area is to improve file fetch performance by caching file contents on a network node that is "close" to the user's computer. This avoids transferring the file repeatedly over a potentially slow connection between the item library node and the user's network.

Constraints

To create an area, you must have the Create Library Cache Areas privilege.

CLEAN – Clean Deployment Area

```
CLEAN "<areaName>"
[/COMMENT="<userComment>"]
[/WORKSET="<projectName>"]
```

Parameters and qualifiers

- `<areaName>`

  The name of the area to clean.

- `/COMMENT=<userComment>`

  A comment to describe the purpose of the clean action. This is stored in the deployment history.

- `/WORKSET=<projectName>`

  The project or stream to which the clean action is constrained.

Description

The CLEAN command removes all controlled content (including directories) from a deployment area. You should do this for all content in all projects that share an area. Non-controlled content will not be deleted.

CMB – Create Merged Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<new-baseline-spec>
/PART=<part-spec>
[/TYPE=<baseline-type>]
/BASELINE_LIST=(<baseline1>,<baseline2>,...)
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/USER_BASELINELIST]
```

Example

```
CMB PROD:"MERGED VER 2A FOR HP"
   /PART=PROD:"RELEASE MANAGEMENT".AAAB
   /BASE=("QUERY RELEASE MOD 2A","R M VERSION 2 FOR HP")
```

Parameters and qualifiers

- `<new-baseline-spec>` comprises:

  `<product-id>:<baseline-id>`

`<baseline-id>`	is the identity to be given to the merged baseline being created.

- `/PART=<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/TYPE=<baseline-type>`

  Specifies the type of the merged baseline being created.

  If omitted, `RELEASE` is used as a default type.

- `/BASELINE_LIST=(<baseline1>,...)`

  Identifies one or more existing release-baselines (usually at least two), whose items are to be merged into the new baseline. All these baselines must be for the same product as is specified for the merged baseline; so each `<baselineN>` can be specified either as `<baseline-spec>` or simply as `<baseline-id>` i.e. the syntax is:

  `[<product-id>:]<baseline-id>`

`<product-id>`	can be omitted (the colon (:) also being omitted), but if present must be the same as that which was specified in `<new-baseline-spec>`.
`<baseline-id>`	is the identity of one of the existing baselines to be used in the creation of the merged baseline.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

  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.

- /USER_BASELINELIST

  Specify a file containing a list of baseline specifications. Enter each specification on a new line.

Description

1 This command creates a new baseline, using <part-spec> as the top design part, to include exactly the same list of design parts (i.e. to have the same scope) as a baseline created by the CBL command would have; but initially this new baseline contains no items.

2 The baselines in /BASELINE_LIST are then considered in turn **in the order specified in that list**; and each of the items in each baseline is checked as follows and ignored:

- if it is not a relevant item in the new baseline
- if any revision of the same item has already been added to the new baseline.

If these checks are passed, each item revision is added to the new baseline.

This continues until all items in all the baselines in the list have been dealt with.

This processing means that, for any item in the merged baseline, the revision added will be the one found in the first baseline in the list which contains that item. Therefore, in order to obtain a merged baseline with satisfactory contents, it will generally be necessary to list the input baselines in increasing order of antiquity: the most recent baselines first and the oldest last.

Merged baselines can be useful in several different ways, and the following is just one possibility (illustrated in the command example given above). A complete system or subsystem is baselined, tested and released. Later a component of it is modified, and its design parts are baselined by themselves and tested. However, the Dimensions function **REL – Release** must always be executed from a single baseline. In order to re-release the same system with just that component modified, these two baselines are merged, with the later baseline first in the baseline-list. A fresh baseline of the whole system might have introduced other modifications done elsewhere in the meantime, which it is undesirable to include in the present re-release.

NOTE Dimensions collates the merged baseline completely automatically according to the specification above; it is entirely the Dimensions user's responsibility to specify baseline-lists that will create sensible and meaningful merged baselines. In any case, the contents will probably not be readily traceable to a consistent set of template rules, and as a reminder of this, it is wise to use some distinctive naming convention for the baseline-ids of merged baselines.

Constraints

Each of the input baselines, as specified in /BASELINE_LIST, must be either a release-baseline which was created using a template with **no** *ALL rules, or else an earlier merged baseline or a revised baseline. They must all be baselines for (any design structure within) the same product as is specified for the new baseline.

The new baseline-id must be unique within the product.

Generally, to create a merged baseline you must have one of the roles for the top design part in the new merged baseline that are required to action the merged baseline from its initial lifecycle state to a new state. However, if a user with the role of PRODUCT-MANAGER has assigned $ORIGINATOR role to the first transition in the lifecycle for this baseline type, any Dimensions user can create a merged baseline of this type provided they have a role (any will suffice) on the top design part on which the merged baseline is being created. There are no role requirements for the top-level design part of any input baselines.

CMD – Execute Dimensions Command File

```
<command-filename>
[/LOGFILE=<log_filename>]
```

Example

```
CMD items.setup /log=items_setup.log
```

Parameters and qualifiers

- `<command-filename>`

  Specifies the name of a file containing commands which are to be executed (see About the Command-Line Interface on page 12 for more information).

- `/LOGFILE=<log_filename>`

  Specifies a log file name into which to redirect the output from a command script. If no `/LOGFILE` parameter is specified, all output will be logged to the screen.

NOTE

`CMD` cannot be run from Dimensions for z/OS.

`CMD` is foreground replacement for the background `XCMD` command available in previous versions of Dimensions.

CMD scripts may NOT contain additional CMD commands. This is because sub-CMDs are not supported.

CMP – Compare Structures or Baselines

NOTE CMP cannot be run from Dimensions for z/OS.

```
<file-name>
[/PART=<part-spec> or /BASELINE=<baseline-spec>]
[<file-name> [/PART1=<part-spec> or /BASELINE1=<baseline-spec>]]
```

Example

```
CMP * /BASELINE=PROD:"R M VERSION 2 FOR HP" -prm3_01.export
   /PART1=PROD:"RELEASE MANAGEMENT".AAAB
```

Parameters and qualifiers

- `<file-name>`

  Specifies where the first or second exported structure is stored, when `<part-spec>` or `<baseline-spec>` is not specified.

  Otherwise, `<file-name>` specifies the name of the file where the exported structure is to be stored. If specified as an asterisk (*), then a temporary file is created which is deleted at the end of the operation.

  If no file name is specified, `compare.out` is created by default.

- `/PART=<part-spec>`

  Specifies the top design part of the current product structure which is to be included in the export file.

  It comprises: `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	must be specified, even if only one variant exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/BASELINE=<baseline-spec>`

  Specifies the baseline on which the structure to be included in the export file is to be based.

  It comprises: `<product-id>:<baseline-id>`

  If the second `<file-name>` is not specified, the specified structure is exported but no report is generated.

Constraints

This command can be run only by the user initiating the report who must have a valid role for the top design part in the structure to be reported.

CNC – Create a Network Node Connection

```
/SERVER_NAME=<server_node_name>
/CLIENT_NAME=<client_node_name>
/NWO_NAME=<network_object_name>
[/FS_NAME=<file_system_name>]
[/CDST_NUMBER=<code_set_number>]
[/DIRECT_FILE_COPY]
[/FILE_COMPRESSION]
```

This command enables you to add a new network node connection to an installation. For details, see the *System Administration Guide*.

CNDO – Create a Node Object

```
/NN_NAME=<network_node_name>
/NWO_NAME=<network_object_name>
```

This command enables you to add a new node object to an installation. See the *System Administration Guide* for details.

CNN – Create a Network Node

```
/NN_NAME=<network_node_name>
/OS_NAME=<operating-system-name>
/LOGICAL=<y|n>
[/PHYSICAL_NAME=<physical_node_name>]
[/CO_NAME=<contact-name>]
[/DESCRIPTION=<description>]
[/RSD_NAME=<resident_software_definition>]
```

Example

```
CNN /NN_NAME=MACHINE.COMPANY.COM
   /LOGICAL=N
   /OS_NAME=XP
   /DESCRIPTION="SOURCES HELD ON DFS DIMENSIONS XP SERVER"

CNN /NN_NAME=TEST_MACHINE.COMPANY.COM
   /LOGICAL=N
   /OS_NAME=XP

CNN /NN_NAME="MY_MAINFRAME"
   /PHYSICAL_NAME=MF390.COMPANY.COM
   /LOGICAL=Y
   /OS_NAME=MVS
   /DESCRIPTION="MAINFRAME SERVER SITUATED AT HEAD OFFICE"
```

This command enables you to create new physical or logical nodes. See the *System Administration Guide* for details.

CNSJ – Cancel Schedule Job Execution

```
<job-id>
```

Example:

```
CNSJ "MyJobName"
```

Parameters and qualifiers

- <job-id>

  Specifies the job-id.

Description

Enables you to cancel the execution of a scheduled job that is currently running.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

CNWO – Create a Network Object

```
/NWO_NAME=<network_object_name>
/PROTOCOL=<communication_protocol>
[/DESCRIPTION=<description>]
[/PROCESS=<network_object_process_name>]
```

Example

```
CNWO /NWO_NAME=PCMS_SDP
   /PROTOCOL=SDP
   /DESCRIPTION="NETWORK OBJECT USING STANDARD DIMENSIONS PROTOCOLS"
```

This command enables you to add a new network object to an installation. See the *System Administration Guide* for details.

COS – Create an Operating System

```
/OS_NAME=<operating_system_name>
/CASE=<operating_system_case_convention>
/LIB_PROTECTION="<type_of_library_protection>"
```

Example

```
COS /OS_NAME=XP
   /CASE=NN
   /LIB_PROTECTION="RWX,RX,R"
```

This command enables you to add a new operating system to an installation. See the *System Administration Guide* for details.

CP – Create Design Part

```
<part-spec>
/CATEGORY=<category-name>
/FATHER_PART=<parent-part-spec>
/DESCRIPTION=<description>
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
```

Example

```
CP PROD:"RELEASE MANAGEMENT"
   /FATHER_PART=PROD:PROD -
   /CAT=SUBSYSTEM
   /DESC="Release Support Sun Version"
```

Parameters and qualifiers

- `<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	if omitted, the default (specified when the product was defined) is used.
`<pcs>`	if omitted, the PCS for new variants (specified when the product was defined) is used.

- `/CATEGORY=<category-name>`

  Specifies the category of design part.

- `/FATHER_PART=<parent-part-spec>`[1] comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/DESCRIPTION=<description>`

  This is mandatory text to describe the function of the design part.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

  `<attr1>` is the Variable Name defined for one of the user-defined attributes for design parts, which has also been declared as usable for this `<product-id>` and `<category-name>`. `<valueN>` is the substitution value to be given to this attribute.

Constraints

Only users with the appropriate management privileges can run this command.

Design part names as specified in the <part-id> cannot include colon (:) or semicolon characters (;).

CPV – Create Design Part Variant

```
<part-spec>
/NEW_VARIANT=<new-variant>
[/FATHER_VARIANT=<parent-variant>]
[/DESCRIPTION=<description>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
```

Example

```
CPV PROD:"RELEASE MANAGEMENT".AAAA
   /NEW_VAR=IBM
   /DESC="Release Support - IBM Version"
```

Parameters and qualifiers

- `<part-spec>`

  Specifies an already existing design part. It comprises:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

  `<variant>` may be omitted if only one variant has existed up to now.

  `<pcs>` is ignored; the current PCS is always used.

- `/NEW_VARIANT=<new-variant>`

  Specifies the identity of the new variant, e.g. AAAB.

- `/FATHER_VARIANT=<parent-variant>`[1]

  Specifies the variant code of the parent design part to which the new variant is to be related.

  It may be omitted if the father design part has only one variant.

- `/DESCRIPTION=<description>`

  This is optional text describing the function of the variant.

  It defaults to the description associated with `<part-spec>`, if it is omitted.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

  `<attrN>` is the Variable Name defined for one of the user-defined attributes for design parts, which has also been declared as usable for this `<product-id>` and design part's category.

  `<valueN>` is the substitution value to be given to this attribute for the new variant.

Constraints

Only users with the appropriate management privileges can run this command.

CRB – Create Revised Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<new-baseline-spec>
/BASELINE1=<existing-baseline-spec>
[/TYPE=<baseline-type>]
[/UPDATE_CHANGE_DOC_IDS=(<ucd1>,<ucd2>,...)]
[/REMOVE_CHANGE_DOC_IDS=(<rcd1>,<rcd2>,...)]
[/CANCEL_TRAVERSE]
[/FILENAME=<report-filename>]
[/SCOPE=WORKSET
[/WORKSET=<project-spec>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/UPDATE_REQUIREMENT_IDS=(<requirement_spec1>{container_name+
    project_name+dbname+rm_browser_url),<requirement_spec2>
    {container_name+project_name+dbname+rm_browser_url},...)]
[/REMOVE_REQUIREMENT_IDS=(<requirement1>{container_name+project_name+
    dbname+rm_browser_url},<requirement2>{container_name+project_name+
    dbname+rm_browser_url},...)]
```

Examples

```
CRB PROD:"REVISED RM VER 2B FOR HP"
   /BASE=PROD:"MERGED R M VER 2A FOR HP"
   /UPDATE=(PROD_DR_25, PROD_DC_16)
   /REMOVE=PROD_DC_16
   /CANCEL_TRAVERSE

CRB "REPX:REV1" /BASELINE1="REPX:AMI_CHAN" /TYPE="BASELINE"
   /UPDATE_REQUIREMENT_IDS=
       ("Marketing_Requirements.MRKT_000020;79{Engineering
       Requirements+RMDEMO6+RM10+http://mars/rtmBrowser/}",
       "Marketing_Requirements.MRKT_000002;54{Engineering
       Requirements+RMDEMO6+RM10+http://mars/rtmBrowser/}")
   /REMOVE_REQUIREMENT_IDS=
       ("Marketing_Requirements.MRKT_000036;40{Engineering
       Requirements+RMDEMO7+RM10+http://mars/rtmBrowser/}",
       "Marketing_Requirements.MRKT_000028;17{Engineering
       Requirements+RMDEMO7+RM10+http://mars/rtmBrowser/}")
       /WORKSET="REPX:REPX" /CANCEL_TRAVERSE
```

Parameters and qualifiers

- `<new-baseline-spec>`

  which comprises:

  `<product-id>:<baseline-id>`

`<baseline-id>`	Specifies the identity of the new revised baseline to be created.

- `/BASELINE1=<existing-baseline-spec>`

  which comprises:

  `<product-id>:<baseline-id>`

`<product-id>`	Must be the same as `<new-baseline-spec>`.
`<baseline-id>`	Specifies the identity of the release baseline on which the revised baseline will be created.

- `[/TYPE=<baseline-type>]`

  Specifies the type of the revised baseline to be created. If omitted the type is the same as `<existing-baseline-spec>`.

- `/UPDATE_CHANGE_DOC_IDS=(<ucd1>,...)`

  Identifies one or more request trees in which there are item revisions with an In Response To relationship, or items that have been used to track project structure (refactoring) changes. Each revision with an In Response To relationship replaces the revision of the same item that was previously in the baseline. If there was no item in the baseline, the In Response To revision is added if it falls within the scope of the baseline). If a request relates to refactoring changes those changes are processed as described in "Project Structure Changes" on page 131.

  When multiple revisions of the same item have an In Response To relationship to one or more requests, the latest revision is selected. If two or more item revisions are on different branches, the CRB command issues a warning and continues processing without changing the revision of that item in the baseline.

NOTE For requests that result in an item revision being added to a revised baseline, an In Response To relationship is automatically created between that baseline and the appropriate request. These relationships are only created if that request has been used to revise the baseline. For example, if two requests with the same relationships are used to revise the baseline, then only the request that was used by CRB to generate the new baseline is related.

- `/REMOVE_CHANGE_DOC_IDS=(<rcd1>,...)`

  Identifies one or more request trees in which there are item revisions with an Affected relationship, or have been used to track project structure (refactoring) changes. Each Affected revision is deleted from the revised baseline.

IMPORTANT: The qualifiers /UPDATE_CHANGE_DOC_IDS and /REMOVE_CHANGE_DOC_IDS are optional however you must specify at least one.

NOTE For requests that result in an item revision being removed from a revised baseline, an Affected relationship is automatically created between that baseline and the appropriate request. These relationships are only created if that request has been used to revise the baseline. For example, if two requests with the same relationships are used to revise the baseline, then only the request that was used by CRB to generate the new baseline is related.

- /CANCEL_TRAVERSE

  Specifies that when /UPDATE_CHANGE_DOC_IDS and /REMOVE_CHANGE_DOC_IDS are processed traversal of tree structures is not performed. Only the requests in the lists are inspected for item revisions that have been related, respectively, as In Response To and Affected.

- /FILENAME=<report-filename>

  Specifies the output file name for a report that lists the impact of the changes to the baseline by each request (does not run the CRB command).

- /WORKSET=<project-spec>

  which comprises:

  <product-id>:<project-id>

  Specifies the project or stream to be used. When an item is to be added to the baseline, the project file name is taken from this project if there is no other revision in the project or stream.

  Default: the user's current project or stream.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

  Sets values for one of more attributes of the new baseline. <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>. <valueN> is the substitution value to be given to this attribute.

- /SCOPE=WORKSET

  Specifies that any items from design parts that have been created after the original baseline was created are included in the revised baseline. If omitted any items from new design parts are excluded from the baseline.

- `/UPDATE_REQUIREMENT_IDS=(<requirement_spec1>{container_name+project_name+dbname+rm_browser_url},<requirement_spec2>{container_name+project_name+dbname+rm_browser_url},...)`

  Specifies a comma separate list of Dimensions RM requirements to be updated. Each requirement is comprised of:

  `<requirement_spec>{container_name+project_name+dbname+rm_browser_url}`

  where:

<requirement_spec>	Comprises `<class_name>.<puid>;objId`. For example, `Marketing_Requirements.MRKT_000020;79`.
container_name	The name of the originating Dimensions RM container (baseline, collection, document, or snapshot) for the requirement. Multiple "versions" of a requirement cannot be related to a single Dimensions CM request. For example, `Engineering Requirements`.
`project_name`	Specifies the Dimensions RM project name, for example, `RMDEMO6`.
`dbname`	Specifies the Dimensions RM database name, for example, `RM10`.
`rm_browser_url`	Specifies the RM Browser URL, for example, `http://mars/rtmBrowser/`.

  Example of a requirement_id:

  ```
  "Marketing_Requirements.MRKT_000020;79{Engineering
      Requirements+RMDEMO6+RM10+http://mars/rtmBrowser/}"
  ```

- `/REMOVE_REQUIREMENT_IDS=(<requirement_spec1>{container_name+project_name+dbname+rm_browser_url},<requirement_spec2>{container_name+project_name+dbname+rm_browser_url},...)`

  Specifies a comma separate list of Dimensions RM requirements to be removed from the baseline. For details see `/UPDATE_REQUIREMENT_IDS` above.

IMPORTANT! You can only specify Dimensions RM requirements if you have:

- Installed the Dimensions RM integration.
- Associated Dimensions CM projects and streams with Dimensions RM collections.
- Associated Dimensions CM products with Dimensions RM projects.

See the *Dimensions CM-Dimensions RM ALM Integration Guide* and the Dimensions RM documentation for details.

Description

The CRB command creates a new baseline that is an exact copy of an existing baseline. The new baseline has the same top design part and list of included design parts as the baseline on which it is based.

NOTE A new baseline created by the CBL command, with the same top design part, may have a different scope because the breakdown and/or usage relationships in the design structure may be different.

The command works as follows:

1. The `/UPDATE_CHANGE_DOC_IDS` list is processed:

   a. A candidate list of requests is constructed from the specified list. Each specified request is considered as the head of a family tree of requests with relationships to it in the Dependent class.

      NOTE The number of requests is limited by the maximum command-line-list length of 16383 characters. The number of requests this translates into depends on the character lengths of the baseline and product name specifications used. If these specification names correspond to the maximum character lengths allowed by Dimensions CM, the request limit is 496. If they are shorter, the request limit is greater than 496.

   b. The specified request and its Dependent-related children are added to the candidate list, followed by any Dependent-related children (i.e. grandchildren), and so on until all descendants have been included. Some requests may already be in the candidate list if they belong to more than one family tree (they can be Dependent-related to more than one parent). The candidate list therefore contains the whole population of requests in all the tree structures specified by the `/UPDATE_CHANGE_DOC_IDS` list. If `/CANCEL_TRAVERSE` is specified the candidate list is just the specified list.

2. All the item revisions with an In Response To (R) relationship to any of the requests in the candidate list are added to the baseline, provided that the items are in the baseline's scope and the revisions are not checked out or suspended. During this process any revisions of those same items that are already in the baseline are removed. The In Response To revisions can be a combination of:

   - New additions to the baseline (where the item was not already included).
   - Substitutions for item revisions that are already in the baseline.

   Structural changes that are tracked by any request in the candidate list are processed as documented in "How the Create Revised Baseline Operation Interprets Structure Changes" on page 132.

3. The `/REMOVE_CHANGE_DOC_IDS` list is processed. A new candidate list of requests is constructed from this specified list as described above.

4. All item revisions with an Affected(A) relationship to any of the requests in this candidate list, and which are still included in the baseline, are deleted from it.

   Structural changes that are tracked by any request in the candidate list that reference the removal of items or project folders cause the removal of those items and folders. See "How the Create Revised Baseline Operation Interprets Structure Changes" on page 132.

   If any, or all, of these revisions are not found in the baseline they are ignored and no message is issued, see Error Conditions below.

The main purpose of 'remove' processing is to clear items from a baseline that are redundant, that is, items for which no In Response To relationship replacement was required. The 'remove' qualifier is typically required less frequently because the normal process of removing superseded revisions is done by the 'update' processing. When /REMOVE_CHANGE_DOC_IDS is used the requests listed may be the same as some of the requests in the /UPDATE_CHANGE_DOC_IDS list, but this is not a constraint.

NOTE A revised baseline may be less self-consistent than one created using a baseline-template. For example, the original baseline may include item revisions by using a *BUILT template rule. Also, the sources from which that revision was built may have been displaced from the revised baseline, but the original built revisions remain in the baseline unless they have been specifically displaced by the 'update' and 'remove' processing.

Use distinctive naming convention for the baseline-ids of revised baselines.

Project Structure Changes

When you create a revised baseline it include change information for all the project structure or refactoring changes related to the specified requests. When you perform commands that involve refactoring, such as AIWS and MWSD, and specify a request in the /CHANGE_DOC_IDS qualifier, those changes are recorded against that request. Specifying those requests in the Update list causes the changes to be applied to the revised baseline.

Project structure change control is normally enabled by default. You can enable it by setting a parameter, DM_PATH_CONTROL, in the DM.CFG file. See the *System Administration Guide* for more details.

Recorded Project Structure Changes

When project structure change control is enabled, the following changes to project structure are recorded against change requests:

Change	Description
Item addition and removal	An item revision is added or removed from a project. The change is recorded in one of the following: ■ The change request that lead to the structure change. ■ The default change request.
Item rename	An item is renamed or moved in the project structure.
Directory creation, deletion, and rename	A directory is created, deleted, or renamed in the project structure.

Project Structure Change Requests

When you create a revised baseline it includes all the structural changes tracked by the specified requests. The structural changes are only applied if they relate to the project whose context matches that in which the baseline is being revised. Structural changes outside of this context are ignored.

How the Create Revised Baseline Operation Interprets Structure Changes

When you create a revised baseline the operation interprets different types of structure changes as follows:

- Item additions related to update requests are interpreted as new candidate items to be added to the baseline.
- Item and directory renames related to update requests are interpreted as candidate changes to the baseline file and folder structure.
- Item removals related to update or remove requests are interpreted as candidates for removal from the new baseline.
- Directory removals related to update or remove requests are interpreted as candidate changes to the baseline structure.

For more information on revised baselines see the *Dimensions CM User's Guide*.

Notes on Structure Changes when Creating Revised Baselines

When you create a revised baseline:

- The operation ignores directory or item removals if the items or directories were not in the original baseline or have not subsequently been added in the context of a structure change.
- All structure changes are performed in their original sequence to ensure integrity is maintained when item and directory renames are interleaved.
- If the operation encounters a rename or removal change to a directory path, it avoids renaming or removing the wrong directory by ensuring there have been no directory additions or renames to the same path since the baseline was created. For example assume that:
  - The original baseline included the directory "`/source`".
  - The original "`/source`" directory has been deleted.
  - A new directory named "`/source`" has been added, with different content.
  - This new `/source` directory was renamed to "`/files`".

  The Create Revised Baseline operation detects that the original "`/source`" directory is not the same as the new `/source` directory and fails with an error.

Error Conditions

If there is a conflict when a revision to be deleted (as a result of 'remove' processing) is the same as the revision that was added or substituted (as a result of 'update' processing), the conflict is reported as a warning and the revised baseline is created.

If after 'remove' and 'update' processing has completed there is no change (the revised baseline is identical to the existing baseline), an error is reported and Dimensions CM automatically deletes the new baseline.

Constraints

- The existing baseline must be a release baseline that was created using a template with no *ALL rules, or an earlier revised or merged baseline.
- The new baseline-id must be unique in the product.
- This command can be run by users who have a role on the top design part in the existing baseline (which is also the top part for the new baseline) that is required to action the merged baseline from its initial lifecycle state to a new state. However, if a user with the PRODUCT-MANAGER role has assigned the $ORIGINATOR role to the first transition in the lifecycle for this baseline type, any Dimensions user can create a revised baseline of this type provided they have any role on the top design part on which the revised baseline is being created.

CRSD – Create a Resident Software Definition

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
/RSD_NAME=<name_RSD>
/RSD_VERSION=<version_number_of_RSD>
[/RSD_DATA=<RSD_data_details>]
```

This command enables you to register an installation Resident Software Definition (RSD) to be associated with the build environment in which Dimensions Make operates. See the *System Administration Guide* for details.

CS – Create a Stream

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
[<product name>:]<stream name>
/DESCRIPTION=<description>
[/STREAM=<stream-spec> or /BASELINE=<baseline-spec> {/BA}]
[/ATTRIBUTES=(<attr1>,attr2,...)] {/AT}
[/DEFAULT_BRANCH=<branch-id>]
[/[NO]CM_RULES]
[/DEFAULT_CM_RULES]
[/[NO]PATH_CONTROL]
[/PRODUCT=<product-id>
[/[NO]COPY_CONFIGS]
[/[NO]FORCE]
[/[NO]KEEP_STAGE]
```

Example

```
CS
QLARIUS:JAVA_BRANCH_TEST
/DESCRIPTION="Stream for testing"
/BASELINE="beta_v1"
```

Parameters and qualifiers

- `[<product name>:]<stream name>`

  where:

  `<product name>`

  (Optional) Specifies the CM product that will own the new stream.

  `<stream name>`

  Specifies the name of the new stream.

- `/DESCRIPTION=<description>`

  Optionally specifies a description for the stream.

- `/STREAM=<stream-id>`

  Optionally specifies the stream on which to base the new stream.

- `/BASELINE=<baseline-id>`

  Optionally specifies the Dimensions baseline on which to base the new stream.

- `/ATTRIBUTES=(<attr1>,attr2,...)`

  Specifies the user-defined attribute values for this stream.

- `/DEFAULT_BRANCH=<branch-id>`

  Specifies the `<branch-id>` to be 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*.

  Note that a new version branch is created in the base database.

- /[NO]CM_RULES

  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_CM_RULES

  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. For details of CM rules see *Process Configuration Guide* section *About Change Management Rules*.

- /[NO]PATH_CONTROL

  Specifies whether a request is required to perform refactoring operations in this stream.

- /PRODUCT=<product-id>

  Optionally specifies the product that will own the stream.

  If omitted, it defaults to the product that owns the stream or baseline on which the stream is based. This is not required when you specify a stream.

- /[NO]COPY_CONFIGS

  Specifies whether build configurations will be copied from the project referenced by the /WORKSET parameter to the new project.

  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.

- /[NO]FORCE

  When copying build information from the project referenced by the /WORKSET parameter to the new project, this option specifies whether the DWS will abort when the first error is encountered, or whether the process will continue to produce as many diagnostic messages as possible. /NOFORCE 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 /NOFORCE.

- /KEEP_STAGE

  When creating a stream based on an existing stream, specify this optional qualifier to control the stages of the items in the new stream.

  - Use /NOKEEP_STAGE to reset the stages of all the items in the new stream to the initial stage.
  - Use /KEEP_STAGE to keep the stages of the items from the source stream.

  This qualifier can only be used when the new stream uses the manual deployment model.

  Default (when the qualifier is not specified): /KEEP_STAGE

Description

The CS command is used to create a new stream in a Dimensions product. The stream can be empty, based on an existing stream, or based on an existing baseline. All the item revisions in a parent stream are included in a new child stream based on this parent.

When CS populates the new stream from an existing stream, and the /COPY_CONFIGS qualifier is specified, build configurations and/or relationships will also be copied to the new stream. See the Dimensions CM Build Tools User's Guide for further details.

Constraints

You need to have the STREAM_CREATE privilege to perform this command.

Because of a restriction with the use of MVS data sets, streams cannot be used with MVS data areas, work areas, or deployment areas.

CSJ – Create Schedule Job

```
<job-id>
/START_TIME
[/REPEAT]
[/JOB_STATUS]
[/JOB_DESC]
```

Example

```
CSJ "MyJobName" /START_TIME="31-12-2008 23:59:59" /REPEAT="30 MINUTES" /
    JOB_DESC="Schedule job description"
```

Parameters and qualifiers

- `<job-id>`

  Specifies the job name.

- `/START_TIME`

  Specifies the job starting time in the format 'DD-MM-YYYY HH24:MI:SS'.

- `/REPEAT`

  Specifies an optional repeat time, can be one of:

  - 0, 10, 20, 30, 40, 50, 60, MINUTES

    For example, /REPEAT="40 MINUTES"

  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, HOURS

    For example, /REPEAT="7 HOURS"

  - 0, 1, 2, 3, 4, 5, 6, 7, DAYS

    For example, /REPEAT="1 DAYS"

- `/JOB_STATUS`

  Specifies the initial job status: INACTIVE (default) or ACTIVE

- `/JOB_DESC`

  Describes the scheduled job.

Description

Enables you to setup a scheduled job.

CUSR – Register User

```
<user-id>
[/WORKSET=<project-spec>]
[/[NO]PASSWORD_SAVE]
[/ATTRIBUTES=(site=<site>,
    group_id=<group-id>,Dept=<dept>,
    full_name=<full-name>,phone=<phone>,
    <attribute-id>=<value>,email_addr=<email-addr>)]
```

Description

This command is the same as UREG.

For details, see the *System Administration Guide*.

CVS – Create Variant Structure

```
<part-spec>
/NEW_VARIANT=<new-variant>
[/FATHER_VARIANT=<parent-variant>]
[/DESCRIPTION=<description>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
```

Example

```
CVS PROD:"RELEASE MANAGEMENT".AAAA
   /NEW_VAR=IBM
   /DESC="Release Support - IBM Version"
```

Parameters and qualifiers

- `<part-spec>`

  Specifies an already existing design part. It comprises:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

  `<variant>` may be omitted if only one variant has existed up to now.

  `<pcs>` is ignored; the current PCS is always used

- `/NEW_VARIANT=<new-variant>`

  Specifies the identity of the new variant, e.g. AAAB.

- `/FATHER_VARIANT=<parent-variant>`[1]

  Specifies the variant code of the parent design part to which the new variant is to be related.

  It may be omitted if the parent design part has only one variant.

- `/DESCRIPTION=<description>`

  This is optional text describing the function of the variant.

  It defaults to the description associated with `<part-spec>`, if it is omitted.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

NOTE To use the `/ATTRIBUTES` qualifier with the CVS command, the attributes specified must be valid for every design part in the structure to be copied. If the attributes do not meet this requirement, then either CPV commands must be entered individually or UP commands must be issued after issuing a CVS command without the `/ATTRIBUTES` qualifier.

`<attrN>` is the Variable Name defined for one of the user-defined attributes for design parts, which has also been declared as usable for this `<product-id>` design part's category.

`<valueN>` is the substitution value to be given to this attribute for the new variant.

NOTE CVS is similar to CPV except that it creates a whole new part variant structure (by including every variant in the structure from the part you specify) i.e. it is the equivalent of a whole series of CPVs.

Roles may then be assigned to specific design part variants.

Constraints

Only users with the appropriate management privileges can run this command.

This command fails if the structure already contains a variant.

CWSD – Create Project Directory

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<directory-path>
[/WORKSET=<project-spec>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
```

Example

```
CWSD src
```

Parameters and qualifiers

- `<directory>`

  The CWSD command creates a project directory <*directory*> in the database project structure relative to the working location for subsequent use in project operations

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project/stream to be used for this command: failing this, the user's current project/stream will be taken.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which this structural change to the project is to be related `In Response To`.

  Specify this optional qualifier if you want this structural change to the project to be recorded against the specified request(s). If path control has been enabled, this qualifier is mandatory. If path control is not enabled, then the request(s) will be ignored.

Constraints

Normally, only users with the appropriate management privileges can run this command.

This constraint can, however, be relaxed using the Set Project Permissions (SWSP) command, as described on page 451.

DAR – Delete Archive

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<archive-id>
```

Example

```
DAR AA12AB
```

Parameters and qualifiers

- `<archive-id>`

  This is the identity of an archive which is to be deleted.

See the *System Administration Guide* for details.

DBC – Delete Build Configuration

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
[ <build-configuration-name> | * ][; [ * | <number> ] ]
[/[NO]EXECUTE]
[/FORCE]
[/WORKSET=<product>:<project>]
[/PRODUCT=<product>]
```

Parameters and qualifiers

- *

  Delete all build configurations in the scope of either the workset or the product.

- <build-configuration-name>

  Delete all build configurations in the specified scope that have this name.

- ;*

  Specifies the tip revision of the build configuration.

- ;<number>

  Specifies a specific revision number for the build configuration.

- /NOEXECUTE

  Used for testing.

- /FORCE

  The default is /NOFORCE. /FORCE deleteS a build configuration even if it is currently checked out.

- /WORKSET=<product>:<project>

  Specifies the scope of the build configuration search.

- /PRODUCT=<product>

  Specifies the scope of the build configuration search.

NOTE /WORKSET and /PRODUCT are mutually exclusive.

Description

Use the Delete Build Configuration command to delete old versions of build configurations from your system, or to remove all records associated with a specific build-configuration name for a project.

NOTE A build configuration creates hidden records that are deleted only when a DBC command is issued without any specified revision number.

If any errors arise during the processing of the DBC command, all actions it has taken are backed out.

DBDB – Unregister an Existing Base Database Entry

```
/BDB_NAME=<base_db_name>
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
```

Example

```
DBDB /BDB_NAME=SERENA-PCMS
   /DB_SERVICE=PCMSUDB
   /NN_NAME=MACHINE.COMPANY.COM
```

This command enables you to unregister base database entries in an installation's network administration tables. See the *System Administration Guide* for details.

DBL – Delete Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
```

Example

```
DBL PROD:"R M VERSION 2 FOR HP"
```

Parameters and qualifiers

- `<baseline-spec>` comprises:

  `<product-id>:<baseline-id>`

Constraints

This command can be run only by the user who created the baseline or by a user with the appropriate management privileges.

A baseline used as a child collection cannot be deleted.

A baseline that has been actioned can be deleted only by a user with the appropriate management privileges.

A baseline cannot be deleted if it has been used to make a release or archive.

DBPROJ – Create a Dimensions Build Project

NOTE This command is no longer available. Refer to the *Dimensions CM Build Tools User's Guide* for more information on how to define build projects.

```
Command no longer available.
```

DBT – Deliver Build Targets

```
DBT <build job>
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/DELETE]
```

Description

If a delivery fails during the execution of a build, or there are other problems, use this command to deliver an incomplete build. For more information about using Dimensions Build see the *Dimensions CM Build Tools User's Guide*.

Example

```
DBT R-1234
```

Parameters and Qualifiers

- `<build job>`

  Specifies the build job in progress that has an incomplete delivery.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...<requestn>)`

  Identifies requests to which the new items created in the stream are to be related In-Response-to (if required by change management rules). If a delivery failed during a build, specify the same requests that were used on the original BLD command.

- `/DELETE`

  Specifies an incomplete changeset to be deleted. The items are not removed and remain in the item library.

DCH – Delete Request

```
<request-id>
```

Example

```
DCH PROD_HELD_349
```

Parameters and qualifiers

- `<request-id>`

  This is the identity of the held request to be deleted.

Constraints

Requests which have been saved in the other types of request lists cannot be deleted.

This command can be run only on requests that are pending (in the Held list). They can be deleted by the user who created them or by a user with the appropriate management privileges.

DCO – Delete an Existing Contact

```
/CO_NAME=<contact_name>
```

Example

```
DCO /CO_NAME="SERVER MANAGER"
DCO /CO_NAME="MAINFRAME MANAGER"
DCO /CO_NAME="DB ADMIN"
```

This command enables you to a delete an existing contact from an installation. See the *System Administration Guide* for details.

DCS – Delete Credential Set

```
DCS <credential-spec>
```

This command enables you to delete a new credential set. See the *System Administration Guide* for details.

DCST – Delete an Existing Codeset

```
/CDST_NUMBER=<codeset_number>
```

Example `DCST /CDST_NUMBER="2000"`

This command enables you to a delete an existing codeset from an installation. See the *System Administration Guide* for details.

DDF – Define Data Formats

```
<format>
[/DESCRIPTION=<format-description>]
[/CLASS=<class-no>]
[/MIME_TYPE=<mime-type>]
[/COMPRESSION_LEVEL=<level>]
[/[NO]USE_DELTA_COMPRESSION]
```

Example

```
DDF TXT /DESCRIPTION="Plain text"
   /CLASS=1 /MIME_TYPE="TEXT/PLAIN"
```

Parameters and qualifiers

- `<format>`

  the format being defined.

- `/DESCRIPTION=<format-description>`

  descriptive name for the format.

- `/CLASS=<class-no>`

  Specifies the file types where:

  1=TEXT
  2=BINARY
  3=OpenVMS
  4=Macintosh
  5=NT

- `/MIME_TYPE=<mime-type>`

  Specifies the Multipurpose Internet Mail Extension (MIME) type. MIME types comprise seven broad categories, with each category also having subcategories defined by using a forward slash (/) separator. The broad categories are: Application, Audio, Image, Message, Multipart, Text and Video. An example of a subcategory is `APPLICATION/WORD`.

- `COMPRESSION_LEVEL=<level>`

  Specifies the compression level to be used when getting item revisions assigned this data format. Use a digit from 0 to 9, where 0 indicates no compression, 1 means fastest compression method (but less compression) and 9 indicates slowest compression method (but best compression). If this qualifier is omitted, text file formats will use fastest compression method (level 1) while all other file formats will use no compression.

- `No parameters or qualifiers`

  If DDF is executed without parameters or qualifiers, it prints a list of existing data formats together with their descriptions.

- `/[NO]USE_DELTA_COMPRESSION`

  Decreases the size of the transferred item by only sending sections that have been modified between revisions.

Description

This command enables a user to define data formats to be assigned to particular item types or request types:

- **For items**, a user with the role of TOOL-MANAGER can define a list of valid data formats for particular item types. These defined data formats can then, where appropriate, be subsequently assigned by a user with the role of TOOL-MANAGER to particular item types using the Assign Data Formats to Item Types (ADF) command (see page 42).
- **For requests**, a user with the role of TOOL-MANAGER or CHANGE-MANAGER can define a valid data format for a particular request type. This defined data format can then, where appropriate, be subsequently assigned by a user with the role of PRODUCT-MANAGER or CHANGE-MANAGER to a particular request type using the Assign Data Formats to Request Types (ACF) command (see page 41).

This function is also available from the Process Modeler, Data Formats and MIME Types option.

Uses for such a list of valid data formats include:

- Validation on item or request creation.
- Specifying file types: All items or requests that have formats not defined with "CLASS=1" (text) will be transferred to and from clients in binary mode. There is no need to assign the format to an item or request type.
- Specifying MIME types for the Dimensions web client. I-Net will use the MIME type associated with the item's or request's format to display the item's or request's content within the user's browser.

If a list of valid data formats is subsequently assigned to an item type, then the use of one of those formats is compulsory when creating items of that type; whereas, if a list of format types has not been assigned, then any format can be used at the time of item creation, even one not defined by a user with the role of TOOL-MANAGER.

If a valid data format is subsequently assigned to a request type, the choice of this format is compulsory when creating requests of that type. If no format has been assigned, binary is assumed.

Existing defined data formats can be removed by the use of the Remove Data Format Definitions (RMDF) command (see page 393).

NOTE For items only, the data format for existing items can also be updated using the Update Item Attributes (UIA) command (see page 479).

See also "SDF – Set Data Format Flags" command on page 423.

Constraints

Only users with the appropriate management privileges on the product can run this command.

DELIVER – Deliver to a Stream

NOTE When a project is specified, or the user's current project/stream is a project, this command behaves the same as the UPLOAD command (for details see page 505).

```
[<file-spec> or /DIRECTORY=<directory-spec>] or
    /USER_FILELIST=<filelist-file>]
[/[NO]RECURSIVE]
[/LOGFILE=<file-spec>
[/ATTRIBUTES=(<name>=<value>, ...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/CODEPAGE=<code-page> or DEFAULT]
[/COMMENT=<text>]
[/DESCRIPTION=<description>]
[/PART=<part-spec>]
[/CONTRIBUTER_STREAMS=(<stream-id>, ...)]
[/ALL]
[/USER_DIRECTORY=<directory-path>]
[/RELATIVE_LOCATION=<directory-spec>]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/UNLOCK_FILES]
[/CONTENT_ENCODING=<file-encoding>]
[/[NO]ADD]
[/[NO]UPDATE]
[/[NO]DELETE]
[/[NO]QUIET]
[/[NO]VERBOSE]
[/[NO]EXECUTE]
[/REMOVAL_SCOPE=<REVISION|STREAM>]
[/IMPORT]
[/NOIGNORE]
```

Description

The DELIVER command delivers a file or directory to a stream. If there are conflicts between the repository and the work area they are reported and the deliver fails. By default, only changes to controlled files are delivered. New or deleted files are only delivered to the repository if you specify the /ADD or /DELETE qualifiers. If a change is delivered for an item that has been locked by another user, the delivery fails. If a change is delivered for an item that has been locked by the user issuing the deliver operation, the delivery succeeds and the file is automatically unlocked.

IMPORTANT! As of Dimensions 14.1 you can only use the DELIVER command to deliver changes from a work area to the stream that was originally used to populate that work area.

Example

```
DELIVER /DIR=C:\temp\work\ /COMMENT="Fixed a bug"
   /ATTRIBUTES=(Complexity="High") /CHANGE_DOC_IDS=(PAYROLL_TDR_2)
```

This command delivers all files found in the directory `C:\temp\work` (and in any directories below it) to the user's current working stream. Any newly created revision are be related to PAYROLL_TDR_2, have the comment "Fixed a bug", and the revision's Complexity attribute is set to "High".

Parameters and Qualifiers

- `<file-spec>`

  Specifies the name of a file to be delivered. /DIRECTORY=<directory-spec>

  Specifies a directory path. The files in the directory are enumerated and each one that has been modified is delivered.

- `/USER_FILELIST=<filelist-file>`

  Specifies a file containing a list of file names to be delivered. Each file name must be on a separate line. File names may be specified as either relative or absolute paths. If the path is absolute it is interpreted as a full stream path. If not, Dimensions obtains the stream path by mapping the file name to the operation root directory, which is the current working location as specified by the last SCWS command. If this a mapping is not possible, the file name is ignored.

- `/[NO]RECURSIVE`

  If `/DIRECTORY` is specified and this qualifier is not present, all files that have been modified in all directories beneath the one specified are delivered. `/NORECURSIVE` specifies that only files at the specified directory level are delivered.

  Default: `/RECURSIVE`

- `/LOGFILE=<file-spec>`

  Generate a log file at the specified file location that contains the results of all the individual Dimensions CM operations executed with this command.

- `/ATTRIBUTES=(<name>=<value>, ...)`

  Specifies the user defined attributes to set on the newly created revisions. All attributes specified must be valid for the item types created.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  Specifies the requests for the items to be related to. The originally fetched versions will be related as "Affected", and the newly created versions will be "In Response To".

- `/CODEPAGE=<code-page>`

  Specifies the code page to be associated with the items.

- `/COMMENT=<text>`

  Specifies a comment to apply to all newly created item revisions.

- `/DESCRIPTION=<description>`

  Specifies a description to apply to all newly created items.

- /PART=<part-spec>

  Specifies the design part specification in this format:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

  The design part to which any new items will belong.

- /CONTRIBUTER_STREAMS=(<stream-id>, ...)

  If a working area contains files that originated from other streams that need to be added to the target stream, use this qualifier to specify which streams to add content from.

- [/ALL]

  Content originating from any stream is also included when delivering files.

- /USER_DIRECTORY=<directory-path>

  Specifies a directory other than the current working location. You can use the Dimensions node:: syntax. For example, the following command delivers to a stream from C:\temp regardless of the current working location:

  ```
  DELIVER /USER_DIRECTORY="C:\temp"
  ```

  The following command delivers to a stream from the /tmp directory on the host "hostname":

  ```
  DELIVER /USER_DIRECTORY="hostname::/tmp"
  ```

- /RELATIVE_LOCATION=<directory-spec>

  Specifies a project, stream, or baseline directory which is to be the "virtual" root directory for the duration of this command. If this parameter is given, the paths specified in <file-spec> or /DIRECTORY must be relative to the directory specified with /RELATIVE_LOCATION.

- /FILTER=<filter-name>

  The command only creates or updates files that satisfy the criteria specified in the area filter <filter-name>.

  An area filter is a regular expression following the same syntax as that used by the Dimensions GREP command.

- /USER_FILTER=<filter-file-spec>

  Specifies the name of a local file containing the definition of a file filter to be used when fetching or checking in files. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502.

  Only files matching the filter (and not excluded by the filter) are delivered when a user filter is specified.

- /UNLOCK_FILES

  Unlocks all items in the stream that were locked by the user issuing the command (even if the files being delivered do not include those that were locked).

- /CONTENT_ENCODING=<file-encoding>

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- /[NO]ADD

  Allows new content to be added to the repository. Default : NOADD.

NOTE If you specify /ADD, you may also need to specify /UPDATE if there are updates that need to be performed as well (specifically moves).

- /[NO]UPDATE

  Allows updating (and refactoring) of existing content in the repository.

  The default is UPDATE

- /[NO]DELETE

  Allows existing content to be deleted from the repository.

  The default is NODELETE

- /[NO]QUIET

  Only print critical messages.

- /[NO]EXECUTE

  Forces the transfer of files while generating a script file containing the equivalent Dimensions commands.

- /[NO]VERBOSE

  Print additional information about the update process.

- /REMOVAL_SCOPE=<REVISION|STREAM>

  Removes a specific revision or all revisions from the stream. The REVISION qualifier removes only the revision that was deleted from your work area. The STREAM qualifier removes all revisions of the file from the stream.

- /IMPORT

  Enables you to re-import files and folders back into a stream, for example:

  ```
  DELIVER /IMPORT /user_dir="/tmp/test/dd_1"
  ```

- /NOIGNORE

  You can use ignore rules to exclude specific files, folders, and file types from deliveries. To skip ignore rules and deliver all files, specify /NOIGNORE. For details about using ignore rules see the *User's Guide*.

  Default: /IGNORE

Constraints

You must have one of the following privileges to run this command:

- Streams: PROJECT_UPLOAD
- Items: ITEM_CREATE

DFS – Delete an Existing File System

```
/FS_NAME=<file_system_name>
```

This command enables you to remove specific file systems definitions for each registered installation operating system. See the *System Administration Guide* for details.

DGRP – Delete Group

```
<group-name>
```

Example `DGRP <group-name>`

where `<group-name>` is the name of the group to be deleted.

Description

The DGRP command deletes a group.

Constraints

Only users with the appropriate management privileges can run this command.

DI – Delete Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for items that belong to a stream.

```
<item-spec>
[/FILENAME=<file-name>]
[/WORKSET=<project-spec>]
```

Example

```
DI PROD:"QUERY RELEASE".AAAA-SRC;1
DI PROD:"QUERY RELEASE".AAAA-SRC;*
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be specified as ***** to delete all revisions of the product item that are in the project used for this command. If omitted, the latest revision is deleted (see note in About the Command-Line Interface on page 12)

- `/FILENAME=<file-name>`

  Specifies the name of the file containing the item in the item-library.

  It may be omitted if `<item-id>` is specified.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken. All item revisions to be affected by the command must be within the project.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Notes

- An item is not deleted if it is:
  - Used in an Item Process Definition (IPD).
  - Not at an initial lifecycle stage.
  - Used as an input to a built item.

- When an item is deleted, all areas are checked to verify that the item is not deployed. If the item is deployed to an area, the item deletion fails.
- An item can be referred to by an earlier version of an area, however, that area version cannot be used.

Constraints

This command can be run only by a user with the appropriate management privileges or, if the item revision is at the initial lifecycle state, by users if it is in their pending list.

Items cannot be deleted if they are included in a baseline.

DINS – Unregister an Existing Database Instance Entry

```
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
```

Example

```
DINS /NN_NAME=MACHINE.COMPANY.COM -
   /NN_NAME=MACHINE.COMPANY.COM -
   /DB_SERVICE=PCMSUDB
```

This command enables you to unregister database instance entries in an installation's network administration tables. See the *System Administration Guide* for details.

DIR – Define Item Relations

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<relationship-id>
/DESCRIPTION=<description>
/SRC_TYPE=<item-type-name>
/DST_TYPE=<item-type-name>
[/[NO]INHERIT_FROM]
[/[NO]INHERIT_TO]
```

Example

```
DIR "INCLUDES"–
   /DESCRIPTION="C source/header relationship"–
   /SRC_TYPE="PROD_X:C" –
   /DST_TYPE="PROD_X:H"
```

Parameters and qualifiers

- `<relationship_id>`

  Specifies the identifier of relationship to be created.

- `/DESCRIPTION=<description>`

  Specifies a description for the definition of the relationship type and is restricted to 240 characters.

- `/SRC_TYPE=<product-id><item-type>`

  Specifies the source product and item type from which the link will start.

- `/DST_TYPE=<product-id><item-type>`

  Specifies the destination product and item type at which the link will end or point.

- `/INHERIT_FROM`

  Specifies that a new item revision inherits the previous revision's children.

- `/INHERIT_TO`

  Specifies that a new item revision inherits the previous revision's parents.

Description

The command DIR is used to define a relationship between Dimensions item types. This relationship may be across products. Once a relationship has been defined it can be used by the RII and XII commands to create and delete relationships. (The RIR command is used to remove a relationship definition.)

Constraints

Only users with the appropriate management privileges can run this command.

DLCA - Download to Library Cache Area

```
[/WORKSET=<project-spec>] or [/BASELINE=<baseline-spec>]
/AREA=<library cache name>
/USER_ITEMLIST=<file with item specs>] or [ITEM_LIST=<list of item
    specs>]
```

Description

Updates a single library cache area with a list of one or more Dimensions items. If you execute the command without any qualifiers, the library cache area that is associated with the current project or stream is updated with all items.

Example

```
DLCA /WORKSET="PROD30:PRJ30" /AREA="LCA30" /ITEM_LIST=(FORGE:FILE2 TXT-
    113668470X280X8.A-SRC;1, )
```

Parameters and Qualifiers

- `/WORKSET=<project-spec>`

  Specifies the project or stream from which to download Dimensions items. If not specified, items are downloaded from the current project or stream.

- `/BASELINE=<baseline-spec>`

  Specifies a baseline from which to download Dimensions items. If not specified, items are downloaded from the project or stream specified in `/WORKSET`.

- `/AREA=<library cache name>`

  Specifies a library cache area.

- `/USER_ITEMLIST=<file with item specs>`

  Specifies the name of a file containing a list of items to be downloaded.

- `/ITEM_LIST=<list of item specs>`

  Specifies a list of items to be downloaded from a project (or stream).

Additional Examples

- To populate a specific library cache area with the tip revisions from a project that is not the current project (or stream):

  ```
  dlca /workset=PROD1:PRJ1 /area=LCA
  ```

- To populate a library cache area with the items specified in a file:

  ```
  dlca /workset=PROD1:PRJ1 /area=LCA /
      user_itemlist="c:\temp\specs.txt"
  ```

- To populate a library cache area with specific items from a project (or stream):

  ```
  dlca /workset=PROD1:PRJ1 /area=LCA /item_list=(PROD1:ITEM1 TXT.A-
      SRC;1,PROD1:ITEM2 TXT.A-SRC;2)
  ```

- To populate a library cache area with all revisions from a baseline:

  ```
  dlca /baseline=PROD1:B1 /area=LCA
  ```

DLGC – Delegate Request

```
<request-id>
[/SITE=<site_id> or /ROLE=<role> /USER_LIST=(<user1>,<user2>,...)]
[/CAPABILITY=<capability>]
[/ADD or /REPLACE or /DELETE]
[/[NO]DELEGATE_ITEMS]
```

Examples

```
DLGC PROD_DR_25 /ROLE=REVIEWER -
   /CAPABILITY=P /USER=SMITH

DLGC PAYROLL_TDR_1 /SITE="earth:intermediate@@dim9"
```

Parameters and qualifiers

- `<request-id>`

  Identifies the request for which the delegation is to be made.

- `/SITE=<site-id>`

NOTE Cannot be used with `/ROLE` and `/USER_LIST` (will be rejected).

Delegates the request to a replication site instead of a specific user, where `<site_id>` is one of the following:

- `LOCAL`: a keyword that can be used to set the ownership to the local base database
- `<node_name>:<dbname>@@<dsn>`

NOTE @@ is used because @ is the default Dimensions escape character for the command line.

`<node_name>`	=	the node name
`<dbname>`	=	the base database
`<dsn>`	=	the database data source name

The DLGC command does not perform an immediate replication and determines who will be the owner when the next scheduled replication occurs.

- `/ROLE=<role>`

NOTE If specified `/SITE` is rejected.

Identifies the role title to be delegated.

- /USER_LIST=(<user1>,...)

NOTE If specified /SITE is rejected.

Identifies by login user name one or more Dimensions users to whom delegation of the role is to be made for this request.

- /CAPABILITY=<capability>

  Specifies that this role delegation is to be **P** for Primary, **S** for Secondary (default) or **L** for Leader. See the AUR command for description of these role types.

  Only one user and only /ADD or /REPLACE are valid for the delegation of the Primary capability.

NOTE Is ignored when /SITE is specified.

- /ADD

  Adds this role for this request to the specified users (in addition to users that currently have this role).

  Is the default if none of /ADD, /REPLACE and /DELETE are specified.

- /REPLACE

  Replaces this role for this request with the specified users instead of the users who currently have it.

- /DELETE

  Deletes the specified users from the list of users who have this role for this request.

- /DELEGATE_ITEMS

  When delegating a request to a user, automatically delegates those items related as Affected and In Response To to the same user. This qualifier invokes the DLGI command logic for all items related to that request using the qualifiers passed down to DLGC.

Description

Delegate a request when you want to assign a request to another user. You can also change role assignments.

Constraints

This command can be run only by a user with the appropriate management privileges or by users for whom the request to be delegated is in their pending list.

DLGI – Delegate Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/ROLE=<role>
/USER_LIST=(<user1>,<user2>,...)
[/WORKSET=<project-spec>]
[/CAPABILITY=<capability>]
[/ADD or /REPLACE or /DELETE]
```

Example

```
DLGI  PROD:"QUERY RELEASE"-SRC -
   /ROLE=REVIEWER /CAPABILITY=P -
   /USER=SMITH /FILENAME=qr.c
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

  `<item-id>` identifies the new item within the product.

  `<variant>` if omitted, the default (specified when the product was defined) is used.

  `<revision>` defaults to **1**, if omitted.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  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. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/ROLE=<role>`

  Identifies the role title to be delegated.

- `/USER_LIST=(<user1>,...)`

  Identifies by login user name one or more Dimensions users to whom delegation of the role is to be made for this item.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  Optionally specifies the project or 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.

- `/CAPABILITY=<capability>`

  Specifies that this role delegation is to be **P** for Primary or **S** for Secondary (default) or **L** for Leader. (See AUR command for description of these role types.)

  Only one user and only `/ADD` or `/REPLACE` are valid for delegation of Primary capability.

- `/ADD`

  Specifies that the user(s) are to have this role for this item in addition to those who already have it.

  This is the default, if none of `/ADD`, `/REPLACE`, `/DELETE` is specified.

- `/REPLACE`

  Specifies that the user(s) are to have this role for this item instead of those who already have it.

- `/DELETE`

  Specifies that the user(s) are to be removed from the list of users who have this role for this item.

Constraints

This command can be run only by a user with the appropriate management privileges or by users for whom the item to be delegated is in their pending list.

DLGS – Delegate Personal Stream

```
/USER=<user ID>
/WORKSET_LIST=("<product>:<stream>", "<product>:<stream>")
```

Description

Delegates a personal stream from one user to another and changes ownership of the stream. For example, a developer is switching to another task so they shelve their changes to a personal stream and delegate it to another user.

Example

```
DLGS /USER=mwatney /WORKSET_LIST=QLARIUS:SHELVING_FEATURE
DLGS /USER=abrown
    /WORKSET_LIST=("QLARIUS:MWATNEY_ENH303","QLARIUS:MWATNEY_ENH263")
```

Parameters and Qualifiers

- `/USER`

  Specifies the user to who the personal stream will be delegated.

- `/WORKSET_LIST`

  Specifies a comma-separated list of personal streams to be delegated.

DMBL – Demote Baseline

```
<baselineName>
/COMMENT=<userComment>
/STAGE=<promotionStage>
/USER_FILENAME=<listFile>
/[NO]QUIET
/AREA_LIST=<areaList>
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
/WORKSET
/SDA_PROCESS=<SDA_Process>
/SDA_COMPONENTS=(<SDA_Component1>=<Version name1>,
    <SDA_Component2>=<Version name2>,...)
```

Example

```
DMBL "QLARIUS:KESTREL_RELEASE_2.1" /COMMENT="Rollback QA environment to
    2.0." /STAGE="DEV" /WORKSET="QLARIUS:KESTREL_BRANCH" /
    SDA_PROCESS="ReleaseAutomation" /
    SDA_COMPONENTS=("BlnComp"="QLARIUS:KESTREL_RELEASE_2.0","3rdPartyCo
    mp"="3.1","DocsComp"="2.0.1")
```

Parameters and Qualifiers

- `<baselineName>`

  Name of a baseline to demote.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the demotion.

- `/STAGE=<promotionStage>`

  Name of the stage to demote the baseline to.

- `/USER_FILENAME=<listFile>`

  A user-specified file containing a list of baselines to be demoted. Allows you to demote multiple baselines in the same operation.

- `/AREA_LIST=<areaList>`

  List of target deployment areas at the specified stage.

- `/[NO]DEPLOY`

  Launches a deployment.

  You cannot use `/NODEPLOY` with `/AREA_LIST`.

- `/DEPLOY_START_TIME`

  The start time for the deployment to begin, use one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS" (Dimensions date time)`
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z" (ISO8601 date time)`

Note that the following formats are not accepted:

- "YYYY-MM-DDTHH24:MI:SSZ" (omission of milliseconds does not work)
- "DD-MM-YYYY HH24:MI:SS"

- /WORKSET

  Specifies a specific project or stream from which to schedule demotion of a baseline. If you do not specify this parameter, the current project or stream is used.

- /SDA_PROCESS=<SDA_Process>

  Specifies the Deployment Automation (DA) application process to run in the environment that is mapped to the stage you are demoting from. Omit this qualifier if you do not want to run an automation during demotion.

- /SDA_COMPONENTS=(<SDA_Component1>=<Version name1>, <SDA_Component2>=<Version name2>,...)

  Specifies DA process components, and the corresponding component versions, to be used during DA process execution. Omitted components are not used in the automation execution. Use "<LATEST>" to specify the latest component version.

Description

This command attempts to roll back a previous, matching deployment. The following rules apply:

- The roll back is performed in all deployment areas at each stage between the current stage (for the deployment) and the target stage for the rollback (but not including the target stage).
- If you do not specify the /STAGE parameter, the stage below the current stage is selected as the target stage.

In a rollback operation, the previous version of each item in the matching deployment is returned to the target area. However, rollbacks may fail silently for many reasons:

- Items to be rolled back have become linked to subsequent changes, for example, later versions of the items have been deployed to the same deployment area.
- The items have already been rolled back.
- The objects to be rolled back do not match a previous deployment. For example, you cannot roll back a single item if it was previously deployed as a part of a baseline deployment.
- A deployment area is offline.
- Objects in the rollback are in use or have other I/O issues.

The /DEPLOY and /AREA_LIST parameters only relate to work done at the target stage. By default, there is no deployment at the target stage. However, if you specify either of these parameters, the items associated with the object being demoted are applied to deployment areas at the target stage. Items are regressed by default and only cause errors if the following symbols have been added to the CM server configuration file:

- To disable regressions, set the following variable to Yes:

  DM_NO_DEPLOYMENT_REGRESSION

- To fail deployments in the event that any have been disallowed, and to display an error message, set the following variable to `Yes`:

  `DM_REGRESSION_ATTEMPT_IS_AN_ERROR`

If you re-run a demotion operation when the current stage is the target stage, a rollback is attempted on all higher areas.

Limitations

If the parent product uses the Deployment Automation (DA) deployment model, the following qualifiers are not supported:

- `/USER_FILENAME`
- `/AREA_LIST`
- `/NODEPLOY`

DMI – Demote Item

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
[<itemSpec>|<fileName>]
/COMMENT=<userComment>
/STAGE=<promotionStage>
/WORKSET=<projectName>
/USER_FILENAME="<listFile>"
/[NO]QUIET
/AREA_LIST="<areaList>"
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
```

Parameters and qualifiers

- `[<itemSpec>|<fileName>]`

  Specifies the file, or item, to be demoted.

- `/COMMENT=<userComment>`

  Describes the reason for the demotion.

- `/STAGE=<promotionStage>`

  Specifies the stage to demote the items to.

- `/WORKSET=<projectName>`

  Specifies the project or stream that contains the items to be demoted.

- `/USER_FILENAME=<listFile>`

  Specifies a file containing multiple items and files to be demoted. Cannot be used with `<itemSpec>|<fileName>`. Each file name must be on a separate line.

- `/AREA_LIST=<areaList>`

  List of target deployment areas at the specified stage.

- `/[NO]DEPLOY`

  Launches a deployment.

  You cannot use `/NODEPLOY` with `/AREA_LIST`.

- `/DEPLOY_START_TIME`

  Specifies the start time of the deployment in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  The following formats do not work:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (no milliseconds specified)
  - `"DD-MM-YYYY HH24:MI:SS"`

Description

See the description for the DMBL command on page 173.

DMRQ – Demote Request

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
<requestId>
/COMMENT=<userComment>
/STAGE=<promotionStage>
/[NO]CANCEL_TRAVERSE
/WORKSET=<projectName>
/USER_FILENAME=<listFile>
/[NO]QUIET
/AREA_LIST=<areaList>
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
```

Parameters and qualifiers

- `<requestId>`

  ID of the request to demote.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the demotion.

- `/STAGE=<promotionStage>`

  Name of the stage to demote the request to.

- `/[NO]CANCEL_TRAVERSE`

  `CANCEL_TRAVERSE` limits demotion to only the specified request.

- `/WORKSET=<projectName>`

  Name of the project or stream that contains the request to demote.

- `/USER_FILENAME=<listFile>`

  A user specified file containing the list of requests to demote. Specifying this option allows you to demote many requests at once.

- `/AREA_LIST=<areaList>`

  List of target deployment areas at the specified stage.

- `/[NO]DEPLOY`

  Launches a deployment.

  You cannot use `/NODEPLOY` with `/AREA_LIST`.

- `/DEPLOY_START_TIME`

  The start time for the deployment to begin, use one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  Note that the following formats are not accepted:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (omission of milliseconds does not work)
  - `"DD-MM-YYYY HH24:MI:SS"`

Description

See the description for the DMBL command on page 173.

DNC – Delete an Existing Network Node Connection

```
/SERVER_NAME=<server_node_name>
/CLIENT_NAME=<client_node_name>
/NWO_NAME=<network_object_name>
```

Example

```
DNC /CLIENT_NAME=TEST_MACHINE.COMPANY.COM -
   /SERVER_NAME=MACHINE.COMPANY.COM -
   /NWO_NAME=PCMS_SDP
```

This command enables you to delete an existing network node connection from an installation. See the *System Administration Guide* for details.

DNDO – Delete an Existing Network Node Object

```
/NN_NAME=<network_node_name>
/NWO_NAME=<network_object_name>
```

This command enables you to delete an existing network node from an installation. See the *System Administration Guide* for details.

DNN – Delete an Existing Network Node

```
/NN_NAME=<network_node_name>
```

Example

```
DNN /NN_NAME=MACHINE.COMPANY.COM
```

This command enables you to delete an existing installation network node. See the *System Administration Guide* for details.

DNP – Define New Product

```
<product-id>
[/BASED_ID=<based-on-product ID>]
[/STRUCTURE=ALL]
/DESCRIPTION=<description>
/PRODUCT_MANAGER=<product-manager>
[/PARTS_CONTROLLER=<parts-controller>]
[/CHANGE_MANAGER=<change-manager>]
/VARIANT=<variant>
/PCS=<pcs>
[/ATTRIBUTES=(<attribute_id>=<value>,...)]
[/SDA_APPLICATION=<SDA_application_name>]
[/SDA_PROCESS=<SDA_default_process>]
```

Example

```
DNP PRODTEST20 /VARIANT=AAAA /PCS=1 -
   /DESC="PROD Rel 2.0 Test Vehicle" -
   /PRODUCT_MANAGER=SMITH
   /ATRIBUTES=(site=dallas, priority=critical,country_orig=germany)
```

Parameters and qualifiers

- `<product-id>`

  Specifies the ID of the new product.

NOTE Only alpha-numeric and "underscore" (_) characters are permitted.

- `/BASED_ID=<based-on-product>`

  Specifies the ID of an existing product on which the new product will be based.

NOTE If the existing product has upload rules defined they are copied to the new product.

  Default based-on product: `$GENERIC`

- `/STRUCTURE=ALL`

  Copies the part structure from the `<based-on-product>` to the new product.

  Default (do not copy):`/STRUCTURE=NO`

- `/DESCRIPTION=<description>`

  Specifies a description of the new product.

- `/PRODUCT_MANAGER=<product-manager>`

  Assigns the role of `PRODUCT-MANAGER` to the specified user.

  By default a user with the role `PRODUCT-MANAGER` is automatically assigned the roles of `PCMS-ROLE-MANAGER` and `PCMS-PART-MANAGER` for the associated project or stream. Other users can also be assigned the `WORKSET-MANAGER` role using the DUR command.

- `/PARTS_CONTROLLER=<parts-controller>`

  Assigns the role `PARTS-CONTROLLER` to the specified user.

  Default role: `<product-manager>`

- /CHANGE_MANAGER=<change-manager>

  Assigns the role CHANGE-MANAGER to the specified user.

  Default: <product-manager>

- /VARIANT=<variant>

  Specifies the default <variant> used for this product when new design parts and items are created.

- /PCS=<pcs>

  Specifies the PCS used for this product when new design part variants are created.

- /ATTRIBUTES=(<attribute_id>=<value>,...)

  Specifies a value for a new mandatory product level attribute. If you do not specify a value an error message is issued.

<attribute_id>	Specifies the name of the new attribute.
<value>	Specifies the attribute value.

- [/SDA_APPLICATION=<SDA_application_name>]

  Specifies a Deployment Automation (DA) application to be associated with the product. The application will be used for deployment during promotion and demotion in this product. Omit this qualifier to use Dimensions CM deployment model.

- [/SDA_PROCESS=<SDA_default_process>]

  Specifies the default DA application process name to be executed when running a promotion.

Constraints

Only users with the appropriate management privileges can run this command.

DNWO – Delete an Existing Network Object

```
/NWO_NAME=<network_object_name>
```

Example `dnwo /nwo_name=pcms_sdp`

This command enables you to delete an existing network object from an installation. See the *System Administration Guide* for details.

DOS – Delete an Existing Operating System

```
/OS_NAME=<operating_system_name>
```

This command enables you to delete an existing operating system from an installation. See the *System Administration Guide* for details.

DOWNLOAD – Download Project or Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE When issuing the UPDATE command, and a project is specified, or the user's current project/stream is a project, this command will invoked instead.

```
[<file-spec> or /DIRECTORY=<directory-spec> or
    /USER_FILELIST=<filelist-file> or /USER_ITEMLIST=<itemlist-file>]
[/[NO]RECURSIVE]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/[NO]TOUCH]
[/[NO]OVERWRITE]
[/LOGFILE=<file-spec> or /SCRIPTFILE=<file-spec>]
[/CODEPAGE=<code-page> or DEFAULT]
[/WORKSET=<project-spec> or /BASELINE=<baseline-spec>]
[/USER_DIRECTORY=<directory-path>]
[/RELATIVE_LOCATION=<directory-spec>]
[/[NO]CONFLICT_CHECK]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/[NO]METADATA]
[/[NO]REFACTORING]
[/EXECUTE]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
[/PERMS=KEEP|READONLY|WRITABLE]
```

Examples

- `DOWNLOAD`

  Downloads the tip of the current project into the working location.

- `DOWNLOAD /DIRECTORY="build"`

  Downloads the tip of "build" into the working location.

- `DOWNLOAD /DIRECTORY=java_build /BASELINE="pvcs:dm10_build_5_11" /USER_DIRECTORY="C:\projects\dm10"`

  Downloads the `java_build` directory from a baseline into the specified directory.

- `DOWNLOAD "C:\temp\build\build.mk" /TOUCH /BASELINE="PVCS:DM10 TIER1 FINAL"`

  Assuming that the project user directory is set to `C:\temp`, this command will update (if necessary) the `C:\temp\build\build.mk` file with the baseline item revision with file name `build\build.mk` from the PVCS:DM10 TIER1 FINAL baseline into the `C:\temp` directory. The modification time of the updated file will be set to the current system time.

- `DOWNLOAD /DIRECTORY="build\include" /TOUCH /WORKSET="PVCS:DM10"`

All files found in the project directory `build\include` and in any directories below it will be considered for download into the current working location. If the user has locally modified any matching files in the current working location, these files will not be updated.

Parameters and qualifiers

- `<file-spec>`

  Specifies the name of the file to be downloaded. (The Dimensions `node::` syntax is also valid.)

- `<directory-spec>`

  Specifies a directory path. It is matched to the corresponding project or baseline directory, the files in that project or baseline directory are enumerated, and each one that has not been been modified is downloaded.

  You can specify the directory using the Dimensions `node::` syntax. It can also be a path relative to the user working location.

- `/USER_FILELIST=<filelist-file>`

  Specifies a file containing a list of file names to be downloaded from the project or baseline. Each file name must be on a separate line.

  File names may be specified as either relative or absolute paths. If the path is absolute, it is interpreted as a full project or baseline file path; otherwise, Dimensions obtains the project or baseline path by mapping the file name to the operation root directory, which is the directory specified by the /USER_DIRECTORY qualifier (if any) or the current working location as specified by the last SCWS command. If such a mapping is not possible, the file name is ignored.

- `/USER_ITEMLIST=<itemlist-file>`

  Specifies a file containing 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 using the complete item specifications. Each item specification must be on a separate line. There is no need to use double quotes with item specifications.

- `/[NO]RECURSIVE`

  If `/DIRECTORY` is specified and this qualifier is not present, all files that have not been modified in all directories beneath the one specified are downloaded. `/NORECURSIVE` specifies that only files at the specified directory level are downloaded.

  Default: `/RECURSIVE`

- `/[NO]EXPAND`

  Specifies substitution variable expansion.

  The default is to expand item header substitution variables provided the item type was defined in the process model with **Enable item header substitution**.

- `/[NO]REEXPAND`

  Specifies whether item substitution header variables will be re-expanded for existing files in the work area even if the item files have not changed in the repository.

  Default: `/NOREEXPAND`

- `/[NO]TOUCH`

  Specifies whether to apply the system date/time to each file being downloaded.

Default: /TOUCH

- /[NO]OVERWRITE

  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 fetched (that is, files that have different <product>:<item-id>.<variant>-<type> pairs).

  If /OVERWRITE is specified, DOWNLOAD overwrites these files with the content of corresponding project or baseline item revisions.

- /LOGFILE=<file-spec>

  Generates a log file at the specified file location containing the results of all the individual Dimensions operations executed through this command.

- /SCRIPTFILE=<file-spec>

  Generates a script file at the specified file location containing the individual Dimensions operations that would have been executed through this command. The script file contains commands that have the *same* affect as DOWNLOAD, thought the operations are not executed. The commands in the script file do not necessarily have the same qualifiers as the DOWLOAD command.

  Note that the resulting script cannot be run against a stream, only a project.

- /CODEPAGE=<code-page>

  Specifies the code page to be associated with the item.

- /WORKSET=<project-spec>

  Specifies the project from which to download the files. If this parameter is not specified, files are downloaded from the current session project.

- /BASELINE=<baseline-spec>

  Specifies the baseline from which to download the files. If this parameter is not specified, files are downloaded from the project specified via /WORKSET or the current session project.

- /USER_DIRECTORY=<directory-path>

  Use /USER_DIRECTORY=<directory-path> to specify a download directory other than the current working location. For example, the following command downloads the project into C:\temp regardless of what the current working location is:

  ```
  DOWNLOAD /USER_DIRECTORY="C:\temp"
  ```

  The following command downloads the project into the /tmp directory on host "hostname":

  ```
  DOWNLOAD /USER_DIRECTORY="hostname::/tmp"
  ```

  The following command downloads the project into the src directory inside the area_name area:

  ```
  DOWNLOAD /USER_DIRECTORY="area_name::src"
  ```

- /RELATIVE_LOCATION=<directory-spec>

  Specifies a project, stream, or baseline directory which is to be made the "virtual" project, stream, or baseline root directory for the duration of this command. If this

parameter is given, then the paths given in <file-spec> or /DIRECTORY must be relative to the directory specified with /RELATIVE_LOCATION.

- /[NO]CONFLICT_CHECK

  Searches for unresolved merge conflicts for each item revision to be fetched. If any unresolved conflicts are found, Dimensions issues a warning and ignores the corresponding revision. If you specify /REFACTORING this qualifier is ignored.

  Default: /NOCONFLICT_CHECK

- /FILTER=<filter-name>

  Only gets files that satisfy the criteria specified in the <filter-name> area filter.

- /USER_FILTER=<filter-file-spec>

  Specifies the name of a local file containing the definition of a file filter to be used when getting files or checking in files. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502.

  Only files matching the filter (and not excluded by the filter) will be downloaded when a user filter is specified.

- /[NO]METADATA

  Disables the creation and usage of metadata files in the local work area.

  Default: /METADATA

- /[NO]REFACTORING

  Specifies whether non-conflicting refactoring changes are to be applied to the files and folders in the local work area. If you specify this qualifier then /CONFLICT_CHECK is ignored.

  Default: /NOREFACTORING

- /EXECUTE

  Use with the /SCRIPTFILE qualifier to force the transfer of files while generating a script file containing the equivalent Dimensions commands.

- /EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.

UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

- /PERMS=KEEP|READONLY|WRITABLE

  Sets permissions for files that you download:

  KEEP: retains the current permissions

  READONLY: makes the files read only

  WRITABLE: makes the files writeable

Description

The DOWNLOAD command downloads repository content (a project or a baseline) to the developer's workspace.

This command compares each item revision selected by the passed parameters with the corresponding on-disk files. If the disk file has been locally modified (by the use of optimistic locking), or does not have Dimensions metadata, or has been locally checked out, then the command issues a warning and skips the file. Otherwise, DOWNLOAD overwrites the disk file with the corresponding item revision.

The DOWNLOAD command represents the most efficient way of getting multiple item revisions from the Dimensions repository. It is particularly optimized for transferring many relatively small files across high-latency wide area network (WAN) connections, where it could be several times faster than an equivalent script of separate FI commands.

Note that for streams you should use the UPDATE command.

DPB – Deploy Baseline

NOTE

This command is not available in the Issue Management–only licensed version of Dimensions.

This command is not supported in products that use the Dimensions Automation deployment model.

```
<baseline-spec>
[/WORKSET=<project-spec>]
[/STAGE=<new-stage>]
[/COMMENT=<comment>]
```

Example

```
DPB "PVCS:DM10 COMMON TOOLS" /STAGE=APPROVED
```

Parameters and qualifiers

- `<baseline-spec>`

  Baseline specification.

- `<project-spec>`

  Specifies the root project to be used for this command. If the /WORKSET qualifier is omitted, the DPB command uses the current project or stream.

- `<new-stage>`

  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>`

  Textual comment.

Description

The DPB command deploys the specified baseline to the specified stage in the context of the specified project or stream. If the project/stream has deployment areas assigned to it, these areas are updated as the baseline is deployed from one stage to another.

If a deployment area has an area filter, the deployment area is updated only with item revisions that match the area filter's rules. If a deployment area has transfer scripts associated with it, these scripts are executed before and after all item revisions are transferred to and / or removed from the deployment area. When transfer scripts are expanded for execution in an area, a number of predefined template variables are set by Dimensions. In particular, when items are deployed as part of the DPB command, the `DMBLNPRODUCT` and `DMBLNID` template variables will contain the product and ID of the baseline that contains the items.

See "DPI – Deploy Item" on page 194 for details on the template variables.

The project in which the baseline is deployed must either be a standalone project or a root project with sub-projects (in other words, the project in which the baseline is to be deployed may not be a sub-project attached to another project). In order for the baseline to be deployable in this project, each item revision included in the baseline must also be

present in the project (or the namespace of the root project in case the specified project is a root project with sub-projects). The project must follow the manual deployment model.

Constraints

Only users with the "Deploy Baseline to Next Stage" or "Deploy Baseline to Any Stage" privileges can run this command.

DPI – Deploy Item

NOTE

This command is not available in the Issue Management–only licensed version of Dimensions.

This command is not supported in products that use the Dimensions Automation deployment model.

```
<item-spec>
[/FILENAME=<file-name>]
[/WORKSET=<project-spec>]
[/STAGE=<new-stage>]
[/COMMENT=<comment>]
```

Example

```
DPI PROD:"QUERY RELEASE".AAAA;2 /FILENAME=query.c /STAGE=APPROVED
```

Parameters and qualifiers

- `<item-spec>`

  Item specification. Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if `<file-name>` is specified.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project/stream. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified within the `<item-spec>` argument.

- `/WORKSET=<project-spec>`

  Specifies the project or stream to be used for this command. If the /WORKSET qualifier is omitted, the DPI command uses the current project/stream.

- `/STAGE=<new-stage>`

  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 DPI command to succeed.

- `/COMMENT=<comment>`

  Textual comment.

Description

The DPI command deploys the specified item revision to the specified stage and updates deployment areas accordingly. If a deployment area has an area filter, the deployment

area is updated only if the item revision matches the area filter's rules. If a deployment area has transfer scripts associated with it, these scripts are executed before and after the item revision is transferred to or removed from the deployment area.

When transfer scripts are expanded for execution in an area, a number of predefined template variables are set by Dimensions. The following is a list of such variables:

```
DMSERVER
    Dimensions server hostname.
DMBLNPRODUCT
    Product of the baseline being deployed, if any.
DMBLNID
    ID of the baseline being deployed, if any.
DMREQUEST
    ID of the request being deployed, if any.
DMAREA
    Area ID.
DMDIR
    Full path to the directory that the file is being transferred to or
    removed from.
DMFILENAME
    File name.
DMCOMMENT
    /COMMENT qualifier value from the current operation.
DMTRANSFERTYPE
    Operation type: "c" for copy into area; "r" for "removal from area.
DMREVISION
    Revision string of the deployed item revision.
DMBRANCH
    Branch name portion of the DMREVISION variable.
DMFORMAT
    Data format of the deployed item revision.
DMPREFIX
    Deployed item revision file name minus the extension.
DMSUFFIX
    Deployed item revision file-name extension.
DMWSPRODUCT
    Product of the current project.
DMWSID
    ID of the current project.
DMPRODUCT
    Product of the deployed item revision.
DMID
    Item ID of the deployed item revision.
DMVARIANT
    Variant of the deployed item revision.
DMTYPE
    Item type of the deployed item revision.
DMCTIME
    Date and time of the transfer.
DMISDIR
    Y or N.
```

DMCMDREQUESTS
    list of change documents associated with this deployment.
DMCOMMAND
    Command used to initiate this deployment. Includes most but not all keywords—sensitive keywords areomitted.
DMAREANODE
    Node on which the area is located. This can be a physical or logical location depending on how your system is set up.
DMCERTIFICATE
    This is generated for MVS deployment or when a REXEC has specified /CAPTURE. It provides a mechanism for logging back in to the server and is needed to return script-execution success/failure in the MVS case. (Using it for another purpose is possible but requires careful testing.)
DMCMDCOMMENT
    If a deployment command that initiated the deployment contained a /COMMENT qualifer, this variable contains that comment.
DMDJQJOBID
    Deployment job UID.
DMJOBID
    UID of the REXEC JOB_QUEUE record. It is the number used by the RLIST and RSTAT commands.
DMJOBTYPE
    This variable can be used to determine the deployment process type. The possible values are ROLLBACK, START_BUILD, AUDIT, COLLECTION, END_BUILD, START_TRANS, END_TRANS, COMMAND, START_SCHEDULED_JOB, and UNKNOWN.
DMSTAGE
    Current stage of the area undergoing the deployment.

For detailed information about writing deployment area scripts, and an overview of the associated templating language, see the *Developer's Reference*.

Constraints

Only users with the "Deploy Item to Next Stage" or "Deploy Item to Any Stage" privileges can run this command.

DPL – Define Product Libraries

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<product-id>
/ITEM_TYPE=<item-type>
[/DELTA]
[/LIBRARY=<directory-name>]
[/NETWORK_NODE=<network-node-name>]
[/PROTECTION=<protection>]
[/ADD or /UPDATE or /DELETE]
```

NOTE For information about managing Dimensions item libraries on a z/OS mainframe for access through Dimensions for z/OS, see the *Dimensions for z/OS User's and Administrator's Guide*.

Example

```
DPL PROD /ITEM_TYPE=DATA /NET=eldarmar -
/LIB="/usr/PROD/library/item/data/"
```

Parameters and qualifiers

- `<product-id>`

  Identifies the product in which an item library is (to be) defined.

- `/ITEM_TYPE=<item-type>`

  Specifies that the library is for items of this type.

  `/ITEM_TYPE=*` may be used to specify a default item library.

- `/DELTA`

  Indicates that the item-library is (to be) a delta library.

  If omitted, the item-library is (to be) a normal library.

NOTE

- If `/UPDATE` is specified, `/DELTA` must be specified if and only if the existing item library is a delta library. It cannot be specified or omitted to change a normal library to a delta library or vice versa.
- The compress storage option is not available for delta library storage (see the *Process Configuration Guide*).

- `/LIBRARY=<directory-name>`

IMPORTANT!

Item libraries must not reside in the root directories of Windows drives or shares! This is unsupported, and many operations may fail.

When the Dimensions server is installed on Windows 2008 server, item libraries cannot be located in the folders beneath the *Program Files* folder.

names the directory to hold the specified library as follows:

UNIX	the absolute directory path, ending with the character / e.g. `/usr/PROD/lib/`
Windows	the absolute directory path, ending with the character \ e.g. `c:\PROD\lib\`

The library `<directory name>` is **not** required when a library definition is being deleted.

NOTE For information about utilizing item libraries on NAS/UNC hosts see the *System Administration Guide*.

- `/NETWORK_NODE=<network-node-name>`

  Identifies to which computer and file system in a network `<directory-name>` is applicable.

  The `<Network_Node>` must be stated even if a local machine is being used.

- `/PROTECTION=<protection>`

  Specifies the protection code for the items library directory in the standard operating system format in UNIX.

  If omitted, the defaults used are:

  UNIX: rwx,rx,r

  For Windows this qualifier **must** be omitted. Once a library directory has been created (and before it is used), the owner should specify its protection by creating an ACL (Access Control List) for it. See separate sub-section below for additional information.

- `/ADD or /UPDATE or /DELETE`

  Indicates whether this library definition is being added, altered or removed.

  The default is `/ADD`.

Remote Storage

Item libraries can be stored on remote servers accessed using UNC path names or via mapped drives. The remote location could be a Windows Network Access Storage (NAS) system, a fibre-connected storage area network (SAN), or a network share on a PC.

Protecting the Item Library Files from Unauthorized Changes on Windows

The Dimensions item libraries are protected from unauthorized changes by setting an ACL on each directory which is defined to hold an item library. This must be done manually (i.e. as a supplementary operation external to Dimensions processing), using the Windows Explorer. Because ACLs are allowed only on files on a disk with an NTFS file system, it is recommended that item libraries are not defined on disks with FAT file

systems, as there would be no way to protect the item libraries from unauthorized changes.

An ACL with the following attributes is recommended:

- System: Full Control
- Administrators: Read Access
- Owner: Read Access

This will ensure that only the Dimensions Listener Service is able to write files into these directories.

Some additional users could be granted Read access to the Item files by adding a group or users (using the Windows Explorer Security | Permissions menu option).

Permissions Only the 'System' user is permitted to Write, Change and Delete ACLs.

Constraints

You cannot define operating system directories for request libraries. Requests are automatically stored by Dimensions in the RDBMS database.

This command can be run only by a user with the appropriate management privileges but **not** while other Dimensions users are using the libraries. **Such operations could cause fatal library access errors**.

Dimensions does not maneuver the contents of the library to correspond with any revisions made—it issues a warning message advising the user with the role of `PRODUCT-MANAGER` to perform this task.

NOTE In a pure LDAP environment, this command requires a local operating-system account.

DPR – Deploy Request

NOTE

This command is not available in the Issue Management–only licensed version of Dimensions.

This command is not supported in products that use the Dimensions Automation deployment model.

```
<request-id>
[/WORKSET=<project-spec>]
[/STAGE=<new-stage>]
[/COMMENT=<comment>]
[/[NO]CANCEL_TRAVERSE]
```

Example

```
DPR PVCS_EC_100 /STAGE=APPROVED
```

Deploys all revisions related to PVCS_EC_100 as IRT to the APPROVED stage.

Parameters and qualifiers

- `<request-id>`

  Request identifier.

- `/WORKSET=<project-spec>`

  Specifies the project or stream to be used for this command. If the /WORKSET qualifier is omitted, the DPR command uses the current project/stream.

- `/STAGE=<new-stage>`

  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 DPR command to succeed.

- `/COMMENT=<comment>`

  Textual comment.

- `/CANCEL_TRAVERSE or /NOCANCEL_TRAVERSE`

  Specifies whether the hierarchy of child requests is traversed.

  The DPR command traverses the child requests by default. CANCEL_TRAVERSE reverses the default behavior.

CAUTION! The old qualifiers /REPORT and /USER_FILENAME are no longer supported and are ignored.

Description

This command finds all items related as IRT ("In Response To") to the specified request and all of its child requests. It then deploys and promotes these items to the specified lifecycle stage in the context of the current project. If the project's deployment model is set to manual, copy on deploy is on, and the project's deployment method is set to request, then this command also finds all refactoring changes performed against this request and any of its child requests and deploys these refactoring changes to the specified stage in the context of the current project. If the current project includes

deployment areas, these areas are updated as the request is deployed from one stage to another.

In order to be deployable, a request must be related to a project and the project must use the request deployment method. The request is then deployable only in the context of that project. When the hierarchy of child requests is traversed, only such child requests that are related to the same project as the main request or that are related to a project in the same project tree as the main request's project will be considered for deployment. If a request has any items checked out against it, it cannot be deployed.

If a deployment area has an area filter, the deployment area is updated only with item revisions that match the filter's rules. If a deployment area has transfer scripts associated with it, these scripts are executed before and after all item revisions/folders are transferred to and / or removed from the area. When transfer scripts are expanded for execution in the area, a number of predefined template variables are set by Dimensions. In particular, when items are deployed as part of the DPR command, the `DMREQUEST` template variable will contain the request contributing the item being deployed. When directory creation or removal is deployed as part of the DPR command, the new DMISDIR variable will contain the "Y" (for items this variable will contain "N").

See "DPI – Deploy Item" on page 194 for details on the template variables.

Constraints

Only users with the "Deploy Request to Next Stage" and "Deploy Request to Any Stage" privileges can run this command.

DPROJ – Define a Dimensions Project

NOTE This command is no longer available. Use DWS to define a new project and Dimensions Build to define a build project.

See the DWS command and the *Dimensions CM Build Tools User's Guide*

DPRP – Define Preservation Rules Policy

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<policy-spec>
[/DESCRIPTION=<description>]
[/DEFAULT_RULE={NORMAL|EXTERNAL|PLACEHOLDER}]
[/ITEM_TYPE={SOURCE|LISTING|etc} /RULE={NORMAL|EXTERNAL|PLACEHOLDER}]]
[/ADD | /DELETE | /UPDATE]
```

IMPORTANT! Depending on whether or not the qualifier /ITEM_TYPE is specified, the /ADD, /DELETE, and /UPDATE qualifiers behave differently. See the qualifier descriptions later in this section for details.

Examples

1 The following commands:

```
DPRP PAYROLL:DEFAULT_POLICY -
/DESCRIPTION="Default site policy" -
/DEFAULT_RULE=NORMAL

DPRP PAYROLL:DEFAULT_POLICY -
/ITEM_TYPE=LISTING /RULE=EXTERNAL

DPRP PAYROLL:DEFAULT_POLICY -
/ITEM_TYPE=OBJ /RULE=PLACEHOLDER /ADD
```

define a new preservation policy that conjunctionally specifies:

- That the default rule for all item types is to be preserved as normal item revisions in an item library, and
- that item revisions of type LISTING are to be preserved as external items, and
- that item revisions of type OBJ are to be preserved as placeholder items.

2 The following command deletes a rule for item type PAYROLL:OBJ

```
DPRP PAYROLL:DEFAULT_POLICY -
/ITEM_TYPE=OBJ /DELETE
```

3 The following command updates a rule for item type PAYROLL:LISTING

```
DPRP PAYROLL:DEFAULT_POLICY -
/ITEM_TYPE=LISTING /RULE=NORMAL /UPDATE
```

4 The following command updates the description of a preservation policy:

```
DPRP PAYROLL:DEFAULT_POLICY -
/DESCRIPTION="default site preservation policy" -
/DEFAULT_RULE=PLACEHOLDER /UPDATE
```

5 The following command deletes the preservation policy:

```
DPRP PAYROLL:DEFAULT_POLICY /DELETE
```

Parameters and qualifiers

- `<policy-spec>`

  comprises `<product-id>:<policy-id>` where:

  - `<product-id>`

    Specifies the product for which the policy is defined.

  - <policy-id>

    Specifies the policy identifier.

- `/DESCRIPTION=<description>`

  Specifies the description.

- `/DEFAULT_RULE={NORMAL|EXTERNAL|PLACEHOLDER}`

  Specifies the default preservation rule for this policy; namely:

  - `/DEFAULT_RULE=NORMAL` for specifying normal items (this is the default and may be omitted), or
  - `/DEFAULT_RULE=EXTERNAL` for specifying external items, or
  - `/DEFAULT_RULE=PLACEHOLDER` for specifying placeholder items.

NOTE Only build targets generated outside a build area can be preserved as external items.

- `/ITEM_TYPE={SOURCE|LISTING|etc}`

  Specifies the item type name (within the product in which the policy is defined) for which a preservation rule is defined. For example, SOURCE specifies the `<product-id>:SOURCE` item type.

- `/RULE-{NORMAL|EXTERNAL|PLACEHOLDER}`

  Specifies whether built targets of the above type are to be preserved in a Dimensions item library, namely:

  - `/RULE=NORMAL` for specifying normal items (this is the default and may be omitted), or
  - `/RULE=EXTERNAL` for specifying external items, or
  - `/RULE=PLACEHOLDER` or specifying placeholder items.

NOTE Only build targets generated outside a build area can be preserved as external.

- `/ADD`

  if `/ITEM_TYPE` is not specified, specifies whether to add a new preservation policy. This is the default, and may be omitted. Otherwise, if `/ITEM_TYPE` is specified, this qualifier specifies whether to add a preservation rule for the specified item type.

- /DELETE

  if /ITEM_TYPE is not specified, specifies whether to delete the preservation policy and all associated rules. Otherwise, if /ITEM_TYPE is specified, this qualifier specifies whether to delete a preservation rule for the specified item type.

- /UPDATE

  if /ITEM_TYPE is not specified, specifies whether to update the description of the preservation policy. Otherwise, if /ITEM_TYPE is specified, this qualifier specifies whether to update a preservation rule for the specified item type.

Description

The DPRP command defines a preservation rules policy within a Dimensions product. The policy has an identifier, a description, and a default rule; optionally, it can contain a list of additional preservation rules (exceptions to the default rule).

A preservation rule is conceptually similar to an upload rule, and defines how built targets are preserved in the Dimensions product. The built targets, which are mapped by upload rules to a particular item type, can be preserved as:

- normal item revisions stored in an item library, or
- "external" item revisions stored externally, outside of the control of the Dimensions product, or
- "placeholder" item revisions stored as zero-byte assets in the Dimensions item library.

Each built target is mapped by upload rules to a particular Dimensions item type. The rules in a preservation policy define whether each built target of this item type is to be preserved as a normal item revision or either as a "placeholder" or "external" item revision. The difference between "placeholder" and "external" item revisions is:

- External item revisions

  External item revisions represent versioned files whose actual content is stored outside of a Dimensions item library. In other words, an external item revision has external storage. The actual location of an external revision, which is comprised of a network node name and a full path to a file on that network node, is stored in the Dimensions base database. Typically, one would use external item revisions to represent compile and link listings generated as a result of a build on z/OS.

  Physically, an external item revision is represented as a zero-byte file in the item library. An external item revision can only be created for a build output (such as listing) that was generated outside a build area, and each external item revision must represent a unique file. External item revisions can be actioned and promoted from one stage to another; however, the file referred to by the external file revision is not promoted between build areas occurs – promotion is reduced to a status change. AUDIT will ignore external item revisions.

  External item revisions can be gotten (fetched) and checked out (extracted). If you use a Dimensions client to revise an external item revision (by executing RI or UI commands), then a normal item revision will be created. External item revisions are only creatable by build outputs collection.

- External item revisions

  Placeholder item revisions represent versioned files with no content in the Dimensions item library. In other words, a placeholder item revision has no storage at all. Typically, one would use placeholder item revisions to represent intermediate targets and made-of relationships, and thus avoiding the overhead of preserving in Dimensions every build output generated as a result of a build job.

  Physically, a placeholder item revision is represented as a zero-byte file in the item library, and can be created for any build output regardless of its location – inside or outside a build area. Placeholder item revisions can be actioned and promoted from one stage to another; however, since a placeholder item revision does not refer to any files at all, no file promotion between build areas occurs – promotion is reduced to a status change. AUDIT will ignore placeholder item revisions.

  Placeholder item revisions can neither be gotten (fetched) nor checked out (extracted). If you use a Dimensions client to revise a placeholder item revision (by executing RI or UI commands), then a normal item revision will be created. Placeholder item revisions are only creatable by build outputs collection.

A build administrator will be able to assign a preservation rule to a build stage within a project. By default, if no preservation rules are assigned to a build stage within a project, then the Dimensions product will default to capturing all built targets as normal Dimensions items.

Constraints

Only users with the appropriate management privileges can run this command.

DPV – Delete Design Part Variant

```
<part-spec>
```

Example

```
DPV PROD:"RELEASE MANAGEMENT".IBM
```

Parameters and qualifiers

```
<part-spec> comprises:
   <product-id>:<part-id>.<variant>;<pcs>
```

`<variant>` may be omitted if only one exists.

`<pcs>` is ignored. All PCSs of the specified variant are deleted.

Constraints

Only users with the appropriate management privileges can run this command.

The design part must meet the following criteria:

- it is not related to any request (regardless of the status of the request)
- it has no child design parts
- it owns no items
- it is not in any baseline
- it is not the 'top' design part

DREL – Delete Release

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<release-spec>
```

Example

```
DREL PROD:"R M 2.0 FOR HP"
```

Parameters and qualifiers

```
<release-spec> comprises:

    <product-id>:<release-id>
```

Constraints

This command can be run only by a user with the appropriate management privileges on the product that owns the release or the owner of the baseline on which the release was based.

DRSD – Delete an Existing Resident Software Definition

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
/RSD_NAME=<name_RSD>
```

This command enables you to unregister an installation Resident Software Definition (RSD). See the *System Administration Guide* for details.

DS – Delete Stream

```
<stream-id>
```

Example

```
DS teststream1
```

Parameters and qualifiers

```
<stream-id>
```

Is the name of the stream

Constraints

This command can be run only by the user who created the stream, or by a user with the PROJECT-STREAM-DELETE privilege.

This command will first attempt to delete the version branch associated with the stream. If any item revisions have been created with this version branch, the stream cannot be deleted.

DSJ – Delete Schedule Job

```
<job-id>
```

Example:

```
DSJ "MyJobName"
```

Parameters and qualifiers

```
<job-id>
```

Specifies the job-id.

Description

Enables you to delete a scheduled job.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

DUR – Define User Roles

```
/ROLE=<role>
[/DESCRIPTION=<description>]
[/ADD or /DELETE]
```

Example

```
DUR /ROLE=DEVELOPER /DESC="Creates and edits items"
```

Parameters and qualifiers

- `/ROLE=<role>`

  Specifies a role title for use in the lifecycles used in the base database for all products.

- `/DESCRIPTION=<description>`

  This is optional text to explain the purpose of this new role title.

- `/ADD`

  Adds the role definition.

- `/DELETE`

  Indicates that the specified role title is an obsolete one no longer required in this product.

  If omitted, the command identifies a new role-title to be used i.e. `/ADD` is the default.

Constraints

Only users with the appropriate management privileges can run this command.

DUSR – Unregister User

```
<user-id>
[/[NO]KEEP]
```

Description

This command is the same as XREG.

For details, see the *System Administration Guide*.

DVB – Define Version Branch

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<branch-id>
/DESCRIPTION=<description>
[/[NO]LOCK]
```

Example

```
DVB MAINT -
/DESCRIPTION="Principal branch for maintenance work"
```

Parameters and qualifiers

- `<branch-id>`

  unique branch identifier.

NOTE Only alpha-numeric and "underscore" (_) characters can be used to specify the branch-id.

- `/DESCRIPTION=<description>`

  brief description of the purpose for the branch.

- `/LOCK`

  Optional flag to specify that the branch is locked and further development along it cannot take place.

- `/NOLOCK`

  Optional flag, negation of `LOCK` and is the default.

- `No parameters or qualifiers`

  if DVB is executed without parameters or qualifiers, it prints a list of defined branches together with their description and lock status.

Description

The DVB command is used to define a new branch-id within a Dimensions database for use with version commands. This function is available only in command mode.

Once branch-ids are defined, a valid list (subset) of them is assigned to a particular **existing** project by use of the Set Project Attributes (SWS) command. Alternatively, a valid list of branch-ids can be associated to a *new* project at the time the project is defined using the Define New Project (DWS) command.

The description and/or lock status of an existing `branch-id` definition can be modified (set) using the Set Version Branch Flags (SVBF) command.

A `branch-id` definition can be removed by the Remove Version Branch (RMVB) command.

Constraints

Only users with the appropriate management privileges can run this command.

DWP – Delete Whole Product

```
<product-id>
```

Example `DWP PRODTEST20`

The DWP command deletes all the information about a product (items, requests, design parts, any existing product-specific upload rules, etc) from the Dimensions database.

CAUTION! This command must be used with great care as there is no undo operation for it.

NOTE

Although the product is deleted from the database, Dimensions will *not* remove the contents of the item libraries. (However, because requests are stored in the database they, and any associated attachments, *will* be deleted.)

The contents of the item libraries will still be available, and you will be able to back them up or move them to other directories. It is your responsibility to delete their contents.

Any references within a project to a product's items will be automatically removed from that project when the product is deleted.

Constraints

Only users with the appropriate management privileges can run this command.

DWS – Define New Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
/DESCRIPTION=<description>
[/WORKSET=<project-spec>]
[/BASELINE=<baseline-spec>]
[/ATTRIBUTES=(<attr1>,attr2,...)]
[/TYPE=<type-name>]
[/STATUS=<status>]
[/BRANCH|/TRUNK]
[/[NO]AUTO_REV]
[/VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)]
[/DEFAULT_BRANCH=<branch-id>]
[/[NO]USE_LOCAL_STAGES]
[/[NO]PATH_CONTROL]
[/[NO]KEEP_STAGE]
[/[NO]COPY_CONFIGS]
[/[NO]FORCE]
```

Example

```
DWS "PROD_X:TEST_WS" -
/DESCRIPTION="Project for portation work" –
/BASELINE="PROD_X:prod 2.3 beta"
```

Parameters and qualifiers

- `<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

- `/DESCRIPTION=<description>`

  Specifies the description to be attached to the project definition.

- `/WORKSET=<project-spec>`

  Optionally specifies the project on which to base the new project.

- `/BASELINE=<baseline-spec>`

  Optionally specifies the Dimensions baseline on which to base the new project.

- `/ATTRIBUTES=(<attr1>,attr2,...)`

  Specifies the user-defined attribute values for this project.

- `/TYPE=<type-name>`

  Specifies the type of the project. If this qualifier is not specified, the type name WORKSET is used.

- `/STATUS=<status>`

  Allows the new project to be created in either a `LOCKED` or `UNLOCKED` (default) state. The `LOCKED` state prevents new Dimensions items being added to the project (baselining is likely to occur in this state).

- /BRANCH

  Optional qualifier to adopt "branching" for the item revision scheme. This means that if an item-revision is at revision maint#5, and the users decide to stay on this maint branch, then subsequent revisions will be maint#5.1, maint#5.2, maint#5.3 etc.

- /TRUNK

  Optional qualifier to adopt "trunking" for the item revision scheme. This means that if an item-revision is at revision maint#5, and the users decide to stay on this maint branch, then subsequent revisions will be maint#6, maint#7, maint#8 etc.

- /AUTO_REV

  Optional qualifier to tell Dimensions to automatically generate a new revision each time an item is edited/updated. If this is specified, Dimensions CM calculates revision strings automatically when you create a new item revision.

- /NOAUTO_REV

  Optional qualifier to tell Dimensions **not** to automatically generate a new revision each time an item-spec is edited/updated, and instead request the user to supply a revision number.

- /VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)

  Identifies one or more branches – each previously defined in a Define (Item) Version Branches (DVB) command – that are to be valid for new item revisions created in this new project. If this parameter is omitted, an empty list is created – the Set Project Attributes (SWS) command can subsequently be used, if desired, to associate a valid list of branch-ids to the project created here.

  This list defines the branches on which newly created item revisions can be placed.

  If the project is defined with **one or more** valid branches, every **new** item revision in the new project must use one of these branch-ids.

  If the project is defined with *no* valid branches, new revisions with no branch-ids in them can continue to be used

- /DEFAULT_BRANCH=<branch-id>

  Selects, from the valid list of branch-ids, the <branch-id> to be the default branch. If a default branch-id is not defined, the first branch-id in the valid-list of branch-ids is taken as the default.

- /[NO]USE_LOCAL_STAGES

  A deployment-related option.

  - (Default) /USE_LOCAL_STAGES

    Preserves an item revision's stage in the local project/stream. The stage is not affected even when stages in the GSL are associated with states in its lifecycle.

    NOTE: The same item revision can be at different stages in different projects/streams.

  - /NOUSE_LOCAL_STAGES

    Changing an item revision's stage in a project/stream also changes its stage in all projects/streams that do not use local stages. This is not a recommended best practice.

    Note: Not supported by Deployment Automation (DA).

- /[NO]PATH_CONTROL

  Specifies whether a request is required to perform refactoring operations in this project.

- /KEEP_STAGE

  When creating a project based on an existing project, specify this optional qualifier to control the stages of the items in the new project.

  - Use /NOKEEP_STAGE to reset the stages of all the items in the new project to the initial stage.
  - Use /KEEP_STAGE to keep the stages of the item revisions from the source project.

  This qualifier can only be used when the new project uses the manual deployment model.

  Default (when the qualifier is not specified): /KEEP_STAGE

- /[NO]COPY_CONFIGS

  Specifies whether build configurations will be copied from the project referenced by the /WORKSET parameter to the new project.

  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.

- /[NO]FORCE

  When copying build information from the project referenced by the /WORKSET parameter to the new project, this option specifies whether the DWS will abort when the first error is encountered, or whether the process will continue to produce as many diagnostic messages as possible. /NOFORCE 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 /NOFORCE.

Description

The DWS command is used to create a new project within a Dimensions product. The project can be empty, based on an existing project or based on an existing baseline.

NOTE The /PROJECT, /FILENAME and /POPULATE qualifiers are no longer applicable in DWS. In order to assign deployment areas to a project and populate them, use the RAWS command.

When DWS populates a project using the /WORKSET or /BASELINE qualifiers, it makes a shallow copy of the populating collection; that is, the project or baseline directory structure is copied and equivalent relationships are created to any child collections. The relative locations of child collections are preserved, but per-user collection roots are not copied.

When DWS populates the new project from a project, and the /COPY_CONFIGS qualifier is specified, build configurations and/or relationships will also be copied to the new project. See the Dimensions CM Build Tools User's Guide for further details.

The Remove Project (RWS) command is used to remove a project.

NOTE The /BRANCH, /TRUNK, /AUTO_REV, /NOAUTO_REV, and DEPLOYMENT_MODEL qualifiers may further be used to alter the options associated with the project. The permitted combinations of these qualifiers are:

```
DWS <project-spec> /BRANCH
DWS <project-spec> /TRUNK
DWS <project-spec> /AUTO_REV
DWS <project-spec> /NOAUTO_REV
DWS <project-spec> /BRANCH /AUTO_REV
DWS <project-spec> /BRANCH /NOAUTO_REV /DEPLOYMENT_MODEL=MANUAL
DWS <project-spec> /TRUNK /AUTO_REV
DWS <project-spec> /TRUNK /NOAUTO_REV
```

Constraints

Normally this command can be run only by a user with the appropriate management privileges for the project concerned.

This constraint can, however, be relaxed using the Set Project Permissions (SWSP) command, as described on page 451.

DWSD – Delete Project Directory

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<directory-path>
[/WORKSET=<project-spec>]
[/[NO]RECURSIVE]
[/[NO]REMOVE_ITEMS]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
```

Example

```
DWSD src
```

Parameters and qualifiers

- `<directory>`

  The DWSD command deletes a project-directory `<directory>` from the database project structure relative to the working location (this subdirectory previously having been used for project operations, but no longer being required).

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

By default, DWSD fails if the specified project directory contains subdirectories or items. The following two qualifiers enable you to remove such a directory from a project.

- `/RECURSIVE`

  Specifies that the DWSD command will be applied first to each subdirectory of the specified project directory. If deletion of any subdirectory fails (for example, if the subdirectory contains items and /REMOVE_ITEMS is not in effect), the overall command fails as well; otherwise, the DWSD command is applied last to the specified project directory itself.

- `/REMOVE_ITEMS`

  Specifies that the DWSD command will remove from the current project each revision of each item in the specified project directory before attempting to delete the project directory itself. If /RECURSIVE is specified also, DWSD will recursively remove item revisions from each subdirectory of the specified project directory as well.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which this structural change to the project is to be related `In Response To`.

  Specify this optional qualifier if you want this structural change to the project to be recorded against the specified request(s). If path control has been enabled, this qualifier is mandatory. If path control is not enabled, then the request(s) will be ignored.

NOTE Whenever a new revision is added to a project, its stage is reset to DEVELOPMENT, and associated deployment areas and library cache areas are updated.

Constraints

Normally this command can be run only by a user with the appropriate management privileges for the project concerned.

This constraint can, however, be relaxed using the Set Project Permissions (SWSP) command, as described on page 451.

ECDI – Extract (Check Out) Request Items

NOTE This command is not available for items that belong to a stream.

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/DIRECTORY=<project-directory-filter>]
[/[NO]RECURSIVE]
[/LOGFILE=<log-file>]
[/[NO]OVERWRITE]
[/[NO]TOUCH]
[/USER_DIRECTORY=<target-directory>
[/WORKSET=<project>]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Example

```
ECDI PAYROLL_CR1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  Specifies requests that are to be related to all extracted items.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items checked out.

- `/RECURSIVE`

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed.

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/OVERWRITE`

  Specifies that Dimensions should overwrite files on disk with files processed by this command.

- `/TOUCH`

  Assigns the current date and time to extracted files.

- `/USER_DIRECTORY`

  Specifies the directory to which files are to be checked out.

- `/WORKSET`

  Specifies the project to be processed by this command.

- `/NOMETADATA`

  This parameter disables creation and usage of metadata files in the local work area.

- `[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]`

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

  See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Description

This command checks out all the items that are related to a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

ECFG – Extract (Check Out) Build Configuration

```
ECFG <configuration-spec>
[/WORKSET=<project-spec>]
```

Example

```
ECFG component1
/WORKSET=component1
```

Parameters and qualifiers

- `<configuration-spec>`

  Specifies the name of the build configuration to be checked out.

- `/WORKSET=<project-spec>`

  Specifies the project that contains the build configuration to be checked out.

  Default: The user's current project.

Description

Checks out a build configuration to the current user ID.

EI – Extract (Check Out) Item for Update

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/BASELINE=<baseline-spec>]
[/USER_FILENAME=<user-filename>]
[/REVISION=<new-revision>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/WORKSET=<project-spec>]
[/[NO]OVERWRITE]
[/[NO]EXPAND]
[/CODEPAGE=<code-page>|DEFAULT|SOURCE]
[/TOUCH]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Examples

```
EI PROD:"QUERY RELEASE".AAAA-SRC;1 /CHANGE=PROD_DC_12
```

```
EI PROD:;1 /FILE=qr.c /CHANGE=PROD_DC_12
```

```
EI "FS:CBEVENT C.A-SRC;b1#4" -
/USER_FILENAME="e:\cpjtest\cbevent.c" -
/REVISION="b1#5" -
/OVERWRITE
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision in the project specified by `/WORKSET`. If `/WORKSET` is unspecified, the user's current project will be assumed.

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- /FILENAME=<file-name>

  Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

  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. The project file name for the same item may *differ* between projects; for example, src/hello.c, hello.c, or src/build/hello.c.

  It may be omitted if <item-id> is specified.

- /BASELINE=<baseline-spec>

  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.) <baseline-spec> comprises:

  <product-id>:<baseline-id>

  If omitted, the specified or default <revision> (as described above) is selected for checking out.

- /USER_FILENAME=<user-filename>

  Specifies the name of the file which will be created in the user-area, and into which the item will be copied.

  If omitted, it defaults to the project file name – i.e. the file created in the user area (the current directory) will have the same name as that of the item's project file name.

- /REVISION=<new-revision>

  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.

  If omitted, Dimensions 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.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

<attrN>	is 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.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

<requestN>	identifies a request to which the item revision is to be related **In Response To**.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  If specified, the new revision of the item will be placed in the project, otherwise it will be placed in the user's current project.

- /[NO]OVERWRITE

  When checking out an item revision, specify whether or not Dimensions is allowed to perform this operation depending on:

  - The existence of a local file of the same name.
  - The status (read-only or writeable) of an existing local file of the same name.

  /NOOVERWRITE – which is normally the default but which can be reassigned using the SET OVERWRITE command described on page 428 – results in a file only being successfully checked out by Dimensions if the local (target) file does not already exist or is marked read-only. This is the traditional Dimensions 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).

  /OVERWRITE results in the file being successfully checked out by Dimensions irrespective of the existence or writeable status of any local (target) file.

  To clarify the above, consider the following example:

```
EI "FS:CBEVENT C.A-SRC;b1#4" -
       /USER_FILENAME="e:\cpjtest\cbevent.c" -
       /REVISION="b1#5" /OVERWRITE
```

  would always overwrite cbevent.c if it existed, irrespective of whether it was marked read-only or not.

- /[NO]EXPAND

  Specifies whether to expand item header substitution variables.

  The default is /NOEXPAND.

  /EXPAND expands item header substitution variables provided the item type was defined in the process model with **Enable item header substitution**.

- /CODEPAGE=<code-page>|DEFAULT|SOURCE

  Specifies the *code page* to be associated with the item. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
  /CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

  For further details concerning code pages and logical nodes see the *Network User's Guide*.

  You are advised to let the parameter default to the code page for the item type or the platform.

The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.
SOURCE	Use whatever code page was associated with the item during check in. This option is relevant only on commands that take an item out of a library i.e. FI and EI.

- /TOUCH

  Sets the modification time of the user file to the current system time instead of the modification time stored in Dimensions for this item revision.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Constraints

This command can be run by users who have one of the roles required to action the item from the initial lifecycle state to a new state.

By default, users can check out different revisions of an item in parallel unless a user with the role of PRODUCT-MANAGER has disabled this facility to allow only one revision of an item to be checked out at any given time.

ESJ – Edit Schedule Job

```
<job-id>
[/START_TIME]
[/REPEAT]
[/JOB_STATUS]
[/JOB_DESC]
```

Example

```
ESJ "MyJobName" /JOB_STATUS="ACTIVE"
```

Parameters and qualifiers

- `<job-id>`

  Specifies the job name.

- `/START_TIME`

  Specifies the job starting time in the format 'DD-MM-YYYY HH24:MI:SS'.

- `/REPEAT`

  Specifies an optional repeat time, can be one of:

  - 0, 10, 20, 30, 40, 50, 60, MINUTES

    For example, /REPEAT="40 MINUTES"

  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, HOURS

    For example, /REPEAT="7 HOURS"

  - 0, 1, 2, 3, 4, 5, 6, 7, DAYS

    For example, /REPEAT="1 DAYS"

- `/JOB_STATUS`

  Specifies the initial job status: INACTIVE or ACTIVE

- `/JOB_DESC`

  Describes the schedule job.

Description

Enables you to edit a scheduled job.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

EXIT – End Dimensions Execution

```
[NO PARAMETERS]
```

NOTE EXIT cannot be run from Dimensions for z/OS.

EXIT may be used (but is not essential) to mark the end of a command file which is specified to CMD.

Constraints

None

EXPORT – Export Build Configuration

```
/USER_FILENAME=<filename>
[/WORKSET=<project-spec>]
[/CONFIGS=(<build-config1>, <build-config2>, ...)]
[/CONFIGS_MASK=config-wild-specification]
[/CONFIG_TYPES=(<config-type1>, <config-type2>, ...)]
[/TOOLS=(<tool-spec1>, <tool-spec2>, ...)]
[/TOOLS_MASK=tool-wild-specification]
[/OPT_GROUPS=(<opt-group-spec1>, <opt-group-spec2>, ...)]
[/OPT_GROUPS_MASK=<opt-group-wild-mask>]
[/RULE_TEMPLATES=[(<rule-template-spec1>, <rule-template-spec2>, ...)]
[/RULE_TEMPLATES_MASK= <rule-wild-spec>]
[/APP_TEMPLATES =(<app-template1>, <app-template2>, ...)]
[/APP_TEMPLATES_MASK=<app-template-wild-specification>]
[/[NO]WORK_AREAS]
[/[NO]DEPLOYMENT_AREAS]
```

Example

```
EXPORT
/USER_FILENAME=build\build_configs\component1.xml
/WORKSET=component1
/CONFIGS=component1
```

Parameters and qualifiers

- `/USER_FILENAME`

  Specifies the name and location of the XML file to be exported. Can be in the following formats:

  - `<nodename>::<filename>`
  - `<area>::<filepath>`
  - `<filepath>`
  - `<file-spec>`

- `/WORKSET=<project-spec>`

  Specifies the name of the parent project/stream containing the build configuration to be exported.

  Default: The user's current project.

- `/CONFIGS=(<build-config1>, <build-config2>, ...)`

  Specifies the name(s) of one or more build configurations to be exported.

  If not specified, all build configurations for the project are included.

- `/CONFIGS_MASK=<config-wild-specification>`

  Specifies a wildcard expression that filters the build configurations to be exported.

  This can include "_" for a single character and "*" or "%" for one or more characters.

  If the /CONFIGS parameter has been specified, this filter will be applied to the specified list of build configurations.

- `/CONFIG_TYPES=(<config-type1>, <config-type2>, ...)`

  Specifies the name(s) of one or more build configurations types to be included.

- `/TOOLS=(<tool-spec1>, <tool-spec2>, ...)`

Specifies the name(s) of one or more build tool specifications to be included.

- `/TOOLS_MASK=<tool-wild-specification>`

  Specifies a wildcard expression that filters the build tools to be included. This can include "_" to represent a single character and "*" or "%" to represent one or more characters.

  If the /TOOLS parameter has been specified, this filter will be applied to the specified list of build tool specifications.

- `/OPT_GROUPS=(<opt-group-spec1>, <opt-group-spec2>, ...)`

  Specifies the name(s) of one or more build option groups to be included.

- `/OPT_GROUPS_MASK=<opt-group-wild-specification>`

  Specifies a wildcard expression that filters the option groups to be included. This can include "_" to represent a single character and "*" or "%" to represent one or more characters.

  If the /OPT-GROUPS parameter has been specified, this filter will be applied to the specified list of option groups.

- `/RULE_TEMPLATES=(<rule-template-spec1>, <rule-template-spec2>, ...)`

  Specifies the name(s) of one or more transition rule templates to be included.

- `/RULE_TEMPLATES_MASK=<rule-wild-specification>`

  Specifies a wildcard expression that filters the transition rule templates to be included. This can include "_" to represent a single character and "*" or "%" to represent one or more characters.

  If the `/RULE_TEMPLATES` parameter has been specified, this filter will be applied to the specified list of transition rule templates.

- `/APP_TEMPLATES=(<app-template1>, <app-template2>, ...)`

  Specifies the name(s) of one or more application rule templates to be included.

- `/APP_TEMPLATES_MASK=<app-template-wild-specification>`

  Specifies a wildcard expression that filters the application rule templates to be included. This can include "_" to represent a single character and "*" or "%" to represent one or more characters.

  If the `/APP_TEMPLATES` parameter has been specified, this filter will be applied to the specified list of application rule templates.

- `[/[NO]WORK_AREAS]`

  This option specifies whether or not work areas are to be included.

- `[/[NO]DEPLOYMENT_AREAS]`

  This option specifies whether or not deployment areas are to be included.

Description

This command enables you to export build configuration information from Dimensions Build to an XML-formatted file. For more information about using Dimensions Build see the *Dimensions CM Build Tools User's Guide.*

For each form where there is /xxx= and an /xxx_MASK= qualifier, if both are specified, the result is the union of the two specifications. The wild card masks can include "_" to represent a single character and "*" or "%" to represent one or more characters.

As a result of processing an EXPORT, it is possible that errors, warnings, or informational messages may be issued. The overall severity of the whole call is set to that of the highest severity message.

FBI – Fetch (Get) Baseline Items

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
[/DIRECTORY=<directory>]
[/USER_DIRECTORY=<user-directory>]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/[NO]OVERWRITE]
[/CODEPAGE<code-page>|DEFAULT|SOURCE]
[/[NO]TOUCH]
[/[NO]RECURSIVE]
[/METADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Examples: The following command fetches the specified baseline to the specified remote directory on a tertiary node. Each file is touched as it is fetched.

```
FBI "PVCS:DM9000 BUILD 5.3" /TOUCH /USER_DIRECTORY="stal-dev-lx1::/
    build/Sources/Dm9000"
```

The following command fetches all items from the `build` directory in the specified baseline.

```
FBI PVCS:DM9000 /DIRECTORY="build" /
    USER_DIRECTORY="E:\Dimensions\dim90-build.2004"
```

Parameters and qualifiers

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items fetched.

- `/USER_DIRECTORY`

  Specifies the directory to which files are to be fetched.

- `/[NO]EXPAND`

  Specifies whether to expand item header substitution variables.

  The default is /NOEXPAND

  /EXPAND expands item header substitution variables provided the item type was defined in the process model with **Enable item header substitution**.

- `/[NO]REEXPAND`

  Specifies whether item substitution header variables will be re-expanded for existing files in the work area even if the item files have not changed in the repository.

  Default: `/NOREEXPAND`

- `/[NO]OVERWRITE`

  Specifies that Dimensions should overwrite files on disk with files processed by this command.

- `/CODEPAGE=<code-page>|DEFAULT|SOURCE`

Specifies the *code page* to be associated with the items. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
/CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

For further details concerning code pages and logical nodes see the *Network User's Guide*.

You are advised to let the parameter default to the code page for the item type or the platform.

The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.
SOURCE	Use whatever code page was associated with the item during check in. This option is relevant only on commands that take an item out of a library i.e. FI and EI.

- /[NO]TOUCH

  Assigns the current date and time to fetched files. Default /TOUCH.

- /[NO]RECURSIVE

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed. Default /RECURSIVE.

- /[NO]METADATA

  This parameter specifies creation and usage of metadata files in the local work area.

  Default: /METADATA

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.

UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Description

The FBI command fetches (gets) all item revisions from a baseline or a baseline directory, regardless of the current project.

Constraints

For each item that FBI fetches (gets) in the specified baseline, FBI executes the FI command. The constraints for that command are as follows: "This command can be run by users who have a role on the design part owning the item or a role on one of the ancestor nodes of that design part."

FCDI – Fetch (Get) Request Items

NOTE This command is not available for items that belong to a stream.

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[/DIRECTORY=<project-directory-filter>]
[/[NO]RECURSIVE]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/LOGFILE=<log-file>]
[/[NO]OVERWRITE]
[/[NO]TOUCH]
[/USER_DIRECTORY=<target-directory>
[/WORKSET=<project>]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Example

```
FCDI PAYROLL_CR1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items fetched.

- `/RECURSIVE`

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed.

- `/[NO]EXPAND`

  Specifies whether item header substitution will take place.

  The default is /NOEXPAND

  /EXPAND expands header substitution variables provided the `item type` was defined in the process model with **Enable item header substitution**.

- `/[NO]REEXPAND`

  Specifies whether item substitution header variables will be re-expanded for existing files in the work area even if the item files have not changed in the repository.

  Default: `/NOREEXPAND`

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/OVERWRITE`

Specifies that Dimensions should overwrite files on disk with files processed by this command.

- `/TOUCH`

  Assigns the current date and time to fetched files.

- `/USER_DIRECTORY`

  Specifies the directory to which files are to be fetched.

- `/WORKSET`

  Specifies the project to be processed by this command.

- `/NOMETADATA`

  This parameter disables creation and usage of metadata files in the local work area.

- `[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]`

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
`SHOW`	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Description

This command fetches all the items that are related to a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

FI – Fetch (Get) Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for items that belong to a stream.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/BASELINE=<baseline-spec>]
[/USER_FILENAME=<user-filename>]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/WORKSET=<project-spec>]
[/[NO]OVERWRITE]
[/CODEPAGE=<code-page>|DEFAULT|SOURCE]
[/[NO]TOUCH]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Examples

```
FI PROD:"QUERY RELEASE".AAAA-SRC;1
FI "FS:CBEVENT C.A-SRC;b1#4" -
/USER_FILENAME="e:\cpjtest\cbevent.c"
/NOEXPAND -
/NOOVERWRITE
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see About the Command-Line Interface on page 12).

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

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 gotten from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

It may be omitted if `<item-id>` is specified.

- `/BASELINE=<baseline-spec>`

  Specifies a release-baseline which contains the particular revision of `<item-spec>` to be gotten. It comprises:

  ```
  <product-id>:<baseline-id>
  ```

  If omitted, the specified or default `<revision>` (as described above in `<item-spec>`) is gotten.

- `/USER_FILENAME=<user-filename>`

  Specifies the name of the file which will be created in the user area, and into which the item will be copied.

  If the user file name is omitted, it will default (with the exception described below) to the project file name – i.e. the file created in the user area (the current directory) will have the same name as that of the item's project file name.

NOTE If the FI command is submitted via the Dimensions desktop client command-line interfaces, the `/USER_FILENAME` qualifier is **compulsory**.

- `/[NO]EXPAND`

  When getting the item, request Dimensions **not** to expand any substitution variables (file heading text and/or Dimensions-defined or user-defined attributes) located in the item header. This is analogous to what happens when an item is *checked out*.

  Refer to "Item Format Templates", in the *Process Configuration Guide* for a discussion of item header substitution.

  The default is `/NOEXPAND`

- `/[NO]REEXPAND`

  Specifies whether item substitution header variables will be re-expanded for existing files in the work area even if the item files have not changed in the repository.

  Default: `/NOREEXPAND`

- `/WORKSET=<project-spec>` comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

- `/[NO]OVERWRITE`

  When getting an item revision, specify whether or not Dimensions is allowed to perform this operation depending on:

  - The existence of a local file of the same name.

- The status (read-only or writeable) of an existing local file of the same name.

/NOOVERWRITE – which is normally the default but which can be reassigned using the SET OVERWRITE command described on page 428 – results in a file only being successfully gotten by Dimensions if the local (target) file does not already exist or is marked read-only. This is the traditional Dimensions 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).

/OVERWRITE results in the file being successfully gotten by Dimensions irrespective of the existence or writeable status of any local (target) file.

NOTE If the file on disk is the same as the item revision stored in Dimensions CM, the file will not be ovewritten regardless of this option being set.

To clarify the above, consider the following example:

```
FI "FS:CBEVENT C.A-SRC;b1#4" -
       /USER_FILENAME="e:\cpjtest\cbevent.c" /NOEXPAND -
       /NOOVERWRITE
```

That would not allow cbevent.c to be overwritten if it existed and was not marked read-only.

- /CODEPAGE=<code-page>|DEFAULT|SOURCE

Specifies the *code page* to be associated with the item. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
/CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

For further details concerning code pages and logical nodes see the *Network User's Guide*.

You are advised to let the parameter default to the code page for the item type or the platform.

The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.

DEFAULT	Use the code page specified for the target node connection.
SOURCE	Use whatever code page was associated with the item during check in. This option is relevant only on commands that take an item out of a library i.e. FI and EI.

- /TOUCH

  Sets the modification time of the user file to the current system time instead of the modification time stored in Dimensions for this item revision.

NOTE If the file on disk is the same as the item revision stored in Dimensions CM, the modification time will not be updated.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

File Permissions

The FI command applies the read-only attribute to the file unless the DM_KEEP_PERMS Y configuration variable is defined. However, if there is no content change and therefore no "get" operation actually occurs, the file permissions remain unchanged.

Constraints

This command can be run by users who have a role on the design part owning the item or a role on one of the ancestor nodes of that design part.

FIF – Find Item File

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/WORKSET=<project-spec>]
```

Example

```
FIF PROD:"QUERY RELEASE".AAAA-SRC;2
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

  `<item-id>` may be omitted if `<file-name>` is specified.

  `<variant>` may be omitted if only one exists.

  `<revision>` may be omitted if the file version is not required.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/WORKSET=<project-spec>`

  Comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Description

This program displays an item's file name, including full directory path and optionally the revision number.

Non-Dimensions operations (e.g. compilation) can sometimes be performed by reading the item directly from the item library (provided that the user has been granted normal operating system read access to the library).

Constraints

This command can be run only by users who have a role for the owner part.

The command is not available for items held in delta libraries.

FRC – Forward a Release to a Customer

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<release-spec>
/CUSTOMER=<name>
/LOCATION=<location>
/PROJECT=<project-spec>
[/DESCRIPTION=<description>]
```

Example

```
FRC PROD:"R M 2.0 FOR HP"
/CUSTOMER="Brown Finances -
/LOCATION="Bristol"
/PROJECT="PAYROLL"
```

Parameters and qualifiers

- `<release-spec>`

  Specifies the releases-spec, which comprises:

  ```
  <product-id.:<release-id>
  ```

- `/CUSTOMER=<name>`

  Specifies the customer's name.

- `/LOCATION=<location>`

  Specifies the customer's physical location.

- `/PROJECT=<project-spec>`

  Specifies the project name.

- `/DESCRIPTION=<description>`

  Optionally associate a description of the release.

Description

The Dimensions product allows you to maintain a list of customers and a record of which Dimensions releases have been sent to each customer.

The FRC command enables you to record the fact that a release has been supplied to a specific customer.

Constraints

Only users with the appropriate management privileges can run this command.

The combination of customer name, location, and project-spec must be unique in the Dimensions database.

You cannot forward the same release to a customer twice.

FWI – Fetch (Get) Project Items

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
[/DIRECTORY=<directory>]
[/USER_DIRECTORY=<user-directory>]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/[NO]OVERWRITE]
[/CODEPAGE=<code-page>|DEFAULT|SOURCE]
[/[NO]TOUCH]
[/[NO]RECURSIVE]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

Examples: The following command recursively fetches (gets) the entire specified project into the user's project directory. Each file is touched as it is fetched.

```
FWI PVCS:DM9000 /TOUCH
```

The following command recursively fetches (gets) all items from the `build` directory into the specified user directory.

```
FWI PVCS:DM9000 /DIRECTORY="build"
    /USER_DIRECTORY="E:\Dimensions\dim90-build.2004"
```

The following command fetches (gets) only items in the `install` directory, excluding any subdirectories, into the specified remote directory on a tertiary node.

```
FWI PVCS:DM9000 /DIRECTORY="install" /NORECURSE /USER_DIRECTORY="stal-
    dev-lx1::/builds/dim90-build.2004"
```

Parameters and qualifiers

- `<project-spec>`

  Comprises `<product id>:<project id>` and specifies the project or stream whose items are to be fetched.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items fetched.

- `/USER_DIRECTORY`

  Specifies the directory to which files are to be fetched.

- `/[NO]EXPAND`

  Specifies whether item header expansion will occur.

  The default is to expand header substitution variables provided the item type was defined in the process model with **Enable item header substitution**.

- `/[NO]REEXPAND`

  Specifies whether item substitution header variables will be re-expanded for existing files in the work area even if the item files have not changed in the repository.

  Default: `/NOREEXPAND`

- /[NO]OVERWRITE

  Specifies that Dimensions should overwrite files on disk with files processed by this command.

- /CODEPAGE=<code-page>|DEFAULT|SOURCE

  Specifies the *code page* to be associated with the items. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
  /CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

  For further details concerning code pages and logical nodes see the *Network User's Guide*.

  You are advised to let the parameter default to the code page for the item type or the platform.

  The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.
SOURCE	Use whatever code page was associated with the item during check in. This option is relevant only on commands that take an item out of a library i.e. FI and EI.

- /[NO]TOUCH

  Assigns the current date and time to fetched files. Default /TOUCH.

- /[NO]RECURSIVE

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed. Default /RECURSIVE.

- /[NO]METADATA

  This parameter specifies creation and usage of metadata files in the local work area.

  Default: /METADATA

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED]

Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Description

The FWI command fetches (gets) the latest item revisions from an entire project or a project directory. By default, FWI traverses all subdirectories when looking for item revisions to fetch.

Constraints

For each item that FWI fetches (gets) in the specified project, FWI executes the FI command. The constraints for that command are as follows: "This command can be run by users who have a role on the design part owning the item or a role on one of the ancestor nodes of that design part."

GENCERT - Generate Certificates

```
[/COUNT=n]
[/NETWORK_NODE=<nodename>]
[/USER=<userid>]
[/PASSWORD=<password>]
```

Description

Generates one-time certificates that allow the currently logged in user to reconnect using one of the following programs:

- `dmcli -cert <cert>`
- `template_test -v <cert>`

  For more details about the template testing program see the *Developer's Reference*.

Example

```
GENCERT
   /COUNT=2
   /NETWORK_NODE=myserver
   /USER=user1
   /PASSWORD=abcde123
```

Parameters and Qualifiers

- `/COUNT=n`

  Specifies the number of certificates to return.

- `/NETWORK_NODE=<nodename>`

  Specifies a network node.

- `/USER=<userid>`

  Specifies a user ID or credential set for the specified node. For information about credential sets see the *System Administration Guide*.

- `/PASSWORD=<password>`

  Specifies a password for the user ID. Not required if you specify a credential set in `/USER`.

NOTE

- To authenticate to a tertiary node specify these qualifiers: `/NETWORK_NODE, /PASSWORD, /USER`
- If Structured Information Return (SIR) is enabled the GENCERT command returns the certificates as symbol table variables. Use this mechanism in conjunction with the `)COPYSYM` directive in the templating mechanism to provide one or more steps of a build with certificates. The SSPM command (see page 442) controls structured information return processing. For full details about SIR see the *Developer's Reference*.

GREP – Search and Replace

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
/SEARCH=<text>
[/TEXT_REPLACE=<text>]
[/FILENAME=<text>]
[/COMMENT=<text>]
[/USER_FILENAME=<text>]
[/[NO]IGNORE_CASE]
[/LATEST_REV]
[/[NO]RECURSIVE]
[/PRODUCT=<text>]
[/TYPE_LIST=(type,type)]
[/FORMAT_LIST=(format,format)]
[/WORKSET=<project-spec>]
```

One of the following must also be specified...

```
/WS_DIR=<directory>
/PART=<part specification>
/CHANGE_DOC_ID=<doc id>
```

Examples

```
GREP  /SEARCH="printf"  /PRODUCT="PAYROLL"
    /FILENAME="%.c" /WS_DIR="src\gui"  /LATEST_REV
    /IGNORE_CASE
```

will cause Dimensions to search for all occurrences of `printf` ignoring case, in all `c` files in the project directory `src\gui` owned by product **PAYROLL** looking only at the latest revision.

```
GREP  /SEARCH="printf" /FILENAME="%c,%.h,%.cpp"
    /TYPE_LIST=(DAT) /PART="PETRAY:PETRAY.A;1"
    /RECURSIVE
```

will find all items which are `CPP`, `C` or `H` files of item type `DAT` which contain the word `printf` (the case must match) owned by the design part PETRAY or any of its children (recursively).

```
GREP /SEARCH="strncasecmp" /TEXT_REPLACE="strnicmp"
    /FILENAME="domain%.c" CHANGE_DOC_ID="PAYROLL_CR_1"
    /COMMENT="replace string routines"
    /USER_FILE="C:\output.log"
```

will find all items containing `strncasecmp` and create new revisions of the matching items replacing occurrences of `strncasecmp` with `strnicmp`. The comment `"replace string routines"` will be used as the comment for the creation of the new item revisions. Only items related to request `PAYROLL_CR_1` will be searched and the output of the command will be placed in the log file `"C:\output.log"`.

Parameters and qualifiers

- `/SEARCH=<text>`

  Specifies the text to find by defining search patterns using regular expressions. For example, if you set this to /SEARCH=/.product, then the search will return only instances of the exact phrase ".product", including the period. If you set this to /SEARCH=.product, the period connotes a search for every instance of the term

"product", with or without a period. Please consult rules for using regular expressions before defining this parameter to ensure that you achieve the correct results.

- /TEXT_REPLACE=<text>

  Specifies the string with which to replace found values.

- /FILENAME=<text>

  Specifies a filter for matching files.

- /COMMENT=<text>

  Specifies comment used for newly created revisions only valid if REPLACE is specified.

- /USER_FILE=<text>

  Specifies optional file to contain output of GREP command.

- /IGNORE_CASE

  ignore case when searching.

- /LATEST_REV

  look at only the latest revision.

- /RECURSIVE

  causes search to be recursive from the given object.

- /PRODUCT=<text>

  only look at items of this product.

- /TYPE_LIST=(type,type)

  only look at items of these types.

- /FORMAT_LIST=(format,format)

  only look at items of these formats.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  This specifies the project to be used for this command: failing this, the user's current project will be taken. The WS_DIR qualifier can be used to further scope the nature of the search.

- /WS_DIR=<directory>

  Specifies the project directory to search

NOTE To search the top directory in a project use only the forward slash character (/).

- /PART=<part specification>

  Specifies the design part to search

- /CHANGE_DOC_ID=<doc id>

  Specifies the request to search from.

All item revisions that are related, either as **Affected** or **In Response To**, will be considered for this command.

Constraints

This command can be run only by users who have the role to get the relevant items.

The GREP command is only for use with files of an ASCII text format. Binary files such as Microsoft Word documents are not searched, they are simply ignored.

HELP – Help

```
<command-name>
timezones
```

Example

```
HELP download
```

Parameters and qualifiers

- `<command-name>`

  The command for which you want a usage summary.

- `timezones`

  Displays all timezone values available to use with the SET TIMEZONE command. See "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

  You can optionally limit the list of timezones using a search parameter, in the following format:

  ```
  HELP timezones=*<wildcard>
  ```

  For example, to only return timezones with Americas in their names:

  ```
  HELP timezones=*Americas
  ```

Description

Displays a usage summary, including required and optional qualifiers, for the specified command.

HIDE - Hide Unused Streams and Projects

```
[/STREAM=PROD:STREAM_ID]
[/PROJECT=PROD:PROJ_ID]
[/USER_WORKSETLIST]
[/NOLOCK]
```

Description

Hides unused streams and projects but does not delete them. Hidden streams and projects are not displayed in clients and at the command line but you can list them using the `LWS /ALL` command. By default HIDE locks all streams and projects after they are hidden.

Use the `SHOW` command to make hidden streams and projects visible, see page 436.

Example

```
HIDE /STREAM=QLARIUS:MAINLINE_JAVA
```

Hides a stream with the specification `QLARIUS:MAINLINE_JAVA`.

Parameters and Qualifiers

- /STREAM

  The specification of a stream to hide.

- /PROJECT

  The specification of a project to hide.

- /USER_WORKSETLIST

  The path of a local file that lists multiple streams and projects to hide (specify each on a new line).

- /NOLOCK

  Does not lock streams and projects after they are hidden.

IMPORT – Import Build Configuration

```
<build-configuration-spec>
[/NEW_NAME=<configuration-spec>]
/USER_FILENAME=<file-name>
[/WORKSET=<project-spec>]
[/[NO]FORCE
```

Example

```
IMPORT component1
/NEW_CONFIGURATION=component2
/USER_FILENAME=stal-dev::C:\data\builds\build_configs\component1.xml
/WORKSET=component02
/FORCE
```

Parameters and qualifiers

- `<build-configuration-spec>`

  Specifies the name of the build configuration to be imported.

- `/NEW_NAME=<configuration-spec>`

  Specifies the name of the build configuration after it is imported.

  Default: The existing build configuration name.

- `/USER_FILENAME=<file-name>`

  Specifies the name of the XML file to be imported. Can be in the following formats:

  - `<node>::<filename>`
  - `<area>::filename`.

- `/WORKSET=<project-spec>`

  Specifies the parent project to be related the imported build configuration.

  Default: The user's current project.

- `/[NO]FORCE`

  Performs an import and ignores all warnings.

  Default: /NOFORCE (stops importing if any warnings are received).

Description

Enables you to import a build configuration in an XML-formatted file into Dimensions Build. For more information about using Dimensions Build see *Dimensions CM Build Tools User's Guide.*

LA – List Areas

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
[/TYPE=<area-type>]
[/WORKSET=<project-spec> [/STAGE=<stage>]]
[/NETWORK_NODE=<machine-name>]
[/OWNER=<user-or-group>]
[/STATUS=ONLINE or OFFLINE]
[/[NO]DETAIL]
```

Example

```
LA <area-name> /TYPE=DEPLOYMENT
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area of which to list details. If this parameter is omitted, the list of areas is potentially filtered by other optional qualifiers.

- `/TYPE=<area-type>`

  WORK or DEPLOYMENT. If this parameter is specified, only areas of this type will be included in the list.

- `/WORKSET=<project-spec>`

  If this parameter is specified, only areas related to this project will be included in the list.

- `/STAGE=<stage>`

  If this parameter is specified, only areas associated with this stage will be included in the list.

- `/NETWORK_NODE=<machine-name>`

  If this parameter is specified, only areas defined for this node will be included in the list.

- `/OWNER=<user-or-group>`

  If this parameter is specified, only areas owned by this user or group will be included in the list.

- `/STATUS=ONLINE or OFFLINE`

  If this parameter is specified, only areas of the specified status will be included in the list.

- `/[NO]DETAIL`

  Specifying /NODETAIL means that only a summary list is reported.

  The default is /DETAIL.

Description

The LA command lists details of the specified area or all areas matching the specified criteria. If no parameters are provided, the command lists details of all areas.

Constraints

Only users with the Run Admin Reports privilege can run this command.

LAST - List Area Structure

```
<areaname[;revrange]>
[/VERBOSE]
[/DETAIL]
[/USER_FILENAME="file.txt"]
[/INCLUDE_PATH_MASK_LIST=(mask1, mask2...]
[/EXCLUDE_PATH_MASK_LIST=(mask1, mask2...]
```

Description

Lists all items and directories in a specific version of an area.

Parameters and Qualifiers

- `<areaname[;revrange]>`

  Specifies an area name where `[;revrange]`:

  - If not specified, or specified as *, lists the tip version of the area.
  - If specified as a single number lists just that version of the area.
  - If specified in the form `mm..nn` lists all versions between versions `mm` and `nn` inclusive. If `mm` is not present `nn` starts at version 1. If `nn` is not present `mm` lists all versions from `mm` to the tip version of the area.

- `/VERBOSE`

  Uids and internal details of the processing are displayed.

- `/DETAIL`

  Additional information about each revision is displayed.

- `/USER_FILENAME="file.txt"`

  The command output is directed to the specified file. The file format is CSV, which can be read by a spreadsheet or imported into a database.

- `/INCLUDE_PATH_MASK_LIST and /EXCLUDE_PATH_MASK_LIST`

  Specify lists of Ant patterns to include and exclude.

  - ** matches directory paths
  - ? matches a single letter

  If you only use one pattern you can omit the parenthesis ( ), for example: `/INCLUDE_PATH_MASK_LIST=mask1`

LAVC – List Deployment Area Versions

```
<area_name>[;<version>]
[/WORKSET=<projectName>]
[ALL]
[/VERBOSE]
```

Example

```
LAVC UT;* /VERBOSE /ALL
```

Output:

```
Deployment operations performed:-
         A       templates/workman.template;1
         A       exec
         A       prt
         A       prt/gra.c;1
         A       prt/graphic.h;1
         A       prt/prt.c;1
         A       prt/prt.hlp;1
         A       prt/print.h;1
         A       prt/makefile.mk;1
```

Parameters and qualifiers

- `<area_name>[;<version>]`

  Specifies the name of a deployment area and optionally an area version. If you do not specify a version, a summary of all selected area versions is listed. If you specify an asterisk '*' as the area version, the details of all selected deployment operations that participated in each version are listed.

  If you specify a deployment area version the details of all the changes in the version are listed:

  - !: item renamed
  - A: item added
  - U: item updated
  - D: item deleted
  - u: item previously updated (requires `/VERBOSE`)
  - a: item previously added (requires `/VERBOSE`)
  - d: directory does not contains items associated with this project/stream but does contain items associated with other projects/streams (and cannot be deleted).

- `/WORKSET=<projectName>`
  - If specified: only lists deployments for the specified project or stream.
  - If not specified: lists deployments for the current project or stream.
- `/ALL`

  Lists deployments for all projects and streams.

- /VERBOSE
  - If specified: also lists the statuses of items previously updated or added ('u' and 'a' conditions).
  - If not specified: only lists the number of references found for each item previously added and directories containing items associated with other projects/streams ('a' and 'd' conditions).

Description

Lists deployment area versions and their content.

LBA – List Build Areas

NOTE This command is no longer available; use LA (List Areas) instead.

See the LA command.

LBDB – List Existing Base Database Entries

No parameters.

Example `LBDB`

This command enables you to list existing registered base databases in an installation's network administration tables. See the *System Administration Guide* for details.

LBPROJ – List Dimensions Build Projects

NOTE This command is no longer available; use LWS instead.

LCK – Lock Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
WORKSET <project-spec>
```

Example

```
LCK WORKSET PROD_X:TEST_WS
```

Parameters and qualifiers

`<project-spec>` is comprised of:

```
<product-id>:<project-id>
```

The specified project must exist.

Description

This command locks the project/stream, with project as a fixed parameter and `<project-spec>` a user-defined parameter. The locked state prevents the addition of new Dimensions items to the project/stream or the removal of existing item revisions that are in a locked project/stream (baselining is likely to occur in this state). In addition, items that exist in a locked project/stream will not be actionable or updateable from another project/stream unless the user has the role of `PRODUCT-MANAGER` for the item's product. Users with the `PRODUCT-MANAGER` role can create new items in a locked project.

Constraints

This command can be run only by a user with the appropriate management privileges for the project concerned.

LCO – List Existing Contacts

No parameters.

Example `LCO`

This command enables you to list an installation's contacts. See the *System Administration Guide* for details.

LCS – List Credential Set

[/USER_FILENAME=<filespec>]

Optional parameter that writes the output to a CSV (comma-separated values) file.

This command enables you to list credential set for the current user. You can also use SIR to retrieve values from the command (see SSPM in the *Developer's Reference*).

LCST – List Existing Codesets

No parameters.

Example `LCST`

This command enables you to list an installation's codesets. See the *System Administration Guide* for details.

LFS – List Existing File Systems

No parameters.

Example `LFS`

This command enables you to list an installation's file systems. See the *System Administration Guide* for details.

LGRP – List Groups

```
[/[NO]DETAIL]
```

Description

This command lists all available groups and, if /DETAIL is specified, group members.

Constraints

Only users with the appropriate management privileges can run this command.

LII – List Item Build Relationships

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=workset filename]
[/USER_FILENAME="user filename"]
[/RELATIONSHIP = BLD_ACTUAL | BLD_PREDICTED | BLD_ALL]
```

Example

```
LII ACCTS: /FILENAME="MAPSET/ACCTSET.MAPSET"
    -/USER_FILENAME="./acctset.csv"
```

Parameters and qualifiers

- `<item-spec>`

  which has the form:

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

  Can be specified in full (`/FILENAME` is then not required) or partially specified (use `/FILENAME` to complete the remaining values).

- `/FILENAME=workset filename`

  Use this qualifier instead of the full item specification to identify a particular file revision. When working with mainframe file names, use the distributed `(LIBRARY/MEMBER.LIBRARY)` format rather than the `LIBRARY(MEMBER)` format.

- `/USER_FILENAME="user filename"`

  Specifies an optional .csv file that contains the item relationships.

- `/RELATIONSHIP = BLD_ACTUAL | BLD_PREDICTED | BLD_ALL`

  Specifies the type of build relationship to be listed.

BLD_ACTUAL	Lists relationships between inputs and outputs observed during the build processing.
BLD_PREDICTED	Lists relationships that exist between a previous revision of this item and the related item (for example, the user has revised a source file).
BLD_ALL	(Default) Lists both actual and predicted relationships.

Description

This command lists the build relationships that exist from/to the specified item revision. It provides a similar function to looking at the "derived items" view of an item in the GUI clients, except that this command can also display the BLD_PREDICTED records, which are normally hidden. If SIR processing is enabled, the LII command returns values as tables in the symbol table.

Constraints

None.

LINS – List Existing Database Instance Entries

No parameters.

Example `LINS`

This command enables you to list existing registered database instances in an installation's network administration tables. See the *System Administration Guide* for details.

LLCA – List Library Cache Areas

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
[/NETWORK_NODE=<machine-name>]
[/OWNER=<user-or-group>]
[/STATUS=ONLINE or OFFLINE]
```

Example

```
LLCA <area-name>
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area of which to list details. If this parameter is omitted, the list of areas is potentially filtered by other optional qualifiers.

- `/NETWORK_NODE=<machine-name>`

  If this parameter is specified, only areas defined for this node will be included in the list.

- `/OWNER=<user-or-group>`

  If this parameter is specified, only areas owned by this user or group will be included in the list.

- `/STATUS=ONLINE or OFFLINE`

  If this parameter is specified, only areas of the specified status will be included in the list.

Description

The LLCA command lists details of the specified library cache area or all library cache areas matching the specified criteria. If no parameters are provided, the command lists details of all library cache areas.

Constraints

Only users with the Run Admin Reports privilege can run this command.

LMNR – List Mail Notification Rules

No parameters.

Description

This command lists all mail notification rules.

LNC – List Existing Network Node Connections

No parameters.

Example `LNC`

This command enables you to list an installation's existing network node connections. See the *System Administration Guide* for details.

LNDO – List Existing Network Node Objects

No parameters.

This command enables you to list existing network nodes. See the *System Administration Guide* for details.

LNN – List Existing Network Nodes

No parameters.

Example `LNN`

This command enables you to list an installation's existing network nodes. See the *System Administration Guide* for details.

LNWO – List Existing Network Objects

No parameters.

Example `LNWO`

This command enables you to list an installation's existing network objects. See the *System Administration Guide* for details.

LOG - Lists Stream or Project Changeset History

```
[<versionRange>]
[/FROM_TIME=<from timestamp>]
[/TO_TIME=<to timestamp>]
[/WORKSET=<workset-spec>]
[/BRIEF]
[/COMMENT]
[/FILENAME]
[/LOGFILE]
[/CHANGE_DOC_ID]
```

Description

Enables you to list the changeset history of your streams and projects. You can use the qualifiers and parameters described below to filter the changesets that are listed.

Example

```
LOG 205..276 /FROM_TIME=2014-01-02 /TO_TIME=2014-02-01
    /WORKSET=QLARIUS:S1 /LOGFILE=C:\output /COMMENT="java"
    /FILE="**/*.java"
```

Parameters and Qualifiers

- <versionRange>

  Specifies a range for listing stream or project versions. If omitted all versions are listed. Format:

  `NN..MM`

  where NN and MM are the stream or project version numbers and `NN < MM`.

  The following formats are also allowed:

  - `MM` - lists the changeset associated with this version.
  - `..MM` - lists all changesets up to version MM.
  - `NN..` - lists all changesets from version NN.

- /FROM_TIME

  Specifies the start timestamp. Only changesets created after this timestamp are listed. Format:

  `"YYYY-MM-DD"`

  or

  `YYYY-MM-DD HH24:MI:SS`

  You can also use these values:

  - `NOW`: the full date time value, for example: 2014-04-15 11:46:00
  - `TODAY` or `"."`: derived from the date in the value NOW, for example: 2014-04-15

- YESTERDAY: derived from the date in the value NOW for example: 2014-04-14

For example:

- LOG /F=. or LOG /F=TODAY lists any changesets that were created today.
- LOG /F=YESTERDAY lists any changesets that have been created since yesterday.

- /TO_TIME

  Specifies the end timestamp. Only changesets created before this timestamp are listed. Format:

  "YYYY-MM-DD"

  or

  YYYY-MM-DD HH24:MI:SS

  You can also use the values described above.

- /WORKSET

  Specifies the stream or project whose history is to be listed. If omitted, the current stream or project history is listed.

- /BRIEF

  Enables brief mode (changeset details are not listed).

- /COMMENT

  Filters the changesets by their comments (uses a regular expression). For example:

  - LOG /comment="java": only lists changesets whose comments include the word "java" (in any case).
  - LOG /comment="^java": only lists changesets whose comments begin with the word "java".
  - LOG /comment="java$": only lists changesets whose comments end with the word "java".

- /FILENAME

  Filters the changesets by paths that contain changes (uses an ANT pattern). For example:

  - LOG /FILE="**/*.java": only lists changesets that contain changes in any Java files.
  - LOG /FILE="build/pcwin/pcwin.cpp": only list changesets that contain changes to "build/pcwin/pcwin.cpp".

- /LOGFILE

  Specifies the name of the output file.

- /CHANGE_DOC_ID

  Finds all the versions of a stream, or project, that contain changes associated with the request that you specify. For example, to find the versions of TESTSTREAM that contain changes related to the request TEST_CR_1:

  LOG /WORKSET=TESTSTREAM /CHANGE_DOC_ID=TEST_CR_1

Output Format

The output of the LOG command has the following format:

```
<stream or project version> | <userId> | <changeset date> | <comment> |
    <change type code> | <object class> | <path|revision> |
    <associated requests>
```

where:

- <change type code> can be one of the following:
  - C - a new item or folder was created.
  - I - an item revision was imported into the stream or project.
  - M - a new item revision was created from another item revision.
  - R - an item or directory was removed from the stream or project.
  - PR - an item or directory was renamed in the stream or project.
  - PM - an item or directory was moved in the stream or project.
  - PI - an item was promoted in the stream or project.
  - DI - an item was demoted in the stream or project.
- <object_class> is either D or I and indicates whether the change is applicable to a folder or an item revision.

Examples:

```
------------------------------------------------------------------------
2822 | USER1 | 2014-02-03 05:40:56 | remove residual metadata files
          M | I |
Cruisecontrol/projects/win-vc10/scripts/cleanBuild.sh;cm_vrs#1 |
DMPROD_EC_5763
------------------------------------------------------------------------
2823 | USER2 | 2014-02-03 05:42:54 | Fixed Solaris compilation
          M | I |
build/libpcmscore/sync_xnode_request_resolver.cpp;cm_vrs#1 |
    DMPROD_EC_5771
          M | I |
build/libpcmscore/sync_xnode_request_resolver.h;cm_vrs#1 |
    DMPROD_EC_5771
------------------------------------------------------------------------
2824 | USER3 | 2014-02-03 09:08:53 | Automatic legacy metadata
    conversion to .dm format on first access
          C | I | build/libmsgservices/timetracer.cpp;cm_vrs#1 |
DMPROD_ECR_42828
          M | I | build/libmsgservices/stdafx.h;cm_vrs#1 |
    DMPROD_ECR_42828
          M | I | build/libmsgservices/libmsgservices.mk;cm_vrs#1 |
DMPROD_ECR_42828
          C | I | build/libmsgservices/msgfun.h;cm_vrs#1 |
    DMPROD_ECR_42828
         PM | I | build/libpcmscore/timetracer.h ->
build/libmsgservices/timetracer.h;cm2010r1_team2#2 | DMPROD_ECR_42828
```

LOS – List Existing Operating Systems

No parameters.

Example LOS

This command enables you to list an installation's existing operating systems. See the *System Administration Guide* for details.

LPRIV – List Privileges

No parameters.

Description

This command lists all privileges.

LPROJ – List Dimensions Projects and Build Projects

NOTE This command is no longer available; use LWS instead.

LPRP – List Preservation Rules Policies

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<product-id>
```

Example

The following command:

```
LPRP PAYROLL
```

list preservation rules policies defined in the PAYROLL product.

Parameters and qualifiers

- `<product-id>`

  Specifies the product restricting the list of policies to be displayed.

Description

List preservation rules policies defined in a product (for information about preservation rules policies, see "DPRP – Define Preservation Rules Policy" on page 203).

Constraints

None.

LPRT – List Existing Network Protocols

No parameters.

This command enables you to list existing network protocols used by an installation network object. See the *System Administration Guide* for details.

LPSP – List Per-Stage Project Properties

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
```

Example

The following command:

```
LPSP "EXEDLL:EXEDLL 2.0"
```

list per-stage project stage properties set in the "EXEDLL:EXEDLL 2.0" project.

Parameters and qualifiers

- `<project-spec>`

  Comprises `<product-id>:<project-id>` and specifies the project specification.

Description

List per-stage project properties set in a project (for information about per-stage project properties, see "SPSP – Set Per-Stage Preservation Policy" on page 439).

Constraints

None.

LRC - List Request Changes

```
<request-id>
[/WORKSET=<workset-spec>]
[/[NO]VERBOSE]
```

Description

Lists the changesets, and associated changes, related to the request that you specify.

Qualifiers

- `<request-id>`

  Specifies a request ID.

- `/WORKSET=<workset-spec>`

  Specifies a project or stream.

- `/[NO]VERBOSE`

  Adds item specifications to the output.

LRSD – List Existing Resident Software Definitions

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

No parameters.

Example `LRSD`

This command enables you to list an installation's existing Resident Software Definitions (RSDs). See the *System Administration Guide* for details.

LSAR – List Archives

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

This command lists all archives created by ART.

LSBL – List Baselines

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
LSBL
[<product-ID>]
```

- `<product-ID>`

  Specifies a product for which baselines are to be listed.

Example Output

```
Dimensions>lsbl
Listing of baselines currently in the database for all products:
        Baseline Id: PAYROLL:BL_DEV_REL_1_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_DEV_REL_1_B
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_DEV_REL_2_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_VBGUI_REL_1_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:INITIAL
        Created by DMSYS
        Associated Project: (None)
Operation completed

Dimensions>lsbl payroll
Listing of baselines currently in the database for user-specified
    product:
        Baseline Id: PAYROLL:BL_DEV_REL_1_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_DEV_REL_1_B
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_DEV_REL_2_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:BL_VBGUI_REL_1_A
        Created by DMSYS
        Associated Project: (None)
        Baseline Id: PAYROLL:INITIAL
        Created by DMSYS
        Associated Project: (None)
Operation completed
```

Description

This command supports the ISPF panels client baseline build facility.

LSJ – List Scheduled Jobs

```
[<job-id>]
[/FROM_TIME]
[/TO_TIME]
[/JOB_STATUS]
[/ORIGINATOR]
[/SORTING]
[/JOB_HIST]
[/COMMANDS]
```

Examples:

```
LSJ /FROM_TIME="07-12-2007 11:06" /JOB_STATUS="ACTIVE, RUNNING" /
    SORTING="NAME"
LSJ "MyJobName" /JOB_HIST /COMMANDS
```

Parameters and qualifiers

- <job-id>

  Specifies the schedule job name. If you omit this qualifier, all scheduled jobs are listed.

- /FROM_TIME

  Displays scheduled jobs whose start time is greater than, or equal to, the value that you specify. Use the format 'DD-MM-YYYY HH24:MI:SS', for example:

  /FROM_TIME="31-12-2008 23:59:59".

- /TO_TIME

  Displays scheduled jobs whose start time is less than, or equal to, the value that you specify. Use the format 'DD-MM-YYYY HH24:MI:SS', for example:

  /TO_TIME="31-12-2008 23:59:59".

NOTE

- If you omit both /FROM_TIME and /TO_TIME, jobs are not filtered by their start time.
- If you specify /FROM_TIME, only jobs with a start time greater than, or equal, to /FROM_TIME are listed.
- If you specify /TO_TIME, only jobs with a start time less than, or equal to, /TO_TIME are listed.
- If you specify both /FROM_TIME and /TO_TIME, all jobs that have a start time between these two times are listed.

- /JOB_STATUS

  Filters job statuses and only displays the statuses that you specify. Can be one or more of the following: ACTIVE, INACTIVE, RUNNING, CANCELLING. For example:

  /JOB_STATUS="ACTIVE, RUNNING"

  /JOB_STATUS="INACTIVE"

- /ORIGINATOR

  Displays jobs created by the user that you specify, for example:

  /ORIGINATOR="DMSYS"

- /SORTING

  Specifies the way that job lists are ordered, can be one or more of the following values:

  - NAME: jobs are ordered by name (job-id).
  - DATE: jobs are ordered by scheduled date/time (START_TIME).
  - STATUS: jobs are ordered by current status (JOB_STATUS).

  To use a combination of values, separate with commas.

  You can also optionally specify keywords to sort a list in ascending (ASC) or descending (DESC) order. The default is ASC.

  For example:

  /SORTING="STATUS ASC, DATE DESC"

- /JOB_HIST

  Displays the execution history of jobs.

- /COMMANDS

  Displays all the commands related to the scheduled job(s).

Description

Lists scheduled jobs.

LSTG – List Stages

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

This command does not have any options.

Sample Output

```
Dimensions>lstg
Listing of stages for this database:
        Stage Id: DEVELOPMENT
                Description: Development stage
                Created: 31-JAN-2001 16:24:53 by dmsys
        Stage Id: EMERGENCY
                Description: Emergency stage
                Created: 31-JAN-2001 16:24:53 by dmsys
        Stage Id: RELEASE
                Description: Release stage
                Created: 31-JAN-2001 16:24:53 by dmsys
        Stage Id: SYSTEM TEST
                Description: System test stage
                Created: 31-JAN-2001 16:24:53 by dmsys
        Stage Id: UNIT TEST
                Description: Unit test stage
                Created: 31-JAN-2001 16:24:53 by dmsys
Operation completed
```

Description

Lists the contents of the Global Stage Lifecycle.

LUPG - List Upgrade History

```
[/HISTORY_TYPE=ALL | LATEST]
[/NETWORK_NODE=<node name>]
```

Description

Lists the upgrade history stored in a database.

Example

List the latest upgrade history for the network node ST6123:

```
LUPG
/HISTORY_TYPE=LATEST
/NETWORK_NODE=ST6123
```

Qualifiers

- `/HISTORY_TYPE=ALL | LATEST`

  Specify LATEST to only include the most recent upgrade history for each network node.

- `/NETWORK_NODE=<node name>`

  Only list history for the specified node.

LWC – List Project Conflicts

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
LWC
[<project-spec>]
```

Examples

```
LWC "PROD_X:WS MAINT DVL"
```

Parameters and qualifiers

- `<project-spec> comprises:`

  `<product-id>:<project-id>`

  Specifies a project for which conflicting item revisions are to be listed. If no project specification is provided, the user's default project is used.

Description

This command will list all the conflicting item revisions in a project that need to be resolved, the user who created those conflicts, and what the common ancestors for those conflicts are.

Constraints

`This command requires the Run Reports privilege.`

LWS – List Projects

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
[/FILENAME=<file-name>]
```

Example

```
LWS /FILENAME=worklist.txt
    [/SEARCH=<regular expression>]
```

- `/FILENAME=<file-name>`

  Specifies that the list of projects is output into the file specified in your 'home' directory.

  LWS without this qualifier outputs the list to the screen (`stdout` on UNIX systems).

  For each project, the associated product, project-id, project-status, project-owner (the user who created the project), associated project, and users with the role of `WORKSET-MANAGER` are detailed, see example below.

- `/SEARCH="<regular expression>"`

  A regular expression pattern matches a target string. For example:

  `LWS /SEARCH="2015R[0-9]"`

  only lists projects and streams with IDs that contain "2015R<any digit>".

- `/ALL`

  Lists streams and projects that have been hidden by the HIDE command on page 256. By default hidden streams are not listed.

- `/FAVORITE`

  Only lists your favorite projects and streams. See the SF command on page 431.

Example Output

```
List of projects currently in use
=================================

Product:        $GENERIC  Project Id:                  $GLOBAL
- Status: UNLOCKED
- Owner : DMSYS

Product:         QLARIUS  Project Id:           VS_TYPICAL_1.0
- Status: OPEN
- Owner : DMSYS
- Project Manager: DMSYS

Product:         QLARIUS  Project Id:          JAVA_TYPICAL_1.0
- Status: OPEN
- Owner : DMSYS
- Project Manager: DMSYS

Product:         QLARIUS  Stream Id :                MAINLINE
- Status: OPEN
- Owner : DMSYS
- Parent: QLARIUS:JAVA_TYPICAL_3.0 (Project)
```

```
- Project Manager: DMSYS

Product:         QLARIUS  Stream Id :                  STREAM_A
- Status: OPEN
- Owner : DMSYS
- Parent: QLARIUS:JAVA_TYPICAL_3.0 (Project)
- Project Manager: DMSYS
```

Constraints

None

LWSD – List Project Directories

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<directory-path>
[/RECURSIVE]
[/LATEST_REV]
[/FILES] or
   [/ITEMS] or
   [/FILES/ITEMS]
[/USER_FILENAME=<file-name>]
[/WORKSET=<project-spec>]
[/PERMISSIONS]
```

Example

```
LWSD src
```

Parameters and qualifiers

- `<directory>`

  The LWSD command lists:

  - The current project-id or that specified by `/WORKSET`.
  - The identities of the operating system directories of the projects at subdirectory `<directory>` relative to the working location.
  - The project description.
  - The date at which the list was taken.
  - The items the project directories contain. The following is detailed for each item: its owner, its update date, its status, whether or not it is checked out and its specification and/or file name.

- `/RECURSIVE`

  recursively list all project directories from the point defined above.

- `/LATEST_REV`

  list only the latest (tip) revisions present in each project directory (if any).

- `/FILES`

  list items using item library file names (the default).

  - `/ITEMS`

    lists items using (traditional Dimensions) item specifications.

  - `/FILES/ITEMS`

    list items using both item-library file names and (traditional Dimensions) item specifications.

- `/USER_FILENAME=<user-filename>`

  Specifies that the list is to be output to a file `<user-filename>` rather than to the screen.

- `/WORKSET=<project-spec> comprises:`

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

- `/PERMISSIONS`

  Specifies that the list will display the file permissions and other properties in a format similar to the UNIX 'ls -l' command.

Constraints

None

MCPC – Move Request To Primary (Main) Catalog

```
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
```

Example

```
MCPC /CHANGE=(PROD_DC_22,PROD_DC_23)
```

Parameters and qualifiers

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>`

  Identifies the request(s) that are to be moved from the secondary catalog to the primary (main) catalog.

NOTE The secondary catalog is intended mainly for requests that are no longer active, and therefore users are not permitted to update a request in this catalog.

Constraints

Only users with the appropriate management privileges can run this command.

Update functions are available only from the main catalog.

MCSC – Move Request To Secondary Catalog

```
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
```

Example

```
MCSC /CHANGE=(PROD_DC_30,PROD_DC_31)
```

Parameters and qualifiers

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>`

  Identifies the request(s) that are to be moved from the primary (main) catalog to the secondary catalog.

NOTE The secondary catalog is intended mainly for requests that are no longer active, and therefore users are not permitted to update a request in this catalog.

Constraints

Only users with the appropriate management privileges can run this command.

Update functions are available only from the main catalog.

MDR – Move Design Part Relationship

```
<part-spec>
/FATHER_PART=<part-spec>
```

Example

```
MDR PROD:"RELEASE MANAGEMENT".AAAA -
    /FATHER_PART=PROD:"CONFIG DEF"
```

Parameters and qualifiers

- `<part-spec>`

  (both for the moved design part and for the new owner parent part[1]) comprises:

  ```
  <prod-id>:<part-id>.<variant>;<pcs>
  ```

  `<variant>` may be omitted if only one variant of that design part exists.

  `<pcs>` is ignored; the current PCS is always used.

Constraints

Only users with the appropriate management privileges can run this command.

MERGE - Merge into Stream Work Area

```
/USER_DIRECTORY=<directory-path>
[<file-spec> or /USER_DIRECTORY=<directory-spec> or
/USER_FILELIST=<filelist-file>]
[/[NO]RECURSIVE]
[/[NO]TOUCH]
[/LOGFILE=<file-spec>]
[/STREAM=<stream-id>]
[/RELATIVE_LOCATION=<directory-spec>]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/BASELINE=<baseline-spec>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,…)]
[/[NO]CHERRYPICK
[/[NO]CANCEL_TRAVERSE]
[/CODEPAGE=<cp>]
[/[NO]QUIET]
[/[NO]VERBOSE]
[/[NO]EXECUTE]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
[/[NO]AUTO_MERGE]
[/ACCEPT=LOCAL|REPOSITORY]
[/ANCESTOR]
```

Description

The MERGE command merges changes from one stream into a work area owned by another stream using a three-way merge process based on the common ancestor stream version. The command compares the target work area with the stream or baseline and automatically applies any non-conflicting content and refactoring changes. After the merge into the target work area is completed use the DELIVER command to commit merged changes from the target work area into the target stream.

NOTE The MERGE command is only available for streams.

Examples

- `MERGE /USER_DIRECTORY=C:\work\mainline /STREAM=FEATURE`

  Merges the work area located in `C:\work\mainline` with the tip of the FEATURE stream.

- `MERGE /USER_DIRECTORY=C:\work\mainline /STREAM=FEATURE /DIRECTORY="build"`

  Merges the work area located in `C:\work\mainline` with the tip of the directory "build" in the FEATURE stream.

- `MERGE /USER_DIRECTORY="C:\work\mainline" "C:\work\mainline\build.mk" /BASELINE="PVCS:DM10 TIER1 FINAL"`

  Merges the file `C:\work\mainline\build.mk` with the baseline item revision with the file name `build.mk` from the baseline PVCS:DM10 TIER1 FINAL.

Parameters and Qualifiers

- /USER_DIRECTORY=<directory-path>

  Specifies the target merge work area. This work area must be empty or owned by a stream other than the one you are merging from. For example

  - The following command merges a stream into C:\temp:

    MERGE /USER_DIRECTORY="C:\temp"

  - The following command merges from a stream to the /tmp directory on the host "hostname":

    MERGE /USER_DIRECTORY="hostname::/tmp"

  - The following command merges from a stream into the src directory inside the area area_name:

    MERGE /USER_DIRECTORY="area_name::src"

- <file-spec>

  Specifies the name of the file to be updated during the merge process. The Dimensions node:: syntax is also valid.

- /DIRECTORY=<directory-spec>

  Specifies a stream folder that is to be merged into the matching folder of the target work area.

- /USER_FILELIST=<filelist-file>

  Specifies a file containing a list of file names to be merged from the stream. Each file name must be on a separate line. File names may be specified as either absolute or relative paths. If the path is absolute, it is interpreted as a full work area path. If the path is relative, Dimensions obtains the stream path by mapping the file name to the operation root directory specified by one of the following:

  - The /USER_DIRECTORY qualifier.
  - The current working location specified by the last SCWS command.

  If this mapping is not possible the file name is ignored.

- /[NO]RECURSIVE

  If /DIRECTORY is specified and this qualifier is not present, all files that have not been modified in all directories beneath the one specified are copied to the work area. /NORECURSIVE specifies that only files at the specified directory level are updated.

  Default: /RECURSIVE

- /[NO]TOUCH

  Applies the system date/time to each file being transferred to the work area.

  Default: /TOUCH

  /OVERWRITE overrides the /NOADD qualifier. If /OVERWIRITE and /NOADD are both specified, /NOADD is ignored.

- /LOGFILE=<file-spec>

  Specifies that a log file is generated at the specified file location. The log contains the results of all the individual Dimensions CM operations executed with this command.

- /STREAM=<stream-id>

  Specifies the stream from which to retrieve the files. If this parameter is not specified, files are retrieved from the current session stream.

- /RELATIVE_LOCATION=<directory-spec>

  Specifies a project, stream, or baseline directory that is to be the "virtual" root directory for the duration of this command. If this parameter is used the paths specified in <file-spec> or /DIRECTORY must be relative to the directory specified in /RELATIVE_LOCATION.

- /FILTER=<filter-name>

  Specifies a filter that will only retrieve files that satisfy the criteria specified in the area filter <filter-name>.

- /USER_FILTER=<filter-file-spec>

  Specifies the name of a local file containing the definition of a file filter to be used when getting files or checking in files. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502. Only files matching the filter (and not excluded by the filter) will be copied to the work area when a user filter is specified.

- /BASELINE=<baseline-spec>

  Specifies a baseline to merge into the area.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  Specifies that all content and refactoring changes associated with the related *In Response To* requests or child requests are applied to the target work area.

- /CHERRYPICK

  Enables you to select a specific set of changes when you merge requests between streams, for details and examples see the *User's Guide*.

  Default: enabled

  To disable cherrypicking specify /NOCHERRYPICK.

- /CANCEL_TRAVERSE

  By default all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the specified request.

- /CODEPAGE=<code-page> | DEFAULT

  Specifies the code page to be associated with the items. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.

/CODEPAGE is relevant only for text files. Whenever a text file is checked out or fetched it must be in the right code page for the target platform so that it displays correctly. Binary files are moved between platforms with no conversion.

For further details about code pages and logical nodes see Network Administration in the System Administration Guide.

You are advised to let the parameter default to the code page for the item type or platform.

The /CODEPAGE options are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.

- /QUIET

  Only print critical messages.

- /VERBOSE

  Print additional information about the update process.

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when updating text files. The options are:

WINDOWS	Fetched text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair regardless of the client operating system.
UNIX	Fetched text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files fetched to a Windows node have each line terminated with a CR/LF pair. Text files fetched to a UNIX node have each line terminated with a single LF character.
UNCHANGED	Text files are fetched as-is from the item library without any end-of-line processing.
SHOW	Display current EOL setting.

  See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

- /[NO]AUTO_MERGE

  If you specify this qualifier the MERGE command tries to perform an automatic merge of conflicting file content when certain types of conflicts are detected. The automatic merge occurs in a temporary location and the result is copied to the work area if the merge completes without any conflicts.

  For example, assume that the home work area of a stream contains revision 2 of a locally modified file, foo.c. The corresponding home stream contains revision 4 of foo.c. By default, the MERGE command flags this as a conflict and leaves the locally modified file as is. If you specify /AUTO_MERGE the MERGE command attempts to

perform an automatic merge of the locally modified revision and the newer repository revision. If the merge succeeds the merged file is placed in the work area and its metadata updated to revision 4. The revision of foo.c in the work area is now the latest version and is the same as the repository.

- `/ACCEPT=LOCAL | REPOSITORY`

  If you specify this qualifier the MERGE command uses the local or repository path of a file when resolving automatic merge conflicts that include file path renames or moves, in addition to file content conflicts.

  For example, assume that the home work area of a stream contains revision 2 of a locally modified file, foo.c. The corresponding home stream contains a renamed revision 4 of foo.c, that is now called bar.c. By default, the MERGE command flags this as a conflict and leaves the locally modified file as is.

  If you specify `/AUTO_MERGE` the MERGE command attempts to perform an automatic merge of the locally modified file and the newer repository revision. Because of the path conflict, if the merge succeeds the merged file is not placed in the work area unless you also specify the /ACCEPT qualifier:

  - If you specify `/ACCEPT=LOCAL` the merged file is copied to the work area under the local path of foo.c and a 'moved-from' property is added.
  - If you specify `/ACCEPT=REPOSITORY` the merged file is copied to the work area under the repository path of foo.c and the old work area file is deleted.

  If you do not specify `/AUTO_MERGE` the `/ACCEPT` qualifier is ignored.

- `/ANCESTOR`

  Explicitly specifies a stream version or a baseline to be used as the ancestor for a three-way merge. This is particularly useful when performing the initial merge of two unrelated streams or baselines or when redoing an erroneous merge.

  Syntax:

  `/ANCESTOR=<workset-spec.[;<stream version>]`

  or

  `/ANCESTOR=<baseline-spec>`

MI – Merge Item Revisions

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
/REVISION_LIST=(<merge-rev-1>,<merge-rev-2>,...)
/USER_FILENAME=<merged-file>
[/REVISION=<new-revision>]
[/COMMENT=<comment-text>]
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<item-filename>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/WORKSET=<project-spec>]
[/[NO]KEEP]
[/CONTENT_ENCODING=<file-encoding>]
[/NOMETADATA]
[/STATUS=<merge-status>]
[/SELF]
```

Example

```
MI PROD:"QUERY RELEASE".AAAA-SRC;main#1 -
/REVISION="main#2" -
/REVISION_LIST=("patch1#1","patch2#2")
/USER_FILENAME="patch3#1"
/KEEP -
/COMMENT="Merge maintenance work into main line"
```

Parameters and qualifiers

- `<item-spec>`

  Specifies the primary item revision that is to be merged. It comprises:

  `product_id>:<item_id>.<variant>–<item-type>;<revision>`

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision in the project specified by `/WORKSET`. If `/WORKSET=` is unspecified, the current project will be assumed.

NOTE This cannot be a directory item.

- `/REVISION_LIST=(<merge-rev-1>,<merge-rev-2,...)`

  Specifies each revision to be merged with the primary item revision.

- `/USER_FILENAME=<merged-file>`

  Specifies the name of the file containing the data for the merged item revision.

- `[/REVISION=new-revision>]`

  Specifies the new revision that is to be created.

- /COMMENT=<comment text>

  comment text to explain the reason for the creation of this merged item revision. The comment text can be up to 1978 characters long.

- /ROOT_PROJECT=<project-spec>

  Comprises:

  <product-id>:<project-id>

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- /FILENAME=<file-name>

  Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, src/hello.c, hello.c, or src/build/hello.c.

  It may be omitted if <item-spec> is specified.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  <requestN> identifies one or more request to which the merged item revision is to be related In Response To.

NOTE Mandatory if CM rules are enabled.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ...)

  <attrN> is the Variable Name defined for one of the user-defined attributes, either applicable to items of all item types, or applicable to those of the <item-type> specified in <item-spec>.

  <valueN> is the value to be given to this attribute.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  This **optionally** specifies the project to be used for this command: failing this, the current project will be taken.

  This qualifiter specifies the workset in which the primary revision must exist. (Other revisions can come from any workset, project, or stream). This qualifier also specifies in which workset, project, or stream the new revision will be placed if the /UDER_FILENAME qualifier is specified.

- [/KEEP]

  Specifies that the <user-filename> which is normally deleted once the item has been placed under Dimensions control, is to be left intact.

- <file-encoding>

Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- `/NOMETADATA`

  This parameter disables creation and usage of metadata files in the local work area.

- `[/STATUS=<merge-status>]`

  Specifies the result status for a new merge.

- `[/SELF]`

  Performs a logical merge of the revisions listed in the /REVISION_LIST into the primary revision.

  You cannot use this qualifier with /USER_FILENAME.

Description

MI is used to merge two or more revisions of the same item.

When the /REVISION qualifier is specified, the <*new-revision*> is created from the revision identified by <*item-spec*> using the specified <*merged-file*> as the user file for <*new-revision*>. Merge records are created showing that each revision specified in /REVISION_LIST has been merged into <*new-revision*>.

/REVISION_LIST must be specified. /USER_FILENAME must be specified unless /SELF is specified and /REVISION is not specfied.

If a user file and /KEEP are specified, the local metadata is updated after a successful merge.

Constraints

This command can be run by users who have one of the roles required to action the item from the initial lifecycle state to a new lifecycle state.

MIP – Move Item to Another Part

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/PART=<part-spec>]
[/WORKSET=<project-spec>]
```

Example

```
MIP PROD:"QUERY RELEASE".AAAA-SRC -
/PART=PROD:"RELEASE CONTROL".AAAA
```

Parameters and qualifiers

- `<item-spec>` comprises:

  `<product_id>:<item_id>.<variant>–<item-type>;<revision>`

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are moved to the specified design part.

- `/FILENAME=<file-name>`

  Specifies the name of the library file name.

- `/PART=<part-spec>` comprises:

  `<product_id>:<part_id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Constraints

1. Only users with the appropriate management privileges can run this command.
2. MIP cannot move an item to another part that belongs to a product that is different from that owing the item, that is, the `<product-id>` component of the `<part-spec>` must be identical to the `<product-id>` component of the `<item-spec>`.
3. If the item is related to a request, then the following rules apply:
   a. MIP cannot move the item to another part if the item is in an Affected or In Response To relationship to the request, except when the request is in a closed, rejected, or held state.
   b. MIP can move the item to another part if the item is in an Info relationship to the request, regardless of the request's state.
   c. MIP can move the item to another part if the item is in a relationship to a request in the secondary catalog.
4. MIP will fail if the part specified is already the owner of the item.
5. MIP will fail if the part specified is already in a USAGE relationship with the item.

MIT – Move (Change) Item Type

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
/TYPE=<new-item-type>
```

CAUTION! *All* revisions of the item are moved (changed) to the new item type *regardless* of the revision (explicit or implied) that is specified in `<item-spec>`.

Example

```
MIT PROD:"MAIN".AAAA-SRC /TYPE=CODE
```

In this example the item's type is changed from `SRC` to `CODE`.

Parameters and qualifiers

- `<item-spec>` comprises:

  `<product_id>:<item_id>.<variant>-<item-type>;<revision>`

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are moved to the specified design part.

- `/TYPE=<new-item-type>`

  Specifies the new type to be assigned to the item.

Constraints

- Only users with the appropriate management privileges can run this command.
- You *cannot* move an item's type if the item:
  - Is stored in a Delta library.
  - Is stored in an item library on a mainframe (z/OS).
  - Is on a named branch that remotely owned e.g. the item has been replicated (see *System Administration Guide*).
  - Is in the Dimensions OFFLINE state (see the *System Administration Guide*).
  - Has been included in a Dimensions build.
  - Is in a Dimensions baseline.
  - Is in a Dimensions release.
  - Is checked out.
  - Is referenced in a request.
  - Has other items related to it.

- If the current state of the item does not match (by name) any state in the new item-type's lifecycle, then the state of the item is reset to the new item-type's initial lifecycle state and a warning is issued.
- If an attribute of the current item-type has a name that does not match one of the attribute names for the new item-type, then the attribute values for that name are not copied and a warning is issued.

MVC – Move Request

```
<top-request-id>
<target-product-id>
[/CH_DOC_TYPE=<target-request-type>]
[/AFFECTED_PARTS=(<target-part-spec1>,...)]
[/CHANGE_DOC_IDS=(<doc1>,<doc2>,...)]
[/[NO]CHECK]
```

Example

```
MVC OBLR_DR_52 PROD
/CHANGE_DOC_IDS=(OBLR_DR_53, OBLR_DR_55, OBLR_DR_56)
/NOCHECK
```

Parameters and qualifiers

- `<top-request-id>`

  Identifies the (principal) request in the source product. Its dependent-related children can also be specified for moving at the same time (as detailed below).

  More exactly, the request(s) are "cloned" rather than moved: the originals are flagged to a special state `$MOVED`, once clones of them have been created in the target product.

- `<target-product-id>`

  Identifies the target product, where the request(s) is (are) to be moved. It must be another product in the same Dimensions database. (To copy requests to a different database, the baseline transfer facility of Dimensions ART may be used.)

- `/CH_DOC_TYPE=<target-request-type>`

  Specifies the request type that the clone of `<top-request-id>` is to have in the target product. The dependent children moved, if of the same type as `<top-request-id>`, are also translated to this type. (Any other dependent children do not receive a type translation.)

  If omitted, all the cloned requests in the target product are created with the same type(s) as the originals in the source product.

- `/AFFECTED_PARTS=(<target-part-spec1>,...)`

  Specifies one or more design parts in the target product that are to be related to the clone of `<top-request-id>`.

  If omitted, just the target product's top (i.e. product level) design part (its original variant) is related to this cloned request.

- `/CHANGE_DOC_IDS=(<doc1>,<doc2>,...)`

  This is a comma separated list of requests, where each `<docN>` is of the form `<source-request-id>`.

  The entries `<source-request-id>` each identify a request that has a relationship to `<top-request-id>` in the Dependent class and that is to be included in the group move.

  If the `/CHANGE_DOC_IDS` qualifier is omitted, then just `<top-request-id>` is moved by itself.

- `/CHECK` or `/NOCHECK`

  Specifies whether MVC is merely to check and report on the feasibility of moving the specified request(s), or is actually to implement the move.

The default is /CHECK: the move actually takes place only if /NOCHECK is specified.

Provided the Constraints are complied with, and /NOCHECK is specified, all the source-product requests specified acquire the status $MOVED, which is regarded as a non-normal final state; i.e. the phase becomes Rejected. The History record of each identifies its clone's request-id in the target product.

The cloned children acquire the relationship Dependent to the cloned parent (regardless of any specific relationship name the children had in the source product). All clones are actioned to their initial lifecycle states, but they are not placed in any users' pending lists. The History record of each identifies the source product's request-id from which it was cloned.

For each request moved, if there are user-defined attributes which have been declared for use by the product request types of both the original and the clone, the values of these are all inherited by the clone.

Constraints

- This command can be run only by a user with the appropriate management privileges for the source product, or by users if all the specified requests are in their Pending list.
- None of these requests can have any items related as Affected or In Response To. Info related items are permitted, but they are simply ignored when the clones are created.
- Related requests that are to be moved must all be related to the parent as Dependent and **not** Info.
- None of these requests can be related in the Info relationship class to any other requests, in either direction.
- The <top-request-id> must not be Dependent related to any grandparent. Dependent children not specified in /CHANGE_DOC_IDS are permissible: they are left unaltered as orphans in the source product. The specified children must not be related to any request other than <top-request-id>.
- The <top-request-id> must be at its initial lifecycle state in the source product, but the children can be at any states except final states (i.e. all the requests must still be Open).
- The parameter $LAST (see note on the CC command – page 90) is not set by MVC, so the cloned requests cannot be referenced subsequently in the same CMD command file.
- This command does not copy attribute history information.
- When CM rules are on, only requests at the Create phase can be moved.

MWS – Merge Projects

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for streams.

```
<project-spec1>
<project-spec2>
[/WORKSET=<project-spec3>]
[/ATTRIBUTES=(<attr1>,attr2,...)]
[/TYPE=<type-name>]
[/[NO]REPORT]
[/USER_FILENAME <filename>]
[/STATUS=<status>]
[/[NO]KEEP_STAGE]
```

Example

```
MWS PROD:WS_001 -
PROD:WS_002
/REPORT
/WORKSET=PROD_X:TEST_WS
```

Parameters and qualifiers

- `<project-spec1>`

  comprises the specification for project 1:
  `<product-id>:<project-id>`

  This project is one of the inputs to the merge operation, and is also the target project if `/WORKSET` is not specified.

- `<project-spec2>`

  comprises the specification for project 2.

  This project is the second input to the merge operation.

- `/WORKSET=<project-spec3>`

  This optionally specifies the target project, which will be created if it does not already exist.

- `/ATTRIBUTES=(<attr1>,attr2,...)`

  Specifies the user-defined attribute values for the target project.

- `/TYPE=<type-name>`

  Specifies the type of the target project. If this qualifier is not specified, the type name WORKSET is used.

- `/REPORT`

  Requests that only the Merge Project report is generated. Contains information about how the merge is processed.

- `/USER_FILENAME <filename>`

  Generates a report and places it in the specified file on the client system from which the command was invoked.

If /REPORT is specified and /USER_FILENAME is not specified, the report is written to the file 'mws_report.txt' in the current directory on the client system from which the command was invoked.

If both /REPORT and /USER_FILENAME are not specified, no report is generated.

- /STATUS=<status>

  allows the merged project to be created in either a LOCKED or UNLOCKED state. The locked state prevents new Dimensions items being added to the project (baselining is likely to occur in this state).

- /KEEP_STAGE

  Specify this optional qualifier to control the stages of the items in the merged project.

  - Use /NOKEEP_STAGE to reset the stages of all the items in the merged project to the initial stage.
  - Use /KEEP_STAGE to keep the stages of the item revisions from the source projects.

  This qualifier can only be used when the merged project uses the manual deployment model.

  Default (when the qualifier is not specified): /KEEP_STAGE

Description

The MWS command merges the two projects specified by <project-spec1> and <project-spec2>. The merged output is placed in a target project, which is specified by one of the following:

- <project-spec3>
- <project-spec1> (if <project-spec3> is not specified).

If /WORKSET is specified and the target project already exists, MWS merges <project-spec2> with the target project, ignoring <project-spec1>.

Constraints

This command can be run only by a user with the appropriate management privileges for the target project concerned.

MWSD – Move Project/Stream Directory

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<directory-path1>
<directory-path2>
[/WORKSET=<project-spec>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/MERGE <dir1> <dir2>]
```

Example

```
MWSD src dst
```

Parameters and qualifiers

- `<directory-pathN>`

  The MWSD command moves the project structure (and items) from the source directory to the destination directory.

`<directory-path1>`	is the source directory.
`<directory-path2>`	is the destination directory.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project/stream to be used for this command: failing this, the user's current project/stream will be taken.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which this structural change to the project is to be related `In Response To`.

  Specify this optional qualifier if you want this structural change to the project/stream to be recorded against the specified request(s). If path control has been enabled, this qualifier is mandatory. If path control is not enabled, then the request(s) will be ignored.

- `[/MERGE <dir1> <dir2>]`

  Merges two directories. Directory `<dir1>` is merged into `<dir2>`.

NOTE Whenever a new revision is added to a project/stream, its stage is reset to DEVELOPMENT, and associated deployment areas and library cache areas are updated.

Constraints

Normally this command can be run only by a user with the appropriate management privileges for the project concerned.

This constraint can, however, be relaxed using the Set Project Permissions (SWSP) command, as described on page 451.

PA – Populate Areas

NOTE

This command is not available in the Issue Management–only licensed version of Dimensions.

This command is not supported in products that use the Dimensions Automation deployment model.

```
<project-spec>
[/STAGE=<stage>]
[/AREA=<area-name>]
[/USER_FILENAME=<population-log-file-name>]
```

Example

```
PA DEPLOYMENT:DMNET /STAGE=RELEASE
```

Parameters and qualifiers

- `<project-spec>`

  Specifies the project for which areas are to be populated.

- `/STAGE=<stage>`

  If this parameter is specified, only areas associated with this stage will be populated.

- `/AREA=<area-name>`

  If this parameter is specified, only this area will be populated. The area must have been previously related to the project with the RAWS command.

- `/USER_FILENAME=<population-log-file-name>`

  Specifies the name of the file to contain the log of the area population.

Description

The PA command repopulates online areas associated with a project. If none of the optional qualifiers is specified, all online areas associated with the project are populated.

NOTE The PA command replaces the PBA command.

Constraints

Only users with the "Populate Area from Project" privilege can run this command.

PBA – Populate Build Area

NOTE This command is no longer available; use PA (Populate Area) instead.

PEND – Update Users' Pending Request Lists

Syntax (1st form)

```
PEND CHDOC /PRODUCT=<product-id>
[/CHANGE_DOC_IDS=(<request-id1>,<request-id2>, ...)]
```

Syntax (2nd form)

```
PEND CHDOC /PRODUCT=<product-id>
[/STATUS=<status>] [/TYPE=<request-type>]
    [/USER=<user-id>] [/ROLE=<role>]
```

NOTE These two forms of syntax are exclusive-or selections; you cannot mix the various parameters. If the syntax is in the second form, the utility processes every request that meets all the criteria specified. In this form, *wild card % may be used in any of the parameters*; and omission of a parameter is the equivalent of its inclusion with just a value of %.

Example

```
PEND CHDOC /PRODUCT=PAYROLL -/CHANGE_DOC_IDS=(PAYROLL_CR1,PAYROLL_CR_2)
```

```
PEND CHDOC /PRODUCT=PAYROLL /STATUS=RAISED /TYPE=CR /USER=FRED -
    /ROLE=DEVELOPER
```

Parameters and qualifiers

- `/PRODUCT=<product-id>`

  This is a string to be matched by the product-id of each request to be processed. This parameter is not optional, so just specify % if the processing is not to be limited by the value of product-id.

- `/CHANGE_DOC_IDS=(<request-id1>,<request-id2>, ...))`

  This is one of a list of request identifiers separated by spaces. Pending lists are recalculated only for these specified requests.

- `/STATUS=<status>`

  This is a string to be matched by the current status (lifecycle state) of each request to be processed.

- `/TYPE=<request-type>`

  This is a string to be matched by the type of each request to be processed.

- `/USER=<user-id>`

  This is a string to be matched by login user names. Each request is not processed unless it is currently in the pending list of at least one matching user name.

- `/ROLE=<role>`

  This is a string to be matched by role-titles of each request to be processed. Each request is not processed unless it is currently in the pending list of at least one user who has been assigned for it a role-title that matches this string.

Description

In the event that a user leaves a project, or change management rules are changed such that the phases are different, or design parts are moved around in the structure that could mean a change to the people responsible for requests, the PEND command allows a change-manager (only) to re-calculate the pending trays for users' requests. It will also recalculate the current phase of the request.

This command also needs to be run whenever rules are enabled (see *Process Configuration Guide*) for one or more of the product's request types, if at the time any requests of these types already exist.

IMPORTANT! Whenever this utility is to process more than just a few requests, it is highly recommended that it is executed only at times when database activity is otherwise light.

Constraints

Only users with the appropriate management privileges can run this command.

PEND – Update Users' Pending Item Lists

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

Syntax

```
PEND ITEM /PRODUCT=<product-id>
[/STATUS=<status>]
[/TYPE=<item-type>]
[/USER=<user-id>]
```

NOTE The utility will process every item that meets all the criteria specified. In this form, wild card % may be used in any of the parameters; and omission of a parameter is the equivalent of its inclusion with just a value of %.

Example

```
PEND ITEM
/PRODUCT=PAYROLL
/STATUS=DEFINED
/TYPE=SRC
/USER=FRED
```

Parameters and qualifiers

- `/PRODUCT=<product-id>`

  This is a string to be matched by the product-id of each item to be processed. This parameter is not optional, so just specify % if the processing is not to be limited by the value of product-id.

- `/STATUS=<status>`

  This is a string to be matched by the current status (lifecycle state) of each item to be processed.

- `/TYPE=<item-type>`

  This is a string to be matched by the type of each item to be processed.

- `/USER=<user-id>`

  This is a string to be matched by login user names. Each item is not processed unless it is currently in the pending list of at least one matching user name.

Description

In the event that a user leaves a project, or design parts are moved around in the structure that could mean a change to the people responsible for items/files (hereinafter referred to as items for brevity), the PEND command allows a product-manager (only) to re-calculate the pending trays for users' items.

NOTE Whenever this utility is to process more than just a few items, it is highly recommended that it is executed when there is little database activity.

Constraints

Only users with the appropriate management privileges can run this command.

This utility only runs on the currently selected project.

PEND – Update Users' Pending Baseline Lists

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

Syntax

```
PEND BASELINE /PRODUCT=<product-id>
[/STATUS=<status>]
[/TYPE=<baseline-type>]
[/USER=<user-id>]
```

NOTE The utility will process every baseline that meets all the criteria specified. In this form, wild card % may be used in any of the parameters; and omission of a parameter is the equivalent of its inclusion with just a value of %

Example

```
PEND BASELINE
/PRODUCT=PAYROLL
/STATUS=OPEN
/TYPE=DEVLPMENTSRC
/USER=FRED
```

Parameters and qualifiers

- `/PRODUCT=<product-id>`

  This is a string to be matched by the product-id of each baseline to be processed. This parameter is not optional, so just specify % if the processing is not to be limited by the value of product-id.

- `/STATUS=<status>`

  This is a string to be matched by the current status (lifecycle state) of each baseline to be processed.

- `/TYPE=<baseline-type>`

  This is a string to be matched by the type of each baseline to be processed.

- `/USER=<user-id>`

  This is a string to be matched by login user names. Each baseline is not processed unless it is currently in the pending list of at least one matching user name.

Description

In the event that a user leaves a project, or design parts are moved around in the structure that could mean a change to the people responsible for items/files (hereinafter referred to as items for brevity), the PEND command allows a product-manager (only) to re-calculate the pending trays for users' baselines.

NOTE Whenever this utility is to process more than just a few baselines, it is highly recommended that it is executed when there is little database activity.

Constraints

Only users with the appropriate management privileges can run this command.

PEND – Update Users' Pending Project Lists

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

Syntax

```
PEND WORKSET /PRODUCT=<product-id>
[/STATUS=<status>]
[/TYPE=WORKSET]
[/USER=<user-id>]
```

NOTE The utility will process every project that meets all the criteria specified. In this form, wild card % may be used in any of the parameters; and omission of a parameter is the equivalent of its inclusion with just a value of %.

Example

```
PEND WORKSET
/PRODUCT=QLARIUS
/STATUS=OPEN
/TYPE=WORKSET
/USER=dmsys
```

Parameters and qualifiers

- `/PRODUCT=<product-id>`

  This is a string to be matched by the product-id of each project to be processed. This parameter is not optional, so just specify % if the processing is not to be limited by the value of product-id.

- `/STATUS=<status>`

  This is a string to be matched by the current status (lifecycle state) of each project to be processed.

- `/TYPE=<item-type>`

  This is a string to be matched by the type of each project to be processed.

- `/USER=<user-id>`

  This is a string to be matched by login user names. Each project is not processed unless it is currently in the pending list of at least one matching user name.

PLCA – Purge Library Cache Area

Syntax

```
<library_cache_area_id>
[/purge_all]
```

Examples

- `plca lc1`

  Removes all files for the library cache area 'lc1' except the latest revisions of each file (allows the cache to stay partially up to date).

- `[plca lc1 /purge_all]`

  Removes all cached files from the library cache area 'lc1'.

Description

Removes files from a library cache area.

Constraints

Requires the privilege 'Update Library Cache Area Properties'.

PMBL – Promote Baseline

```
<baselineName>
/COMMENT="<userComment>"
/STAGE="<promotionStage>"
/USER_FILENAME="<listFile>
/[NO]QUIET
/AREA_LIST="<areaList>"
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
/WORKSET
/SDA_PROCESS=<SDA_Process>
/SDA_DEPLOY_COMPONENT=(<SDA_Component>=<Version name>)
/SDA_COMPONENTS=(<SDA_Component1>=<Version name1>,
    <SDA_Component2>=<Version name2>,...)
```

Example

```
PMBL "QLARIUS:KESTREL_RELEASE_2.1" /COMMENT="Kestrel release is ready
    for QA." /STAGE="QA" /WORKSET="QLARIUS:KESTREL_BRANCH" /
    SDA_PROCESS="ReleaseAutomation" /
    SDA_DEPLOY_COMPONENT=("BlnComp"="QLARIUS:KESTREL_RELEASE_2.1") /
    SDA_COMPONENTS=("3rdPartyComp"="3.4","DocsComp"="<LATEST>")
```

Parameters and Qualifiers

- `<baselineName>`

  Name of the baseline to promote.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the promotion.

- `/STAGE=<promotionStage>`

  Name of the stage to promote the baseline to.

- `/USER_FILENAME=<listFile>`

  A user-specified file containing a list of baselines to be promoted. Allows you to promote multiple baselines in the same operation.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/[NO]DEPLOY`

  Prevents deployment. Cannot be combined with /AREA_LIST.

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

Note that the following formats are not accepted:

- "YYYY-MM-DDTHH24:MI:SSZ" (omission of milliseconds does not work)
- "DD-MM-YYYY HH24:MI:SS"

- /WORKSET

  Specifies a specific project or stream from which to schedule promotion of a baseline. If you do not specify this parameter, the current project or stream is used.

- /SDA_PROCESS=<SDA_Process>

  Specifies the Deployment Automation (DA) application process to run in the environment that is mapped to the stage you are promoting to. Omit this qualifier if you do not want to run an automation during promotion.

- /SDA_DEPLOY_COMPONENT=(<SDA_Component>=<Version name>)

  Specifies a DA process component and component version to be created. The process component is used to deploy promoted baseline items.

  - <SDA_Component>

    Specifies the name of an existing process component.

  - <Version name>

    Specifies the name of a new component version. If the version name already exists it is reused without redeploying items to it. To identify which baseline is mapped to a component version, name the version after the specification of the promoted baseline. May be omitted when a new version is not required.

- /SDA_COMPONENTS=(<SDA_Component1>=<Version name1>, <SDA_Component2>=<Version name2>,...)

  Specifies DA process components, and the corresponding component versions, to be used during DA process execution. Omitted components are not used in the automation execution. Use "<LATEST>" to specify the latest component version.

Description

Use the PMBL command to schedule the promotion of a Dimensions baseline to a lifecycle stage.

Limitations

If the parent product uses the Deployment Automation (DA) deployment model, the following qualifiers are not supported:

- /USER_FILENAME
- /AREA_LIST
- /NODEPLOY

PMI – Promote Item

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
[<itemSpec>|<fileName>]
/COMMENT=<userComment>
/STAGE=<promotionStage>
/WORKSET=<projectName>
/USER_FILENAME=<listFile>
/[NO]QUIET
/AREA_LIST=<areaList>
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
```

Parameters and qualifiers

- `[<itemSpec>|<fileName>]`

  Filename or specification of the item to promote.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the promotion.

- `/STAGE=<promotionStage>`

  Name of the stage to promote the item to.

- `/WORKSET=<projectName>`

  Name of the project or stream that contains the item to promote.

- `/USER_FILENAME=<listFile>`

  A user specified file containing the list of items or files that are to be promoted. Specifying this option allows you to promote many items at once. This option is mutually exclusive to specifying `<itemSpec>|<fileName>`.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/[NO]DEPLOY`

  /NODEPLOY prevents deployment. Cannot be combined with /AREA_LIST.

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  Note that the following formats are not accepted:

  - "YYYY-MM-DDTHH24:MI:SSZ" (omission of milliseconds does not work)
  - "DD-MM-YYYY HH24:MI:SS"

Description

Use the PMI command to schedule the promotion of a Dimensions item to a lifecycle stage, and in turn activate deployment.

PMRQ – Promote Request

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
<requestId>
/COMMENT="<userComment>"
/STAGE="<promotionStage>"
/[NO]CANCEL_TRAVERSE
/WORKSET="<projectName>"
/USER_FILENAME="<listFile>"
/[NO]QUIET
/AREA_LIST="<areaList>"
/[NO]DEPLOY
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
```

Parameters and qualifiers

- `<requestId>`

  ID of the request to promote.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the promotion.

- `/STAGE=<promotionStage>`

  Name of the stage to promote the request to.

- `/[NO]CANCEL_TRAVERSE`

  `CANCEL_TRAVERSE` limits promotion to only the specified request.

- `/WORKSET=<projectName>`

  Name of the project or stream that contains the request to promote. If you do not specify a project or stream, the currently active project or stream will be used.

- `/USER_FILENAME=<listFile>`

  A user specified file containing the list of requests that are to be promoted. Specifying this option allows you to promote many requests at once.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/[NO]DEPLOY`

  /NODEPLOY Prevents deployment to default areas attached to the related project or stream. It cannot be combined with the `/AREA_LIST` option.

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  Note that the following formats are not accepted:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (omission of milliseconds does not work)
  - `"DD-MM-YYYY HH24:MI:SS"`

Description

Use the PMRQ command to schedule the promotion of a Dimensions request to a lifecycle stage. This may then trigger deployment of associated items and dependent requests to default areas or to a list of explicitly specified areas.

PRIV – Manage Privileges

```
<privilege-id>
/RULE=<rule-id>
/ADD or /DELETE or /REPLACE
[/ROLES=(<list-of-role-names>)]
[/USERS=(<list-of-users-and-groups>)]
[/PRODUCT=<product-id>]
[/WORKSET=<project-name>]
[/STAGE=<stage-name>]
[/AREA=<area-name>]
```

Usage

- `PRIV <database-level-privilege-id> /RULE=<rule-id> /USERS=(<list-of-users-and-groups>) {/ADD|/DELETE|/REPLACE}`

  For database-level privileges

- `PRIV <product-level-privilege-id> /PRODUCT=<product-id> /RULE=<product-level-rule-id> {/ADD|/DELETE}`

  For product-level privileges with product-level rules

- `PRIV <product-level-privilege-id> /PRODUCT=<product-id> /RULE=<user-or-group-level-rule-id> /USERS=(<list-of-user-and-group-ids>) {/ADD|/DELETE|/REPLACE}`

  For product-level privileges with user-level or group-level rules

- `PRIV <product-level-privilege-id> /PRODUCT=<product-id> /RULE=<role-level-rule-id> /ROLES=(<list-of-role-names>) {/ADD|/DELETE|/REPLACE}`

  For product-level privileges with role-level rules

- `PRIV <product-level-privilege-id> /WORKSET=<project-name> /RULE=<role-level-rule-id> /ROLES=(<list-of-role-names>) {/ADD|/DELETE|/REPLACE}`

  For product-level privileges with project-level rules, to a specific project

- `PRIV <product-level-privilege-id> /WORKSET=<project-name> /STAGE=<stage-name> /RULE=<role-level-rule-id> /ROLES=(<list-of-role-names>) {/ADD|/DELETE|/REPLACE}`

  For product-level privileges with project-level rules, to a specific stage in a specific project

- `PRIV <product-level-privilege-id> /WORKSET=<project-name> /AREA=<area-name> /RULE=<role-level-rule-id> /ROLES=(<list-of-role-names>) /STAGE=<stage-Name> {/ADD|/DELETE|/REPLACE}`

  For product-level privileges with project-level rules, to a specific deployment area in a specific project

Description

Use the PRIV command to enable or disable the rules for privileges and to assign privileges to or deassign privileges from roles, groups, and users.

There are two types of privilege: base database–level privileges (administrator privileges) and product-level privileges.

There are four types of rules: database-level rules, product-level rules (that is, global rules), user-level or group-level rules, and role-level rules.

Constraints

The PRIV command can be run only by users who have the appropriate privilege management capabilities.

The /WORKSET, /STAGE, and /AREA qualifiers are only valid for deployment or promotion privileges.

Examples

Refer to "Appendix D: Dimensions CM Privileges" in the *Process Configuration Guide* for detailed information on the privileges and their rules, including privilege and rule ID, and a definition of each privilege and each rule.

Example 1

The following example enables the user *jon* to create baselines in the Qlarius product:

```
PRIV BASELINE_CREATE /RULE=user_enable /add /users=("jon")
    /PRODUCT=QLARIUS
```

Example 2

The following example disables the user *jonny* from creating new streams in the Qlarius product:

```
PRIV PROJECT_STREAM_CREATE /rule=user_disable /user=jonny  /add
    /PRODUCT=QLARIUS
```

Example 3

This example enables the users *kim* and *jim* to create products:

```
PRIV ADMIN_CREATE_PRODUCT /RULE=user_enable /add /users=("kim","jim")
```

Example 4

This example enables the user *devuser* to create requests in the Qlarius product:

```
PRIV REQUEST_CREATE /RULE=user_enable /add /users=("devuser")
    /PRODUCT=QLARIUS
```

QUIT – Quit

Description

Alias for EXIT.

RA – Remove Area

```
<area-name>
[/[NO]FORCE]
[/CLEAN]
```

Example

```
RA MY_AREA /CLEAN
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area. Area names must be unique within the base database.

- `/[NO]FORCE`

  Specifying /FORCE means that the area will be removed even if there are associated build areas.

  The default is /NOFORCE, which means that the command will fail if there are associated build areas.

- `/CLEAN`

  Removes VDA records when an area is deleted.

Description

The RA command deletes an area definition. This command does not delete the contents of the area (files or folders) on disk.

Constraints

To delete a work area, you must have the Delete Work Areas privilege. To delete a deployment area, you must have the Delete Deployment Areas privilege.

RABC – Relate Area to Build Configuration

```
RABC <area-name>
/WORKSET=<project-spec>
/BUILD_CONFIG=<configuration-spec>
[/RELATIVE_LOCATION=<relative-path>]
```

Example:

```
RABC build-component1
/WORKSET=build-project-component1
/BUILD_CONFIG=build-config-component1
/RELATIVE_LOCATION=component1
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of a pre-defined build area.

  Only work areas can be related to build configurations that belong to products that use the Deployment Automation deployment model.

- `/WORKSET=<project-spec>`

  Specifies the project to be related the area.

- `/BUILD_CONFIG=<configuration-spec>`

  Specifies the build configuration to be related to the area.

- `/RELATIVE_LOCATION=<relative-path>`

  (Only available for work areas) Specifies the relative path in the area's file system directory to which the project's files are copied. For example, if <relative-path> is `component1` and the area's base directory is `stal-dev-lx1::/work_areas`, a recursive copy operation places all project items into `stal-dev-lx1::/work_areas/component1`.

Description

Enables you to relate an area to a build configuration. For more information about using Dimensions Build see *Dimensions CM Build Tools User's Guide.*

RAI – Remove Archived Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/[NO]CHECK]
[/SELECT_REVISION=LAST_MODIFIED|BRANCH_LATEST]
```

Example

```
RAI "PRODX:DECODER.AAAA-SRC;1"
```

See the *System Administration Guide* for details.

RAMA – Remove Archived Material Selected by Archive

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<archive-id>
[/[NO]CHECK]
```

Example `RAMA ARCH_BL5`

See the *System Administration Guide* for details.

RAMP – Remove Archived Material Selected by Product

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<product-id>
[/NOCHECK]
```

Example `RAMP PRODX`

See the *System Administration Guide* for details.

RAT – Read Archive Tape

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<archive-id>
/DEVICE=<device-id> or /DEVICE=NONE
/TAPE=<tape no.>
/VOLUME=<volume-id>
[/DIRECTORY=<directory>]
```

Example

```
RAT AA12AB /DEVICE="/dev/rmt0h" -
   /TAPE="aa100"/VOLUME="bb100"
```

See the *System Administration Guide* for details.

RAWS – Relate Area to Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
/WORKSET=<project-spec>
[/RELATIVE_LOCATION=<relative-path>]
[/FILTER=<area-filter-name>]
[/[NO]POPULATE]
[/[NO]KEEP]
[/[NO]DEFAULT]
[/SEQUENCE_ORDER=<sequence-order>]
```

Examples

- `RAWS DM10-WIN32 /WORKSET="PVCS:DM10" /RELATIVE_LOCATION="win32"`

  Relates an area to a project or stream. Does not update area *contents*.

- `RAWS DM10-WIN32 /WORKSET="PVCS:DM10" /RELATIVE_LOCATION="windows" /POPULATE`

  Changes the relative location and populates the area. Does not delete existing area files that originated from this project/stream.

- `RAWS DM10-WIN32 /WORKSET="PVCS:DM10" /RELATIVE_LOCATION="windows" /POPULATE /NOKEEP /DEFAULT`

  Changes the relative location, deletes existing area files originating from this project, populates the area with new contents, and sets the relationship as the default for future deployments from this project.

Parameters and qualifiers

- `<area-name>`

  The name of the area that you want to relate to the specified project/stream.

  Only work areas can be related to configurations whose parent product uses the Deployment Automation deployment model.

- `/WORKSET=<project-spec>`

  The project/stream to which you want to relate the specified area.

- `/RELATIVE_LOCATION=<relative-path>`

  Specifies the relative path within the area's file system directory to which this project's files will be deployed. For example, if `<relative-path>` is `"component1"` and the area's directory is `stal-dev-lx1::/deployment_areas`, a recursive deployment operation would copy all project items into `stal-dev-lx1::/deployment_areas/component1`.

NOTES

- Specifying an empty string clears the relative location.
- If there is no relative location, the area's directory is used for deployment.
- The relative location cannot contain `".."`.

- `/FILTER=<area-filter-name>`

  Specifies the set of inclusion/exclusion rules to be associated with this area, when used against this project.

CAUTION! Audit and area filters can easily be confused. See *Correct Use of Area and Audit Filters* in the *Area Definitions* chapter in the *Process Configuration Guide.*

- `/POPULATE or /NOPOPULATE`

  Specifies whether to populate the area with item revisions from this project (and its child collections) given the specified relative path. By default, the area is not populated (/NOPOPULATE is the default). If /POPULATE is specified, files corresponding to item revisions from the specified project (including item revisions in child collections) are transferred into the area one by one (after any deletions caused by /NOKEEP are performed).

- `/KEEP or /NOKEEP`

  Specifies whether to keep files corresponding to the previously transferred item revisions in the area or delete them if the relative path changes. By default, existing area files are not deleted (/KEEP is the default). If /NOKEEP is specified, files corresponding to item revisions from the specified project (including item revisions in child collections) previously transferred to this area are deleted one by one before any re-population caused by /POPULATE.

- `/DEFAULT` or `/NODEFAULT`

  `/DEFAULT` specifies that when an area is related to a project or stream, any items promoted to this stage are automatically deployed.

  If an item is demoted under these conditions then an attempt to undeploy the related change is made. This may or may not succeed depending on other transactions.

  If you do not specify either of these qualifiers /DEFAULT processing is used.

- `/SEQUENCE_ORDER`

  Allows the areas that are associated with a project to be assigned an order in which those areas will be populated. Areas will be sequentially populated based on this number. The string "default" is also a valid value for the <sequence-order> parameter.

  For more information see the *Dimensions CM Deployment Guide*.

Description

Creates or modifies the relationship between an area and the specified project.

If the relationship specified by an RAWS command already exists, the value of /RELATIVE_LOCATION is used to update the relationship. If both /POPULATE and /NOKEEP are specified, /NOKEEP is processed first.

Constraints

Only users with the "Assign Deployment Areas to Project" privilege can run this command.

RBA – Remove Build Area

NOTE This command is no longer available; use RA (Remove Area) instead.

See the RA command.

RBBL – Relate Baseline to Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<child-baseline-spec>
/BASELINE=<parent-baseline-spec>
[/RELATIVE_LOCATION=<relative-path>]
```

Example

```
RBBL <child-baseline-spec> /BASELINE=<parent-baseline-spec>
     /DIRECTORY=<baseline-root>
```

Parameters and qualifiers

- `<child-baseline-spec>`

  Specifies the child baseline in the parent-child relationship.

- `/BASELINE=<parent-baseline-spec>`

  Specifies the parent baseline in the parent-child relationship.

- `/RELATIVE_LOCATION=<relative-path>`

  Specifies the relative path of the file system directory to which the child baseline's top-level directory is mapped with respect to the file system directory to which the parent baseline's top-level directory is mapped. For example, if `/RELATIVE_LOCATION` is `"../component1"` and the top-level directory of the parent baseline is mapped to `"/raid1/home/pjwr/work/project/main"`, a recursive `get` operation would map the top-level directory of the child baseline to `"/raid1/home/pjwr/work/project/component1"`.

Description

Creates or modifies a parent-child relationship between the specified baselines. If there is no relative location, the user must specify a value for /DIRECTORY; otherwise, item operations involving the content of the baseline will fail with an error.

If a relationship already exists, the values of /INFO, /RELATIVE_LOCATION, and /DIRECTORY are used to update the relationship. If both /RELATIVE_LOCATION and /DIRECTORY are cleared, a warning is issued.

This command is intended primarily to allow component baselines within a larger release baseline to be updated. The parent baseline must be at the initial lifecycle state for the relationship to be created or for any attribute of the relationship to be changed other than /DIRECTORY.

Constraints

Only users with the appropriate management privileges can run this command.

RBCD – Relate Baselines to Requests

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/AFFECTED or /IN_RESPONSE_TO or INFO]
```

Example

```
RBCD "PAYROLL:ACME_2.1" -
/CHANGE_DOC=("PAYROLL_TDR_1","PAYROLL_TDR_2")
/INFO
```

Parameters and qualifiers

- `<baseline-spec>` comprises:

  `<product-id>:<baseline-id>`

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>` identifies a request to which the specified baseline is to be related.

- `/AFFECTED or /IN_RESPONSE_TO or /INFO`

  Specifies the type of relation to be set up between the given baseline and the associated requests. The qualifiers are mutually exclusive.

  The default is `/AFFECTED`.

Constraints

RBCD is restricted to merge, release, and revised baselines. If it is run with respect to an archive or design baseline, then an appropriate error will be returned.

RBCD will only work successfully if you have both the baseline and requests in your pending list. If you specify an `/INFO` relationship, however, then this pending list restriction is relaxed.

There is no support for phase rules or change management rule enhancements within the context of baseline to request relationships.

Only the three relationship types – Info, Affected and In Response To – are supported. There is no support for user-defined relationship types.

RBPROJ – Delete a Dimensions Build Project

NOTE This command is no longer available. Use Dimensions Build to delete a build project.

```
Command no longer available.
```

RBWS – Relate Baseline to Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
/WORKSET=<project-spec>
[/RELATIVE_LOCATION=<relative-path]
```

Example

```
RBWS <child-baseline-spec> /WORKSET=<parent-project-spec>
```

Parameters and qualifiers

- `<baseline-spec>`

  Specifies the child baseline in the parent-child relationship.

- `/WORKSET=<project-spec>`

  Specifies the parent project in the parent-child relationship.

- `/RELATIVE_LOCATION=<relative-path`

  Specifies the relative path of the file system directory to which the child baseline's top-level directory is mapped with respect to the file system directory to which the parent project's top-level directory is mapped.

NOTES

– Specifying an empty string clears the relationship-level baseline root for the user.

– If there is no relationship-level baseline root, /RELATIVE_LOCATION is used.

– If the path contains `::` (that is, it has the format `<node-or-area>::<path>`, and `<node-or-area>` matches the name of an existing area), the path is interpreted as an offset (relative path) with regard to the area root directory. (This offset may be empty, in which case the file system directory is set to equal the area directory.) Otherwise, standard Dimensions interpretation of the path applies.

Description

Creates or modifies a parent-child relationship between the specified child baseline and the specified parent project.

If a relationship already exists, the value of /RELATIVE_LOCATION is used to update the relationship.

This command is intended primarily to allow component baselines within a larger release baseline to be updated. The parent baseline must be at the initial lifecycle state for the relationship to be created or for any attribute of the relationship to be changed.

Constraints

Only users with the appropriate management privileges can run this command.

RCCD – Relate Requests to Request

```
<request-id>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/RELATIONSHIP=<relname> or /DEPENDENT or /INFO]
```

Example

```
RCCD PROD_CN_4 /CHANGE=PROD_DR_25
```

Parameters and qualifiers

- `<request-id>`

  This is the identity of the request to be the parent in the relationship to be created.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  Specifies the identities of one or more requests to be children in the relationship to be created.

- `/RELATIONSHIP=<relname>`

  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.

  The default is `/RELATIONSHIP=DEPENDENT`.

- `/DEPENDENT` or `/INFO`

  This is equivalent to specifying either of these as settings of the `/RELATIONSHIP` qualifier.

Constraints

This command can be run by users who have a role (any role will suffice) on the product or products owning the specific request concerned. Such users must have the parent request in their Pending List.

However, the user with the role of `PRODUCT-MANAGER` can setup the Process Model to specify that no request relationships can be created for certain request types.

RCDI – Return Request ID

NOTE This command cannot be used for items that belong to a stream.

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[/COMMENT=<comment>]
[/DIRECTORY=<project-directory-filter>]
[/[NO]RECURSIVE]
[/[NO]FORCE_UPDATE]
[/[NO]KEEP]
[/LOGFILE=<log-file>]
[/USER_DIRECTORY=<target-directory>
[/WORKSET=<project>]
[/CONTENT_ENCODING=<file-encoding>]
[/NOMETADATA]
```

Example

```
RCDI PAYROLL_CR_1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/COMMENT=<comment>`

  Specifies a comment associated with this return.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items processed.

- `/RECURSIVE`

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed.

- `/FORCE_UPDATE`

  Forces an item update.

- `/KEEP`

  Specifies that files are not to be deleted when they are checked in.

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/USER_DIRECTORY`

  Specifies the target directory to which files are to be returned.

- `/WORKSET`

  Specifies the project to be processed by this command.

- /CONTENT_ENCODING

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

Description

This command returns all the items that are checked out against a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

RCDWS – Remove Request Items from Project

NOTE This command cannot be used for streams.

```
<request-id>
[/[NO]CANCEL_TRAVERSE]
[/DIRECTORY=<project-directory-filter>]
[/[NO]RECURSIVE]
[/LOGFILE=<log-file>]
[/WORKSET=<project>]
```

Example

```
RCDWS PAYROLL_CR1
```

Parameters and qualifiers

- `<request-id>`

  The name of a Dimensions request.

- `/CANCEL_TRAVERSE`

  By default, all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the request specified.

- `/DIRECTORY`

  Enables you to specify a project directory filter to restrict the number of items processed.

- `/RECURSIVE`

  Used with /DIRECTORY, this specifies that the filter is to be processed recursively; that is, subdirectories are to be processed.

- `/LOGFILE`

  Specifies a local log file to which the command is to divert all messages.

- `/WORKSET`

  Specifies the project to be processed by this command.

Description

This command removes from the current project (or a specified project) all the items that are related to a specified request.

When multiple revisions of the same item are related to requests processed by this command, only the latest is processed.

RCFG – Return (Check In) Build Configuration

```
<configuration-spec>
[/WORKSET=<project-spec>]
[/COMMENT=<user-comment>]
```

Example

```
RCFG component1
/WORKSET=component1
/COMMENT=Updated with new targets
```

Parameters and qualifiers

- `<configuration-spec>`

  Specifies the name of the build configuration to be checked in.

- `/WORKSET=<project-spec>`

  Specifies the project containing the build configuration to be checked in.

  Default: The user's current project.

- `/COMMENT=<user-comment>`

  Specifies a comment. A generated comment is added by default.

Description

Checks in a build configuration and releases the update lock currently being held.

RCI – Report Current Items

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<file-name>
[/NEW or /OLD]
[/PART=<part-spec> or /BASELINE=<baseline-spec>]
[/SORT=<sort>]
[/WORKSET=<project-spec>]
```

NOTE RCI cannot be run from Dimensions for z/OS.

Example

```
RCI sunrm3_01.export
/NEW -
/PART="PROD:RELEASE MANAGEMENT.AAAA"
```

If /NEW *is specified*

Parameters and qualifiers

- <file-name>

  Specifies the name of the export file where the exported structure is to be stored. If specified as ***** (asterisk), then a temporary export file is created which is deleted at the end of the operation.

Either <part-spec> or <baseline-spec> *must* be specified:

- /PART=<part-spec>

  Specifies the top design part of the current product structure which is to be included in the export file.

  It comprises: <product-id>:<part-id>.<variant>;<pcs>

<variant>	**must** be specified, even if only one variant exists.
<pcs>	is ignored; the current PCS is always used.

- /BASELINE=<baseline-spec>

  Specifies the baseline on which the structure to be included in the export file is to be based.

  It comprises: <product-id>:<baseline-id>

NOTE The above descriptions of <file-name>, <part-spec>, and <baseline-spec>, with the /NEW option, are identical for RCI, RCP and RDS commands.

- /SORT=<sort>

  Indicates the report sequence. It should be omitted, as currently the only valid option is IID to list current items in item-id order.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

If /NEW *is omitted*

- <file-name>

  Specifies where an exported structure has previously been stored.

- /OLD

  may be specified, but is optional as this is the default when /NEW is omitted.

- /PART=<part-spec>

  This is normally omitted. If specified, the report is restricted to the items in <part-spec> and any design parts below it in the tree structure, which are also within the exported structure.

- /BASELINE=<baseline-spec>

  This is normally omitted. If specified, the report is restricted to those items which are both within the scope of <baseline-spec> and within the exported structure.

- /SORT=<sort>

  should be omitted (as above for the /NEW option).

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  This qualifier is only meaningful if /NEW is specified.

Constraints

This command can be run only by the user initiating the report who must have a valid role for the top design part in the structure to be reported. In a structure report which includes items, if an item has two or more revisions currently in the same lifecycle state, only the latest (most recently created/updated) of these is shown.

RCP – Report Current Parts

```
<file-name>
[/NEW or /OLD]
[/PART=<part-spec> or /BASELINE=<baseline-spec>]
[/SORT=<sort>]
```

NOTE RCP cannot be run from Dimensions for z/OS.

Example

```
RCP sunrm3_01.export /OLD
```

If /NEW *is specified*

Parameters and qualifiers

- <file-name>

  Specifies the name of the file where the exported structure is to be stored. If specified as * (asterisk), then a temporary file is created which is deleted at the end of the operation.

  If no file name is specified, part_list.out is created by default.

Either <part-spec> or <baseline-spec> *must* be specified:

- /PART=<part-spec>

  Specifies the top design part of the current product structure which is to be included in the export file.

  It comprises: <product-id>:<part-id>.<variant>;<pcs>

  <variant> **must** be specified, even if only one variant exists.

  <pcs> is ignored; the current PCS is always used.

- /BASELINE=<baseline-spec>

  Specifies the baseline on which the structure to be included in the export file is to be based.

  It comprises: <product-id>:<baseline-id>

NOTE The above descriptions of <file-name>, <part-spec>, and <baseline-spec>, with the /NEW option, are identical for RCI, RCP and RDS commands.

- /SORT=<sort>

  Indicates the report sequence. Valid options are:

  - PNO to list current design parts in part number order; or
  - PID to list current design parts in part-id order.

If omitted, it defaults to PID.

If /NEW *is omitted*

- <file-name>

  Specifies where an exported structure has previously been stored.

- /OLD

  may be specified, but is optional as this is the default when /NEW is omitted.

- /PART=<part-spec>

  This is normally omitted. If specified, the report is restricted to those design part which are both within the tree structure specified by <part-spec> and within the exported structure.

- /BASELINE=<baseline-spec>

  This is normally omitted. If specified, the report is restricted to those design parts which are both within the scope of <baseline-spec> and within the exported structure.

- /SORT=<sort>

  Indicates the report sequence (as above for the /NEW option).

Constraints

This command can be run only by the user initiating the report who must have a valid role for the top design part in the structure to be reported.

RCSJ – Relate Command to Schedule Job

```
<job-id>
/CMD
```

Example:

```
RCSJ "MyJobName" /CMD="DPI  @"QLARIUS:4K3TK DBIO VB.A-SRC;1.0@" /
    WORKSET=@"QLARIUS:UW_DOTNET_2.0@" /STAGE=@"SYSTEM TEST@""
```

Parameters and qualifiers

- <job-id>

  Specifies the job-id.

- /CMD

  Specifies the command to be related to the scheduled job.

Description

Enables you to relate a command to a scheduled job.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

RCU – Remove Customer

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<name>
/LOCATION=<location>
/PROJECT=<project-spec>
```

Example

```
RCU "Brown Finances"
/LOCATION="Bristol" -
/PROJECT="PAYROLL"
```

Parameters and qualifiers

- `<name>`

  Specifies the customer's name.

- `/LOCATION=<location>`

  Specifies the customer's physical location.

- `/PROJECT=<project-spec>`

  Specifies the project name.

Description

The Dimensions product allows you to maintain a list of customers and a record of which Dimensions releases have been sent to each customer.

The RCU command enables you to remove a customer's details.

Constraints

The combination of customer name, location, and project-spec must be unique in the Dimensions database.

You cannot remove a customer to whom releases have been forwarded.

RDEL – Delete Jobs from the Job Queue

```
<job-key>
[/[NO]FORCE]
[/AREA]
[/CHANGE_DOC_IDS]
[/DATE=”dd-mmm-yy”]
[/FROM=”dd-mmm-yy”]
[/TO=”dd-mmm-yy”]
[/NETWORK_NODE=nodename]
[/OWNER=userid]
[/STATUS=”status”]
[/TEMPLATE=templatename]
[/USER=remote_userid]
[/RC=return code]
```

Description

Enables you to delete jobs from the job queue. For information about remote job execution see the *System Administration Guide*.

Examples

- `RDEL B-123456`

  Deletes job `B-123456`.

- `RDEL /FROM="14-01-15" /TO="20-03-15"`

  Deletes all jobs from 14th January 2015 to 20th March 2015.

Parameters and qualifiers

- `<job-key>`

  String in the format `<id>-<job-uid>` that identifies the job to be deleted from the job queue.

- `/[NO]FORCE`

  Deletes all jobs from the job queue except those that are at the status FAILED or SUCCEEDED.

  Default: `/NOFORCE`

- `/AREA`

  Specifies an area to which the delete operation is restricted.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  Specifies the request ID(s) required to delete job(s) from streams and projects that are under CM rules.

- /DATE="dd-mmm-yy"

  Specifies a date in this format: 03-JAN-15

- /FROM="dd-mmm-yy" or /FROM=-n

  Specifies the start of a date range. You can also delete all jobs from a previous number of days. For example, to delete all jobs from the previous 10 days:

  /FROM=-10

- /TO="dd-mmm-yy" or /TO=-n

  Specifies the end of a date range. You can also delete all jobs that are older than a specific number of days. For example, to delete all jobs that are older than two days:

  /TO=-2

- /NETWORK_NODE=nodename

  Specifies the network node where the logs are located.

- /OWNER=userid

  Specifies a Dimensions CM user ID that created the logs.

- /STATUS="status"

  Specifies the status of the jobs.

- /TEMPLATE=templatename

  Specifies the template that was used to create the logs.

- /USER=remote_userid

  Specifies a user ID for the remote node.

- /RC=return code

  Only deletes jobs that match the specified return code.

NOTE /OWNER, /STATUS, /TEMPLATE, and /USER can include the Oracle wildcard '%' and use the LIKE operator to search.

RDS – Report Design Structure

```
<file-name>
[/NEW or /OLD]
[/PART=<part-spec> or /BASELINE=<baseline-spec>]
[/SORT=<sort>]
[/LEVEL=<level>]
[/STRUCTURE=(<option>,...)]
[/WORKSET=<project-spec>]
```

NOTE RDS cannot be run from Dimensions for z/OS.

Example

```
RDS sunrm3_01.export /STRUCTURE=ALL
```

If /NEW *is specified*

Parameters and qualifiers

- <file-name>

  Specifies the name of the file where the exported structure is to be stored. If specified as * (asterisk), then a temporary file is created which is deleted at the end of the operation.

  If no file name is specified, design_structure.out is created by default.

Either <part-spec> or <baseline-spec> *must* be specified:

- /PART=<part-spec>

  Specifies the top design part of the current product structure which is to be included in the export file.

  It comprises: <product-id>:<part-id>.<variant>;<pcs>

  <variant> **must** be specified, even if only one variant exists.

  <pcs> is ignored; the current PCS is always used.

- /BASELINE=<baseline-spec>

  Specifies the baseline on which the structure to be included in the export file is to be based.

  It comprises: <product-id>:<baseline-id>

NOTE The above descriptions of <file-name>, <part-spec>, and <baseline-spec>, with the /NEW option, are identical for RCI, RCP and RDS commands.

- /SORT=<sort>

  Indicates the report sequence. Valid options are:

  - PNO to list current design parts in part number order at each structural level.
  - PID to list current design parts in part-id order at each structural level.

  If omitted, it defaults to PID.

- /LEVEL=<level>

  Specifies the number of levels of the structure which are to be reported, working downwards either from <part-spec> if specified, or otherwise from the highest-level design part in the tree structure which is being reported on.

  All levels of that structure are included if this parameter is omitted.

- /STRUCTURE=<option>

  Specifies what contents of the design part structure are to be included in the report:

  - USAGE to include USED design parts.
  - ITEM to include items.
  - include requests.
  - ROLE to include user roles.
  - ALL to include all options.

  By default, i.e. if /STRUCTURE is not specified, only the structure of BREAKDOWN design parts is reported.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

If /NEW *is omitted*

- <file-name>

  Specifies where an exported structure has previously been stored.

- /OLD

  may be specified, but is optional as this is the default when /NEW is omitted.

- /PART=<part-spec>

  This is normally omitted. If specified, the report is restricted to those design parts (and items if specified - see <option>) which are both within the tree structure specified by <part-spec> and within the exported structure.

- /BASELINE=<baseline-spec>

  This is normally omitted. If specified, the report is restricted to those design parts (and items if specified - see <option>) which are both within the scope of <baseline-spec> and within the exported structure.

- `/SORT=<sort>`

  as above for the `/NEW` option.

- `/LEVEL=<level>`

  as above for the `/NEW` option.

- `/STRUCTURE=<option>`

  as above for the `/NEW` option.

- `/WORKSET=<project-spec>`

  This qualifier is only meaningful if `/NEW` is specified.

Constraints

This command can be run only by the user initiating the report who must have a valid role for the top design part in the structure to be reported.

REL – Release

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<release-spec>
/BASELINE=<baseline-spec>
[/TEMPLATE_ID=<template-id>]
[/DESCRIPTION=<description>]
[/DIRECTORY=<directory>]
[/FILENAME=<file-name>]
[/[NO]DELTA /PREV_RELEASE=<release-id>]
[/[NO]EXPAND]
[/[NO]REEXPAND]
[/[NO]TOUCH]
[/[NO]OVERWRITE]
[/NOMETADATA]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
```

NOTE When a release is created, Dimensions records the target directory and the person who was responsible for creating the release. You can obtain this information through the published view 'PCMS_RELEASE_DATA' (for details of this view, see the *Reports Guide*).

Example

```
REL PROD:"R M 2.0 FOR HP" -
/BASE=PROD:"R M VERSION 2 FOR HP"
```

Parameters and qualifiers

- `<release-spec>`

  Comprises:

  `<product-id>:<release-id>`

- `/BASELINE=<baseline-spec>`

  Comprises:

  `<product-id>:<baseline-id>`

  `<product-id>` *must be the same as that for the* `<release-spec>`*.*

- `/TEMPLATE=<template-id>`

  Specifies a release template to be used for this release.

- `/DESCRIPTION=<description>`

  This is a description of the release.

- `/DIRECTORY=<directory>`

  Specifies the top level directory where the release is to be stored.

  If omitted, it defaults to the current directory.

NOTE On a UNIX installation where the Dimensions System Administrator user-id is not the default `dmsys`, you will need to include the node name in the directory specification, for example, `/DIRECTORY="hp6::/tmp/project/rel1"`

- /FILENAME=<file-name>

  Generates command script to <file-name> but does not run the command

- /DELTA

  Specifies a delta release is to be created; that is, items that are identical to the /PREV_RELEASE are not exported.

NOTE /PREV_RELEASE comprises <release-id> **not** <release-spec>. For example, if you wish to release PROD:REL_2 based on a previous release of PROD:REL_1, then the /DELTA part of the REL syntax would be:
REL "PROD:REL_2"... /DELTA /PREV_RELEASE="REL_1"
You **do not** specify PROD for /PREV_RELEASE.

- /[NO]EXPAND
  /[NO]REEXPAND
  /[NO]TOUCH
  /[NO]OVERWRITE

  Adds the specified qualifier to each FI command in the script generated by REL.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when downloading text files. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Notes

REL processes the content of child baselines when the relationship specifies a relative location. Each child baseline for which no relative location exists receives a diagnostic indicating that the child baseline was not processed and why.

Constraints

This command can be run by users who can 'get the items' that compose the release. In practice this usually means being assigned one or more roles whose scope is at least either for the top design part in the baseline used, extending to all of the design structure segment below that design part, or for the project specified when that Baseline was created, or for both.

If you are going to be creating several releases of (varying aspects or segments of) a particular product, it will be simpler if you are assigned a role for the product-level design part, so that it extends to every design part, and therefore to every item, in the product. (Even so, you might sometimes need some permissions additional to this, whenever the baseline used, and thus the release made from it, contains some foreign Items – i.e. Items that do not belong to the baselined / released product.)

The baseline used must be of release mode (i.e. it cannot contain more than one revision of any one Item). This means that revised baselines and merged baselines also qualify as release mode, as well as those of release mode created using a baseline template; but that baselines of design and archive modes do not qualify.

If a release is created using a release template, that template then cannot be altered (until all releases that used that template have been deleted). This is to ensure that each new release operation is repeatable. (Another release template based on this one can be created and then altered, if such a template is needed for some later release.)

Although this operation will place files in the release directory, automatically creating sub-directories if and as required, it will do so only if the operating system where the release directory is located would have permitted you to create such files and subdirectories yourself.

RENAME – Rename a Product, Baseline, Design Part, Project, or Item

```
[/PRODUCT=<old-product-spec>] /NEW_ID=<new-product-id>
or
[/BASELINE=<old-baseline-spec>] /NEW_ID=<new-baseline-id>
or
[/PART=<product-id:old-part-spec>] /NEW_ID=<new-part-id>
or
[/WORKSET=<old-project-spec>] /NEW_ID=<new-project-id>
or
[/ITEMID=<old-item-spec>] /NEW_ID=<new-item-id>
```

Examples

```
RENAME
/BASELINE=MINAKO:RELEASE_1.0
/NEW=RELEASE_DUFF
```

```
RENAME
/WORKSET=MINAKO:DEVELOPMENT_WORK
/NEW=SOURCE_CODE
```

Parameters and qualifiers

- `<old-product-spec>`

  Specifies the old product name

- `<new-product-id>`

  Specifies the new product name.

- `<old-baseline-spec>`

  Specifies the old, full baseline specification.

- `<new-baseline-id>`

  Specifies the new baseline-id. This is the name of the baseline only, not the full specification.

- `<old-part-spec>`

  Specifies the old, full part specification.

- `<new-part-id>`

  Specifies the new part-id. This is the name of the part only, not the full specification.

- `<old-project-spec>`

  Specifies the old, full project specification.

- `<new-project-id>`

  Specifies the new project-id. This is the name of the project only, not the full specification.

- `<old-item-spec>`

  Specifies the old, full item specification.

- `<new-project-id>`

  Specifies the new item-id. This is the name of the item only, not the full specification.

Description

The "product" version of this command enables an existing product to be renamed.

The "baseline" version of this command enables an existing baseline to be renamed within the context of the same product.

The "design part" version of this command enables an existing design part to be renamed within the context of the same product.

The "project" version of this command enables an existing project to be renamed within the context of the same product.

The "item" version of this command enables an existing item to be renamed within the context of the same product.

Constraints

The "product" version of this command can be run only by a user with the appropriate management privileges on the project.

The "baseline" version of this command can be run only by a user with the appropriate management privileges on the baseline.

The "design part" version of this command can be run only by a user with the appropriate management privileges on the design part.

The "project" version of this command can only be run by a user with the appropriate management privileges on the project or the owner of the project (the user who created the project).

REQC – Request Dimensions Request

```
<request-id>
[/CANCEL]
```

Examples

```
REQC "PAYROLL_TDR_1"
REQC "PAYROLL_TDR_1" /CANCEL
```

- `<request-id>`

  Identifies the request.

- `/CANCEL`

  cancel an existing issue replication "request" for a request. This is only valid if executed by the user who made the original request or by a user with the role of CHANGE-MANAGER or PRODUCT-MANAGER.

Description

Upon running the REQC command, the request for the request is logged locally until the next scheduled replication occurs. When the replication actually happens, all the requests for requests are processed by the site that currently owns them and they are reassigned to those sites that requested them. If multiple sites have requested the same request, then this reassignment is done on a first-come-first-served basis, with the other requests being returned with an appropriate error message.

When the request has been processed as a result of replication, the result of the request – whether it was accepted or rejected – will automatically be e-mailed to the request requester. This e-mail will notify the user whether or not they can now work on that request.

Constraints

1 To successfully run REQC you must either:

- have the requested request in your pending list and have valid privileges for the work that you intend to undertake, or
- be a user with the appropriate management privileges.

2 A request will only succeed if the request in question is not currently being worked on. This means that if any items are checked out against that request then the request will be denied. This will also be the case for any requests that have been locked using the /EXCLUSIVE_LOCK qualifier of the CC or UC commands, see pages 90and 461 respectively.

REXEC – Execute a Job on a Network Node

```
/NETWORK_NODE=<nodename>
/TEMPLATE_ID=<template-name>
[/[NO]BATCH]
[/[NO]CAPTURE]
[/USER=<userid or credential-set-name>]
[/PASSWORD=<password>]
[/PARAMETERS=(<name1=value1,name2=value2,...,nameN=valueN>)]
[/DESCRIPTION=<text>]
[/EXECUTION_DIRECTORY=<directory>]
[/PRESERVE]
[/[NO]SHOW]
[/USER_FILENAME=[node::]filename]
[/[NO]LOCK]
```

Description

Remote job execution (REXEC) executes a job on a tertiary node and records the job in the job queue. The output prints the job key in this format:

`R-<job-uid>`

If you specify `/BATCH` the job is executed asynchronously and its status is set to SUBMITTED in the job queue. You can use the `RSTAT` command to update the status of a batch job, and use the command `RLIST /WAIT` to wait until a job finishes.

If you do not specify `/BATCH` the job is executed synchronously. After execution is complete the job status is set to FAILED or SUCCEEDED.

For more information about remote job execution see the *System Administration Guide*.

Parameters and Qualifiers

- `/NETWORK_NODE=<nodename>`

  Specifies the Dimensions CM network node on which to run the job.

NOTE If an `AUTH` command was previously issued against the node specified by `/NETWORK_NODE`, the same credentials are reused.

- `/TEMPLATE_ID=<template-name>`

  Specifies the template name to be used to create the job. Maps to the template file located in the tertiary node directory specified by the value of the `DM_TEMPLATE_CATALOG<n>` symbols. Normally these symbols are defined in the `dm.cfg` file on the network node. If this symbol is not defined these templates are used:

  Windows: `%DM_ROOT%templates\<template-name>`

  UNIX and z/OS: `$DM_ROOT/templates/<template-name>`

- /BATCH

  Runs the remote job asynchronously.

- /CAPTURE

  Generates a one-time certificate for reconnecting to the remote node.

- /PARAMETERS=(<name1=value1,name2=value2,...,
  nameN=valueN>)

  Specifies a list of comma separated keyword and values, or an arrays of values, to be passed into the template being executed.

  Names in lowercase are converted to uppercase during execution. Names in templates must be written in uppercase, for example: %NAME1. %NAME2

- /DESCRIPTION

  A description of the job.

- /USER=<userid or credential-set-name>

  Specifies the user-id, or credential set name, to be used to execute the job on the network node. For more information about credential sets see the *System Administration Guide*.

- /PASSWORD=<password>

  Specifies the password to be used to execute the job on the network node. Not required if you specify a credential set name in the /USER parameter.

- /EXECUTION_DIRECTORY=<directory>

  Specifies the directory in which the remote command is to be started.

- /PRESERVE

  Preserves the job's log as an item in Dimensions CM. Specify one of these parameters:

  - YES: preserve the log.
  - ONERROR: only preserve the log if there is a job error.
  - ONSUCCESS: only preserve the log if the job succeeds.
  - NO: do not preserve the log.

  Dimensions CM administrators can specify the location where logs are preserved in CM. For details see the *System Administration Guide*.

- /[NO]SHOW

  Displays the log file in your current session.

  Default: /SHOW

- /USER_FILENAME=[node::]filename

  Saves the job's log file in the path and folder that you specify. Can be used in conjunction with /PRESERVE.

- /[NO]LOCK

  Locks the job so that it cannot be deleted.

  Default: /NOLOCK

RI – Return (Check In) Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/USER_FILENAME=<user-filename>]
[/[NO]KEEP]
[/STATUS=<status>]
[/COMMENT=<comment text>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/WORKSET=<project-spec>]
[/[NO]FORCE_UPDATE]
[/[NO]CANCEL_UNCHANGED]
[/CODEPAGE=<code-page>|DEFAULT]
[/CONTENT_ENCODING=<file-encoding>]
```

Example

```
RI PROD:"QUERY RELEASE".AAAA-SRC;1 -
/KEEP /STATUS="UNDER TEST"" -
/COMMENT="checked in for CRB 91"
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if you have checked out only one revision of this item.

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- /USER_FILENAME=<user-filename>

  Specifies the file in the user-area from which the item will be copied.

  It may be omitted if the file has the same name as had been given to the user-area file when this item was checked out (EI command).

- /KEEP

  Specifies that the user area file, which is normally deleted after its data has been placed under Dimensions 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.

- /STATUS=<status>

  Specifies a valid changed status (lifecycle state) for the checked in revision.

NOTE The only equivalent to this parameter in GUI mode is RI followed by AI. The status, if specified, must be one which would be valid if AI had been used separately. If omitted, the revision remains at the initial state (in the lifecycle defined for <item-type>).

- /COMMENT=<comment text>

  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.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

  <attrN> is 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.

- /WORKSET=<project-spec> comprises:

  <product-id>:<project-id>

  and is optional (subject to the following). 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). Therefore, if your current project is not that project, either this qualifier must be used to specify the correct check in <project-spec> or your current project must be set to that <project-spec> using the Set Current Project (SCWS) command.

- /FORCE_UPDATE

  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.

- /CANCEL_UNCHANGED or /NOCANCEL_UNCHANGED

  /CANCEL_UNCHANGED performs a CIU (Cancel Item Update) command if the user file does not differ from the base revision and Dimensions is configured to allow updates only if a real change is made.

- /CODEPAGE=<code-page>|DEFAULT

  Specifies the *code page* to be associated with the item. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
  /CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

  For further details concerning code pages and logical nodes see the *Network User's Guide*.

  You are advised to let the parameter default to the code page for the item type or the platform.

  The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.

- <file-encoding>

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

NOTE RI results in the creation of a new revision in the project. If DEVELOPMENT deployment areas are in use, this command automatically updates the corresponding areas.

Constraints

This command can be run only by the user who previously checked out the item revision or by a user with the appropriate management privileges.

RICD – Relate Item to Requests

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/AFFECTED or /IN_RESPONSE_TO or /INFO]
[/WORKSET=<project-spec>]
```

Example

```
RICD PROD:"QUERY RELEASE".AAAA-SRC;1 /CHANGE_DOC=PROD_DR_25
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

  `<item-id>` may be omitted if `<file-name>` is specified.

  `<variant>` may be omitted if only one exists.

  `<revision>` defaults to the latest revision (see About the Command-Line Interface on page 12).

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>` identifies a request to which the specified item is to be related.

- `/AFFECTED or /IN_RESPONSE_TO or /INFO`

  Specifies the type of relation to be set up between the given item and the requests. The qualifiers are mutually exclusive.

  The default is `/AFFECTED`.

- `/WORKSET=<project-spec>` comprises:

  ```
  <product-id>:<project-id>
  ```

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Constraints

This command can be run only by a user who has the request in his/her pending list or who has the appropriate management privileges.

You cannot relate an item belonging to one product to a request that belongs to a different product.

RII – Relate Item to Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<src-item-spec>
<dst-item-spec>
/RELATIONSHIP=<relationship-id>
[/FILENAME=<src-filename>]
[/FILENAME=<dst-filename>]
[/COMMENT=<comment-text>]
[/WORKSET=<project-spec>]
```

Example

```
RII "PROD_X:INTERFACE_C.AAAA-C;1" -
"PROD_X:INTERFACE_H.AAAA-H;1" /RELATIONSHIP="INCLUDES" –
/COMMENT="INCLUDED HEADER"
```

Parameters and qualifiers

- `<src-item-spec>`

  Specifies the source item from which the relationship instance will start. Partial specification is acceptable only when the filename is specified.

- `<dst-item-spec>`

  Specifies the destination item to which the relationship will be made. Partial specification is acceptable only when the filename is specified.

- `/RELATIONSHIP=<relationship-id>`

  Specifies the relationship type as defined by the DIR command.

  Note that BLD_ACTUAL and BLD_PREDICTED are not allowed. See Constraints below.

- `/FILENAME=<src-filename>`
  `/FILENAME=<dst-filename>`

  Specify the names of the project file names.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/COMMENT=<comment-text>`

  This is an optional qualifier and specifies a comment to be attached to the relationship instance.

- `/WORKSET=<project-spec>` `comprises:`

  `<product-id>:<project-id>`

  this optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Description

The RII command is used to create instances of relationships defined by the DIR command. The source and destination item types must be consistent with the relationship definition.

(The XII command (on page 531) is used to remove such relationships.)

Constraints

This command can be run only by a user who has the items in their Pending list or by a user with the appropriate management privileges.

You cannot use this command to manually create build relationships, as these have to be created by Build. So BLD_ACTUAL and BLD_PREDICTED are not allowed for the / RELATIIONSHIP qualifier.

RIP – Relate Item to Part

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/PART=<part-spec>
[/WORKSET=<project-spec>]
```

Example

```
RIP PROD:"QUERY RELEASE".AAAA-SRC /PART=PROD:"RELEASE MANAGEMENT".IBM
```

Parameters and qualifiers

- `<item-spec>` comprises:

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

  `<item-id>`

  `<variant>` may be omitted if only one exists.

  `<revision>` is ignored; all revisions are related to the specified design part.

- `/FILENAME=<file-name>`

  Optionally specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

- `/PART=<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

  `<variant>` may be omitted if only one exists.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Constraints

This command can be run only by a user who has the item revision in their pending list or by a user with the appropriate management privileges.

RIR – Remove Item Relation Definition

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<relationship-id>
/SRC_TYPE=<product-id>:<item-type-name>
/DST_TYPE=<product-id>:<item-type-name>
```

Example

```
RIR "INCLUDES" -
/SRC_TYPE="PROD_X:C" -
/DST_TYPE="PROD_X:H"
```

Parameters and qualifiers

- `<relationship-id>`

  Specifies the identifier of the relationship to be removed

- `/SRC_TYPE=<prod-id>:<item-type-name>`

  Specifies the source product and item type from which the link will start

- `/DST_TYPE=<prod-id>:<item-type-name>`

  Specifies the destination product and item type at which the link will end or point.

Description

The RIR command is used to remove an item-to-item relationship definition that was defined by the DIR command.

Constraints

Only users with the appropriate management privileges can run this command.

RIWS – Remove Item Revision from Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
/WORKSET=<project-spec>
[/FILENAME=<file-name>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/USER_ITEMLIST="item list path"]
```

Example

```
RIWS PROD_X:"HELLO WORLD".AAAA-SRC;2.6 /WORKSET=PROD_X:"WS DVLP"
```

Parameters and qualifiers

- `<item-spec>`

  Comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if `<file-name>` is specified.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This specifies the project from which the item revision is to be removed.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request which will be used to track this structural change to the project.

  Specify this optional qualifier if you want the change (i.e. item addition) to the project to be recorded against the specified request(s). If path control has been enabled, this qualifier is mandatory.

- `/USER_ITEMLIST="item list path"`

  Specify this qualifier to export or remove multiple items and to submit a single deployment job for all of the specified items. For example:

  `/USER_ITEMLIST="C:\itemlist.txt"`

The item list file has the following format:

```
"item spec1" "ws_filename1"
"item spec2" "ws_filename2"
"item specN" "ws_filenameN"
```

Notes:

- You can omit the "ws_filename" column.
- You do not need to specify <itemSpec>.
- /FILENAME and /WS_FILENAME are ignored.

Description

This command will remove the item specified from the given project. The specified item and project must exist. If the specified item is not in the project a warning will be given.

NOTE Whenever a new revision is added to a project, its stage is reset to DEVELOPMENT, and associated deployment areas and library cache areas are updated.

Constraints

Users must have been granted the privileges:

- `deploy item to next stage`
- `deploy item to any stage`

This constraint can be relaxed using the Set Project Permissions (SWSP) command, see page 451.

RLCA – Remove Library Cache Area

NOTE To delete a library cache area definition you must have a Delete Library Cache Areas definition.

```
<area-name>
```

Example

```
RA <area-name>
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the library cache area. Area names must be unique within the base database.

Description

The RLCA command deletes a library cache area definition. This command does not delete the contents of the area (files or folders) on disk.

Constraints

To delete an area definition, you must have the DELETE_AREA_PRIV privilege. It must be associated with the `object_owned_by_user` rule; that is, in order to delete an area definition, the user must be the owner of the area or a member of the group that is the owner of the area.

RLIST – Lists Jobs in the Job Queue

```
<job-key>
[/USER=<userid>]
[/NETWORK_NODE=<nodename>
[/DATE<date>]
[/TEMPLATE_ID=<template-name>
[/STATUS=<status>]
[/RC=xx]
[/[NO]DETAIL]
[/[NO]WAIT]
[/AREA]
[/[NO]SHOW]
[/USER_FILENAME=<path>]
[/[NO]FORCE]
[/CHANGE_DOC_IDS]
[/FROM=”dd-mmm-yy”]
[/TO=”dd-mmm-yy”]
[/NETWORK_NODE=nodename]
[/OWNER=userid]
[/USER=remote_userid]
```

Description

Lists jobs in the job queue. For information about remote job execution see the *System Administration Guide*.

Parameters and Qualifiers

NOTE Any of the qualifier values described below may contain the % character to enable pattern matching in SQL.

- `<job-key>`

  String in the format `R-<job-uid>` that identifies the job to list in the job queue. To list all jobs in the queue execute RLIST with no parameters.

- `/USER=<userid>`

  Searches for jobs that match the user name that you specify.

- `/NETWORK_NODE=<nodename>`

  Searches for jobs that match the network node name that you specify.

- `/DATE=<date>`

  Searches for jobs that match the date that you specify. Must be in the format `DD-MTH-YY`, for example: `03-FEB-04`

- /TEMPLATE_ID=<template-name>

  Searches for jobs that match the template name that you specify. The template name maps to the template file located in the tertiary node directory specified by the value of the DM_TEMPLATE_CATALOG<n> symbols. Normally these symbols are defined in the dm.cfg file on the network node. If this symbol is not defined these templates are used:

  Windows: %DM_ROOT%templates\<template-name>

  UNIX and z/OS: $DM_ROOT/templates/<template-name>

- /STATUS=<status>

  Searches for jobs that match the status that you specify. Can be FAILED, SUCCEEDED, or SUBMITTED.

- /RC=xx

  Only searches for jobs that match the return code that you specify.

- /[NO]DETAIL

  Provides detailed job information in the list.

- /[NO]WAIT

  Waits for the job being queried to finish. /WAIT is only accepted if the display is for a single job.

  Default: /NOWAIT

- /AREA

  Specifies an area to which the operation is restricted.

- /[NO]SHOW

  Spools the log file to the user's session.

  Default: /SHOW

- /USER_FILENAME=<path>

  Puts the log file in the location that you specify on the client node.

- /[NO]FORCE

  Searches all jobs from the job queue except those that are at the status FAILED or SUCCEEDED.

  Default: /NOFORCE

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  Specifies the request ID(s) required to search for streams and projects that are under CM rules.

- /DATE="dd-mmm-yy"

  Specifies a date in this format: 03-JAN-15

- /FROM="dd-mmm-yy" or /FROM=-n

  Specifies the start of a date range. You can also search for all jobs from a previous number of days. For example, to search for all jobs from the previous 10 days:

  /FROM=-10

- /TO="dd-mmm-yy" or /TO=-n

  Specifies the end of a date range. You can also search for all jobs that are older than a specific number of days. For example, to search for all jobs that are older than two days:

  /TO=-2

- /NETWORK_NODE=nodename

  Specifies the network node where the logs are located.

- /OWNER=userid

  Specifies a Dimensions CM user that created the logs.

- /STATUS="status"

  Specifies the status of the jobs.

- /USER=remote_userid

  Specifies a user on the remote node.

NOTE /OWNER, /STATUS, /TEMPLATE, and /USER can include the Oracle wildcard '%' and use the LIKE operator to search.

RMDF – Remove Data Formats

```
<format>
```

Example

```
RMDF TXT
```

Parameters and qualifiers

- `<format>`

  the format definition being removed.

Description

The RMDF command removes existing defined item or request type data formats. For a discussion on item or request type data formats refer to the DDF command (see page 153).

Constraints

Only users with the appropriate management privileges can run this command.

RMVB – Remove Version Branch

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<branch-id>
```

Example

```
RMVB MAINT
```

Parameters and qualifiers

- `<branch-id>`

  Identifies a branch that was defined by the DVB command. If the branch is used to label any item pedigree trees it may **not** be removed.

Description

The RMVB command removes existing defined version branch-ids.

Constraints

Only users with the appropriate management privileges can run this command.

ROA – Retrieve Offline Archive

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<archive-id>
[/DIRECTORY=<directory>]
```

Example `ROA AA12AB`

See the *System Administration Guide* for details.

RP – Relate Design Part

```
<part-spec>
/FATHER_PART=<parent-part-spec>
```

Example

```
RP PROD:"LIBRARY CONTROL".AAAA /FATHER_PART=PROD:"RELEASE
    MANAGEMENT".AABB
```

Parameters and qualifiers

- `<part-spec>`

  (both for the child design part and its new usage parent part[1]) comprises:

  `<prod-id>:<part-id>.<variant>;<pcs>`

  `<variant>` may be omitted if only one variant of that design part exists.

  `<pcs>` is ignored, the current PCS is always used.

Constraints

Only users with the appropriate management privileges can run this command.

RPCD – Relate Part to Requests

```
<part-spec>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/PENDING]
```

Example

```
RPCD PROD:"RELEASE MANAGEMENT".AAAA -
/CHANGE_DOC=(PROD_DR_25,PROD_DR_26)
```

Parameters and qualifiers

- `<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	is the identity of a request to which the specified design part is to be related.

- `/PENDING`

  This option runs the PEND command after the completion of the RPCD command.

Constraints

- This command can be run only by the user who has the request in their Pending list or by a user with the appropriate management privileges.
- This command cannot be used with the secondary catalog list.
- This command cannot be run if the request is at its end (closed) lifecycle state – even by a change-manager; however, a change-manager can action it back to an earlier lifecycle state perform the relate operation, and then action the request back to its closed state.

RPCP – Report Product Control Plan (Process Model)

```
<product-id>
[/[NO]DETAIL]
[/SECTION=(<section>,...)]
```

NOTE RPCP cannot be run from Dimensions for z/OS.

Example

```
RPCP PROD /DETAIL /SECTION=(ITEM,CDOC)
```

Parameters and qualifiers

- `<product-id>`

  Specifies the product which is to be reported.

- `/DETAIL`

  means print full details of the process model (formerly control plan) sections requested.

  If omitted, the report defaults to summary lists, except for RULE which is given in full.

- `/SECTION=(<section>,...)`

  Specifies which sections of the process model are to be reported. The following sections can be requested:

  - PCAT: reports on design part categories.
  - ITEM: reports on item types.
  - CDOC: reports on request types.
  - ROLE: reports on roles, users, privileges, and groups.
  - RULE: reports on change management rules.
  - ALL: (or if `/SECTION` is not specified) reports on all sections of the process model.

NOTE RPCP creates the report, with file name `cntl_plan.out`, in the current working directory. If you run the RPCP from Windows `Serena | Dimensions 14.4 | Command Client`, the current working directory will be `C:\Documents and Settings\<username>`.

Constraints

Only users with the appropriate management privileges can run this command.

RPNO – Report Part Numbers

```
<product-id>
```

Example

```
RPNO PROD
```

Parameters and qualifiers

- `<product-id>`

  Specifies for which product a part numbers report is to be produced. If specified as ***** (asterisk), then the report covers part numbers for all products in the database.

NOTE RPNO creates the report, with file name `pcms_partno.out`, in the current working directory. If you run the RPNO from the Windows Serena | Dimensions | Command Client, the current working directory will be `%DM_ROOT%\prog`.

Constraints

Only users with the appropriate management privileges can run this command.

RPROJ – Remove a Dimensions Project

NOTE This command is no longer available. Use RWS instead.

RPT – Report Requests

```
<product-id>
<request-category>
/NAME=<report-type>
/CATALOGUE or /CATALOGUE/SECONDARY or /PENDING
/FILENAME=<outfile>
[/ATTRIBUTES=(<attr1>=<string1>,<attr2>=<string2>,...)]
[/CHANGE_DOC_ID=<ch-doc-id>]
[/CH_DOC_TYPE=<ch-doc-type>]
[/PART=<part-spec>]
[/PHASE=<phase>]
[/STATUS=<request-status>]
[/USER=<user-name>]
[/FROM=<date-from>]
[/TO=<date-to>]
[/[NO]HEADING_WRAP]
[/[NO]WRAP]
[/[NO]INDENT]
[/[NO]DETAIL]
[/[NO]HOLD]
[/[NO]PRINT]
```

Example

```
RPT PROD 4 /NAME=CH_DOC_LIST /CATALOGUE /DETAIL -
/FILENAME=workpack_all.list /PRINT
```

Parameters and qualifiers

- `<product-id>`

  This is the product for which a report is required.

- `<request-category>`

  This is the request category for all requests to appear in the report. Valid values for this indicator are 1, 2, 3, or 4. Category 1 is for bugs or problems, category 2 is for change or enhancement requests, category 3 is for "other" (miscellaneous), and category 4 is reserved for work packages.

  The value may be specified as **%** (percent sign) to permit requests from all categories to be included.

- `/NAME=<report-type>`

  Specifies which particular report is required (e.g. `CH_DOC_LIST`). For a full list of possible report types, refer to the "Change Management Reports" section of the *Reports Guide*.

  For a `BASELINE_DETAIL REPORT`, refer to the separate entry on page 405

- /CATALOGUE or /CATALOGUE/SECONDARY or /PENDING

  Determines the basis for the report. Only one of these qualifiers must be specified.

/CATALOGUE	report based on the primary (main) catalog of requests.
/CATALOGUE /SECONDARY	report based on the secondary catalog of requests.
/PENDING	report based on the requests pending to the user.

- /FILENAME=<outfile>

  Specifies the name of a file to receive the generated report. If /DETAIL is present, the file will include that output.

Additional Criteria to Ensure Inclusion in Report

All the remaining parameter values are used to specify additional criteria which requests must satisfy in order to be included in the report. Except for the parameters <date-from> and <date-to>, these parameters may include **%** (percent) characters as wild cards (i.e. each **%** character is considered to match zero or more characters in the corresponding request attribute).

The default in each case is to specify no additional criteria: i.e. all requests, which are otherwise valid for inclusion, may have any value for the corresponding attribute.

- /ATTRIBUTES=(<attr1>=<string1>,...)

  Specifies strings to be matched by the corresponding user-defined attributes in each of the requests to be reported on. Each <attrN> is the Variable Name defined for one of the attributes. Each <stringN> is a string for that attribute's value to match.

  Wild card **%** may be used in each <stringN> (see above).

  Please see "Requirements for Request Attributes" on page 26 for more information on required request attributes for the RPT command.

- /CHANGE_DOC_ID=<ch-doc-id>

  This is a string to be matched by the identifier of each of the requests to be reported on.

  Wild card **%** may be used (see above).

- /CH_DOC_TYPE=<ch-doc-type>

  This is a string to be matched by the type of each of the requests to be reported on.

  Wild card **%** may be used (see above).

- /PART=<part-spec> comprises:

  <product-id>:<part-id>.<variant>;<pcs>

<pcs>	is ignored in the matching criteria.
<part-spec>	must match a design part which is related to a request, in order for that request to be included in the report.

Wild card **%** may be used (see above).

- /PHASE=<phase>

  This is a string to be matched by the current phase of each of the requests to be reported on.

  Wild card **%** may be used (see above).

- /STATUS=<request-status>

  This is a string to be matched by the current status of each of the requests to be reported on.

  Wild card **%** may be used (see above).

- /USER=<user-name>

  This is a string to be matched by a Dimensions user associated with each of the requests to be reported on.

  Wild card **%** may be used (see above).

  A /CATALOGUE or /SECONDARY report includes all requests which (meet all other specified criteria and) have been created or actioned by a Dimensions user matching <user-name> at any time between <date-from> and <date-to>.

  A /PENDING report includes all requests which (meet all other specified criteria and) are awaiting actioning by a Dimensions user matching <user-name> at any time between <date-from> and <date-to>. (A request can be awaiting actioning by any of several Dimensions users. For valid inclusion of the request in the report, at least one of these users must match <user-name> as specified here, but they need not all do so.)

- /FROM=<date-from>

  Specifies the initial date associated with each of the requests to be reported on. (For usage, see details immediately above.) It must be in the format 25–DEC–1996.

  The default is 01–JAN–1900.

- /TO=<date-to>

  Specifies the final date associated with each of the requests to be reported on. (For usage, see details immediately above.) It must be in the format 25–DEC–1996.

  The default is the current system date.

- /NOHEADING_WRAP

  Specifies that any columns in the report headings where the data exceeds the column width are to be truncated.

  The default is that these columns are word wrapped.

- /NOWRAP

  Specifies that any columns in the body of the report where the data exceeds the column width are to be truncated.

  The default is that these columns are word wrapped.

- /INDENT

  Specifies that any lines in the body of the report that start with a number are to be indented by four times that number of columns.

By default there is no indentation.

- /DETAIL

  Specifies that the full text of each request listed in the report is to be printed following the report.

  By default just the report itself is printed.

- /HOLD

  Indicates that the report request is to be processed so as to produce a command script file, but that this is not to be executed.

  By default a Dimensions batch job to generate the report is submitted immediately.

- /PRINT

  Indicates that the report is also to be sent to a printer.

  By default the completed report is merely placed in <outfile>.

Constraints

Only users with the appropriate management privileges can run this command.

RPT – Baseline Detail Report

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<product-id>
<request-category>
/NAME=BASELINE_DETAIL
/USER_DEFINED
[/BASELINE=<baseline-spec>]
[/FROM=<date-from>]
[/TO=<date-to>]
/FILENAME=<outfile>
[/[NO]HEADING_WRAP]
[/[NO]WRAP]
[/[NO]INDENT]
[/[NO]HOLD]
[/[NO]PRINT]
```

Example

```
RPT PROD 1 /NAME=BASELINE_DETAIL /USER_DEFINED -
/FILENAME=hp_rm_chdocs96.rep -
/BASELINE=PROD:"R M VERSION 2 FOR HP" -
/FROM=01-JAN-1989 /TO=31-DEC-1996
```

Parameters and qualifiers

- `<product-id>`

  This is the product for which the report is required.

- `<request-category>`

  This is any valid request category. (Its value is not used in this report.)

- `/NAME=BASELINE_DETAIL`
  `/USER_DEFINED`

  are specified as shown.

  For other types of report, refer to the **RPT** entry for **Report Requests**, which precedes this one on page 401.

- `/BASELINE=<baseline-spec>`

Specifies the baseline(s) to be reported on. It comprises:

`<product-id>:<baseline-id>`

Wild card **%** may be used in `<baseline-spec>` to determine which baselines are to be included (i.e. each **%** character is considered to match zero or more characters in the corresponding baseline).

The default is to report on all baselines in the product.

- `/FROM=<date-from>`
  `/TO=<date-to>`

  specify the first and final dates on which an item, included in a reported baseline, may have been related to a request, in order for the request to be listed in the report. Each must be in the format 25–DEC–1996.

The defaults are 01–JAN–1900 and the current system date.

NOTE Any other RPT qualifiers from the Report Requests command on page 401 used here (including /DETAIL) will be ignored.

Constraints

Only users with the appropriate management privileges can run this command.

RRCD – Relate Requirement to Request

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
RRCD /CHANGE_DOC_ID=<request_id> /REQUIREMENT_ID=<requirement>
/CONTAINER_NAME=<container_name> /RM_PROJECTNAME=<project_name>
/RM_DBNAME=<dbname> /RM_URL=<url>
```

Example

```
RRCD /CHANGE_DOC_ID=REPX_CR_114
   /REQUIREMENT_ID= Marketing_Requirements.MRKT_000020;9
   /CONTAINER_NAME="Engineering Requirements"
   /RM_PROJECTNAME=QLARIUS_RM /RM_DBNAME=RM10
   /RM_URL="http://hostname/rtmBrowser"
```

Description

- `<request-id>`

  Specifies the request to be related to the requirement.

- `<requirement>`

  Specifies the requirement to be related to the request and comprises:

  ```
  <class_name>.<puid>;objId
  ```

  For example:

  ```
  Marketing_Requirements.MRTK_000020;4
  ```

- `<container_name>`

  Specifies the name of the originating Dimensions RM baseline, collection, document, or snapshot for the requirement (multiple "versions" of a requirement cannot be related to a single Dimensions CM request).

- `<project_name>`

  Specifies the Dimensions RM project name.

- `<dbname>`

  Specifies the Dimensions RM database name.

- `<url>`

  Specifies the Dimensions RM Browser URL.

IMPORTANT! You can only specify Dimensions RM requirements if you have installed the Dimensions RM integration, have associated Dimensions CM projects with Dimensions RM containers, and have associated Dimensions CM products with Dimensions RM projects. See the "Miscellaneous Functions" chapter in the Dimensions CM *User's Guide*, the *Dimensions CM-Dimensions RM ALM Integration Guide*, and the Dimensions RM documentation for details.

Constraints

Only users with the privilege "Perform Requirement Related Operation" (PRODUCT_REQUIREMENTMAN) can run this command.

RREG – Reassign User Registration

```
<existing-user-id>
/USER=<new-user-id>
```

Description

This command is documented in the *System Administration Guide*.

RSJ – Run Schedule Job

<job-id>

Example: `RSJ "MyJobName"`

Description

Enables you to force the execution of a scheduled job immediately, rather than at the scheduled time.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

RSTAT – Update Job Status

```
<job-key>
[/STATUS=<status>]
[/RESPONSE_FILE=[<node>::]<filespec>]
[/DESCRIPTION=text
[/RC=xx]
[/USER_FILENAME=<report-file>]
[/[UN]LOCK]
```

Description

RSTAT enables you to update job attributes such as return code, status, and description.

RSTAT is also used by the default templates to report the status of asynchronously submitted build jobs and to implement the collection of build outputs.

For details about remote job execution see the *System Administration Guide*.

Parameters and Qualifiers

- `<job-key>`

  String in the format `<id>-<job-uid>` that identifies the job whose attributes are to be updated.

- `/STATUS=<status>`

  Updates the status of the job in the job queue table. Valid values are FAILED or SUCCEEDED.

NOTE The status of an asynchronously submitted job cannot be reset to SUBMITTED.

- `/RESPONSE_FILE=[<node>::]<filespec>`

  The fully specified name of a file that is copied into a table in *Dimensions CM*, and which provides details of the remote job. You can use the syntax `node::` to specify a Dimensions tertiary node.

  This summary can only be viewed via the Administration Console Remote Jobs display.

- `/DESCRIPTION=text`

  Updates the job description.

- `/RC=xx`

  Species a numeric return code that indicates if the remote job successfully completed.

- `/USER_FILENAME=<report-file>`

  Specifies a file where the RSTAT execution report is generated.

- `[/[UN]LOCK]`

  Locks the job so that it cannot be deleted. Default: `/[UN]LOCK`

RUR – Run User-Defined Report

```
<report name>
/PRODUCT=<product-range>
[/PARAMETER=(<param2>,<param3>,...,<param8>)]
[/FILENAME=<output file>]
[/[NO]PRINT]
[/[NO]HOLD]
```

Example

```
RUR "VARIANTS TYPES FORMATS" /PROD="CAR%" -
/PARAM=(AAA_,"%",C) /PRINT
```

Parameters and qualifiers

- `<report name>`

  Specifies the kind of report to be produced. It must be one of the report names containing report definitions that has been set up via one of the following means:

  - The User Reports Administration cluster of the web-based Administration Console.
  - A JavaScript, utilizing the Dimensions `dmpmcli` scripting interface. Refer to the *Process Configuration Guide* for instructions on how to run such scripts. Example reports scripts for UNIX and Windows are available, online, in the directories `$DM_ROOT/AdminConsole/examples/reportsDemo.js` and `%DM_ROOT%\AdminConsole\examples\reportsDemo.js` respectively

- `/PRODUCT=<product-range>`

  This is a string to be matched by the product-id of one or more of the products in the database. "Wild card" characters may be used in the specified string — this is explained further in the *Reports Guide*.

  The Dimensions user executing the `RUR` command must hold a role in *every* product matched by the `<product-range>` string; otherwise execution of the command is terminated with an error message.

- `/PARAMETER=(<param2>,<param3>,...,<param8>)`

  This is a set of parameter values, separated by commas and enclosed in parentheses, which are the parameter values to be supplied to this report following the `<product-range>` string (which is always the value of parameter number 1).

  The last parameter, for which a value is supplied, may well be less than number 8 – it depends on the particular requirements of the command script file for the specified report name. However, note the following:

  There must be the exact number of non-blank values, in the correct order, as required by the `<report name>` *specified.*

  Lower case letters may be included in parameter values, in addition to the standard character set. **%** (percent) characters are also accepted. Any parameter value which contains embedded blanks must be enclosed in a set of double-quotation characters (" ").

NOTE The default of **%** for an omitted parameter value **is not applicable** in command mode. If a single **%** is wanted as a parameter value, then that must be coded. Two consecutive commas in the set of parameter values will be flagged as a syntax error.

There is no default. The parameters **must be supplied exactly** as required by the `<report name>`. The p

ose of the square brackets is to indicate that the /PARAMETER qualifier **must be omitted**, if the report name is one which takes **no** input parameter values, apart from the single <product-range> string.

- /FILENAME=<output file>

  Specifies the name of a file which RUR will create in the user area, and into which it will write the output from the report. If omitted, the file name will be user_report.out

- /PRINT

  Specifies that the report output is also to be spooled for printing. If omitted, the report is merely placed in <output file> (specified or default).

- /HOLD

  Specifies that the batch job to produce the report is to be created, but that it is to be left in the user area for the user to start its execution later. If omitted, execution of the batch job is initiated automatically as soon as it has been created.

Constraints

This command can be run by users who hold a role, either for the single product-id specified or for every one of the product-ids designated by a string using wildcard characters. However, if a user with the role of PRODUCT-MANAGER assigns a special role PCMS-CM-USER to the wild card user *, then any user will able to run a report on the product even if they do not have such a role explicitly assigned.

RVDA – Remove VDA Records

```
[/REPORT]
[/ALL]
[/AREA_LIST]
```

Description

VDA records store information in a database about the status and history of deployment jobs. Use this command to remove unused records of deleted areas.

Example

```
RVDA /AREA_LIST=(AREA_DEV,AREA_QA,AREA_LIVE)
```

Parameters and Qualifiers

- `/REPORT`

  Lists all orphaned VDA records.

- `/ALL`

  Cleans up all orphaned areas.

- `/AREA_LIST=(area1,area2,...)`

  Removes the specified orphaned areas.

RWCD – Relate Project to Request

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
/CHANGE_DOC_IDS=(<request1,<request2>,request3>,...)
```

Example `RWCD PAYROLL:PRJ_INITIAL /CHANGE_DOC_IDS=(PAYROLL_CR_21)`

Description

The example above relates the PAYROLL_CR_21 request to the PAYROLL:PRJ_INITIAL project.

Constraints

Only users with the appropriate management privileges can run this command.

RWS – Remove Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
```

Example

```
RWS "PROD_X:TEST_WS"
```

Parameters and qualifiers

- `<project-spec>` comprises:

  `<product-id>:<project-id>`

Constraints

This command can be run only by a user with the appropriate management privileges for the project concerned.

A project used as a child collection cannot be deleted.

RWWS – Relate Project to Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<child-project-spec>
/WORKSET=<parent-project-spec>
[/RELATIVE_LOCATION=<relative-path]
```

Example

```
RWWS <child-project-spec> /WORKSET=<parent-project-spec>
```

Parameters and qualifiers

- `<child-project-spec>`

  Specifies the child project in the parent-child relationship.

- `<parent-project-spec>`

  Specifies the parent project in the parent-child relationship.

- `<relative-path>`

  Specifies the relative path of the file system directory to which the child project's top-level directory is mapped with respect to the file system directory to which the parent project's top-level directory is mapped.

NOTES
- Specifying an empty string clears the relationship-level project root for the user.
- If there is no relationship-level project root, /RELATIVE_LOCATION is used.
- If the path contains `::` (that is, it has the format `<node-or-area>::<path>`, and `<node-or-area>` matches the name of an existing area), the path is interpreted as an offset (relative path) with regard to the area root directory. (This offset may be empty, in which case the file system directory is set to equal the area directory.) Otherwise, standard Dimensions interpretation of the path applies.

Description

Creates or modifies a parent-child relationship between the specified child project and the specified parent project.

If a relationship already exists, the value of /RELATIVE_LOCATION is used to update the relationship.

Constraints

Only users with the appropriate management privileges can run this command.

SAVE – Save to Persistent Symbol Table

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<symbol>
[<symbol2>]
[/LITERAL=value]
```

- `/LITERAL=value {/LIT}`

  Creates a symbol in the persistent table that has the value specified by `<symbol>`.

The SAVE command copies data from the temporary or persistent symbol table to the persistent symbol table and is part of structured information return processing. For full details, see the *Developer's Reference*.

SCWS – Set Current Project

```
[<project-spec>]
[/ROOT_PROJECT=<project-spec>]
[/DIRECTORY=<directory>]
[/LIBRARY_CACHE_AREA=<area-name>]
[/CHANGE_DOC_ID=<request-id>]
[/[NO]DEFAULT]
[/[NO]CHECK]
[/USERS]
[/USER_BRANCH]
[/PART]
```

Examples

```
SCWS "PROD:WS CUSTOM" /DIR="/product9" /NODEFAULT
```

```
SCWS "QLARIUS:PROJ" /NOCHECK
    /DIRECTORY="D:\PROJECTS\CM\DEV_REPOSITORY\web\PROJ"
    /USER_BRANCH="java_p1_1" /CHANGE="." /PART="."
    /LIBRARY_CACHE_AREA="."  /NORESET
```

NOTE If you specify SCWS without any parameters or qualifiers the following information is displayed about a stream or project:

- Login user name
- Project specification of the current Dimensions session
- Current version
- Default work area
- Locked or unlocked
- Stream is personal
- Trunk and branch revisioning is enabled or disabled
- Revisioning is enforced
- Parallel checkout is allowed
- Uses local stages
- Request path control is enforced
- Branches are assigned to the project
- Uses a library cache area
- Default request (if specified)
- Default design part (if specified)
- Request provider (CM or SBM)

Parameters and qualifiers

- <project-spec>

  Comprises:

  <product-id>:<project-id>

<project-spec>	Must be stated when /DIRECTORY and/or /(NO)DEFAULT are included in the command.

- /ROOT_PROJECT=<project-spec>

  Comprises:

  <product-id>:<project-id>

  This optionally specifies the root project. Use this when the current project occurs in more than one project tree. If you set this here, you do not need to set it on subsequent commands.

- /DIRECTORY=<directory>

  Specifies the top level directory specification for the project. This "working location" defines the point in the directory hierarchy structure below which (or relative to which) the project file name is placed e.g. in UNIX <dir>/<ws_filename>. This overrides what would have been the default working location.

  Operating system permissions permitting, you can specify a Windows network path when running SCWS from a UNIX host as follows:

  /DIRECTORY=/"nodename::<Drive>\<path>"

  For example:

  /DIRECTORY=/"earth::F:\Optimus"

  Checked out or fetched items will then be directed to the project file whose path is identified by this new path name (e.g. in UNIX, <dir>/<ws_filename>). You can also use a work area.

NOTE If the path contains :: (that is, it has the format <node-or-area>::<path>, and <node-or-area> matches the name of an existing work area), the path is interpreted as an offset (relative path) with regard to the work area root directory. (This offset may be empty, in which case the file system directory is set to equal the area directory.) Otherwise, standard Dimensions interpretation of the path applies.

- /LIBRARY_CACHE_AREA=<area-name>

  Specifies a library cache area defined with the CLCA (Create Library Cache Area) command. During fetch operations (FI, FWI, FBI, FCDI, EI, EWI, EBI, ECDI, DOWNLOAD), Dimensions checks whether the library cache area associated with the current project already contains a copy of of the requested file. If so, Dimensions copies the file from the library cache to the user file area instead of from the library itself, which improves performance when the connection between the item library node and the user's network is slow.

- /CHANGE_DOC_ID=<request-id>

  Sets a default per-user request on the project. To unset a default request, set /CHANGE_DOC_ID to ".".

- /DEFAULT or /NODEFAULT

  /DEFAULT — Specifies that the current project specified by this command will remain the default for all future Dimensions sessions until respecified. /DEFAULT is the default.

  /NODEFAULT — Specifies that the current project specified by this command is for the duration of the present Dimensions session only (this should not be confused with your operating-system session). The current project will revert to its former setting once the session is exited. This project will also be used for batch jobs started during the Dimensions session.

  The /NODEFAULT qualifier should, however, be treated with caution when submitting command files containing SCWS and commands that spawn separate operating-system processes e.g. Create Baseline (CBL). The scope of the /NODEFAULT qualifier does not get extended to such commands. To guarantee correct behavior, it is recommended that SCWS /DEFAULT is used to set the current project before such commands as CBL, and then used again afterward to reset it to its earlier specification (if so desired).

- /CHECK or /NOCHECK

  Specifies whether SCWS is merely to check and report on the feasibility of performing the requested action, or is actually to implement it. /CHECK is the default.

- [/USERS]

  Enables you to change the default project settings of another user account or a group of users. The main purpose is to provide a method with a single command for setting the library cache area for a group of users. You may specify one or more user IDs and/or group IDs. Requires the privilege Manage Users and Group Definitions (ADMIN_USERMAN).

  If you specify a group ID the command expands the group into its constituent user IDs and then iterates through these users updating the project settings for each one in turn. If a new user is later added to the same group, this new user will not inherit the projects settings and a specific SCWS command for the new user is required.

  Examples:

  To change the project settings for 'user1' and 'user2':

  ```
  scws repx:repx4 /users=(user1, user2) /dir="d:\temp\user4"
  /USER_BRANCH="4" /LIBRARY_CACHE_AREA=lca4 /PART=REPX:P4
  /CHANGE_DOC_ID=REPX_TDR_4
  ```

  To change the project settings for all the users in the group 'dev_users':

  ```
  scws repx:repx4 /users=(dev_users) /dir="d:\temp\user4"
  /USER_BRANCH="4" /LIBRARY_CACHE_AREA=lca4 /PART=REPX:P4
  /CHANGE_DOC_ID=REPX_TDR_4
  ```

  To change the project settings for all the users defined in the group 'dev_users' and for the users 'fred' and 'george':

  ```
  scws repx:repx4 /users=(fred, dev_users, george) /
  dir="d:\temp\user4"
  /USER_BRANCH="4" /LIBRARY_CACHE_AREA=lca4 /PART=REPX:P4
  /CHANGE_DOC_ID=REPX_TDR_4
  ```

- /USER_BRANCH

  Specifies a valid named branch that will be set as the current branch. To use the default named branch for this project, not the branch set by the user, specify ".".

- /PART

  Specifies a design part.

Example Output

```
Dimensions>scws
The current project is PVCS:DOCS_DIM_10
The working location is D:\Dimensions
The current user is JDOE
        Description                 : Work Set DOCS_DIM_10.AAAA
        Status                      : UNLOCKED
        Trunk                       : TRUE
        Branch                      : FALSE
        Enforce Revisioning         : FALSE
        Parallel Checkout           : TRUE
        Deployment                  : AUTOMATIC
        Copy on deploy              : FALSE
There are no valid Named Branches for this project
The project library cache area is not defined
Operation completed
```

Description

This command sets the user's current project and optionally their default project.

To set the current project to the "Global Workset", the <project-spec> must be set to $GENERIC:$GLOBAL (see Introduction on page 12).

Constraints

This command can be run by users who have a role on the project being set.

SDF – Set Data Format Flags

```
<format>
[/DESCRIPTION=<format -description>]
[/CLASS=<class-no>]
[/MIME_TYPE=<mime-type>]
[/COMPRESSION_LEVEL=<level>]
[/[NO]USE_DELTA_COMPRESSION]
```

Example

```
SDF TXT /DESCRIPTION="Plain text" -
/CLASS=1 /MIME_TYPE="TEXT/PLAIN"
```

Parameters and qualifiers

- `<format>`

  the format definition being updated.

- `/DESCRIPTION=<format-description>`

  the new descriptive name for the format.

- `/CLASS=<class-no>`

  Specifies the new file types, where:

  1=TEXT
  2=BINARY
  3=OpenVMS
  4=Macintosh
  5=NT

- `/MIME_TYPE=<mime-type>`

  Specifies the new Multipurpose Internet Mail Extension (MIME) type. MIME types comprise seven broad categories, with each category also having subcategories defined by using a forward slash (/) separator. The broad categories are: Application, Audio, Image, Message, Multipart, Text and Video. An example of a subcategory is APPLICATION/WORD.

- `COMPRESSION_LEVEL=<level>`

  Specifies the compression level to be used when getting item revisions assigned this data format. Use a digit from 0 to 9, where 0 indicates no compression, 1 means fastest compression method (but less compression) and 9 indicates slowest compression method (but best compression). If this qualifier is omitted, text file formats will use fastest compression method (level 1) while all other file formats will use no compression.

- `/[NO]USE_DELTA_COMPRESSION`

  Decreases the size of the transferred item by only sending sections that have been modified between revisions.

Description

This command enables a user with the role of `TOOL-MANAGER` to update the file format definition. These defined file formats can then, where appropriate, be subsequently assigned:

- By a user with the role of `TOOL-MANAGER` to particular item types using the Assign Data Formats to Item Types (ADF) command (page 42).

- By a user with the role of PRODUCT-MANAGER or CHANGE-MANAGER to a particular request type using the Assign Data Formats to Request Types (ACF) command (page 41).

This function is also available from the Process Modeler, Data Formats and MIME Types option.

See the DDF command (page 153) for a description of the uses for file formats.

Constraints

This command can be run only by a user with the appropriate management privileges for the product.

SDPBL – Submit Deploy Baseline

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
<baselineName>
/AREA_LIST=(<areaList>)
[/COMMENT=<userComment>]
[/USER_FILENAME=<listFile>]
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
[/WORKSET=<project>]
[/[NO]FORCE]
```

Parameters and qualifiers

- `<baselineName>`

  Name of the baseline to deploy.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the deployment.

- `/USER_FILENAME=<listFile>`

  A user specified file containing a list baselines to be deployed. You can omit the `<baselineName>` parameter and use this option to deploy many baselines at onc example:

  ```
  /USER_FILENAME="C:\temp\list_of_baselines.txt" /
  AREA_LIST=("MY_AREA_1","MY_AREA_2") /COMMENT="my comment"
  ```

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS" (Dimensions date time)`
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z" (ISO8601 date time)`

  Note that the following formats are not accepted:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (omission of milliseconds does not work)
  - `"DD-MM-YYYY HH24:MI:SS"`

- `/WORKSET=<project>`

  The project or stream associated with this action.

- `/[NO]FORCE`

  Set `/FORCE` to force rollback to the highest revision when multiple revision matches are found. Set `/NOFORCE` for the command to list the multiple matches and stop.

Description

Use the `SDPBL` command to schedule the deployment of a Dimensions baseline.

SDPI – Submit Deploy Item

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
[<itemSpec>|<fileName>]
/COMMENT=<userComment>
/USER_FILENAME=<listFile>
/WORKSET=<projectName>
/AREA_LIST=(<areaList>)
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
/[NO]FORCE
```

Parameters and qualifiers

- `[<itemSpec>|<fileName>]`

  Name or specification of the item to deploy.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the deployment.

- `/USER_FILENAME=<listFile>`

  A user specified file containing the list of items or files that are to be deployed. Specifying this option allows you to deploy many items at once. This option is mutually exclusive to specifying `<itemSpec>|<fileName>`.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  Note that the following formats are not accepted:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (omission of milliseconds does not work)
  - `"DD-MM-YYYY HH24:MI:SS"`

- `/[NO]FORCE`

  Set `/FORCE` to force rollback to the highest revision when multiple revision matches are found. Set `/NOFORCE` for the command to list the multiple matches and stop.

- `/WORKSET=<projectName>`

  Name of the project or stream that contains the item to promote.

Description

Use the `SDPI` command to schedule the deployment of a Dimensions item.

SDPRQ – Submit Deploy Request

NOTE This command is not supported in products that use the Dimensions Automation deployment model.

```
<requestId>
/COMMENT=<userComment>
/USER_FILENAME=<listFile>
/WORKSET=<projectName>
/AREA_LIST=(<areaList>)
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
/[NO]CANCEL_TRAVERSE
/[NO]FORCE
```

Parameters and qualifiers

- `<requestId>`

  ID of the request to deploy.

- `/COMMENT=<userComment>`

  Comment that describes the reason for the deployment.

- `/USER_FILENAME=<listFile>`

  A user specified file containing the list of requests that are to be deployed. Specifying this option allows you to deploy many items at once.

- `/WORKSET=<projectName>`

  Project or stream associated with the request to deploy.

- `/AREA_LIST=<areaList>`

  List of target deployment areas.

- `/DEPLOY_START_TIME`

  The start time for the operation to begin, in one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z" (ISO8601 date time)`

  Note that the following formats are not accepted:

  - `"YYYY-MM-DDTHH24:MI:SSZ"` (omission of milliseconds does not work)
  - `"DD-MM-YYYY HH24:MI:SS"`

- `/[NO]CANCEL_TRAVERSE`

  Set `/CANCEL_TRAVERSE` to limit deployment to just the specified request.

- `/[NO]FORCE`

  Set `/FORCE` to force rollback to the highest revision when multiple revision matches are found. Set `/NOFORCE` for the command to list the multiple matches and stop.

Description

Use the `SDPRQ` command to schedule the deployment of a Dimensions request.

SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment

```
PRINTER <printer-cmd>
or
DIRECTORY <directory-spec>
or
OVERWRITE ON|OFF
or
CMD_TRACE ON|OFF [/USER_FILENAME=<file-name>]
or
INFO ON|OFF
or
TIMEZONE <timezone-name>
EOL WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW
```

Example

```
SET PRINTER "lpr -Pdev"
```

Parameters and qualifiers

- `PRINTER <printer-cmd>`

  Set a valid UNIX or Windows printer command. This is used in preference to the value assigned to the *DM_PRINT* symbol.

- `DIRECTORY <directory-spec>`

  Set the current Dimensions default directory is set to the directory specified. This must be an appropriate format for the host machine operating system, and it can be absolute or relative to the current directory.

- `OVERWRITE [ON]|OFF`

  When checking out or getting an item revision, you can specify whether or not Dimensions is allowed to perform such an operation depending on:

  - The existence of a local file of the same name.
  - The status (read-only or writeable) of an existing local file of the same name.

  No overwriting is the normal default and results in a file only being successfully checked out or gotten by Dimensions if the local (target) file does not already exist or is marked read-only. This is the traditional Dimensions 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). This default can be overridden on a per command basis using `/OVERWRITE` qualifier of EI and FI as explained on page 225 and page 240 respectively.

  Setting /OVERWRITE to ON will also override the following qualifiers for commands that support them: [NO], [FORCE], [ADD], [DELETE]. Consider this when using any command that includes these qualifiers.

  The `SET OVERWRITE ON | OFF` command allow you to control – on a per Dimensions session basis – the default behavior globally without needing to specify the `/OVERWRITE` or `/NOOVERWRITE` qualifier on every FI or EI command. At the beginning of a Dimensions session, by default `SET OVERWRITE OFF` would be in effect; however, you could change this environment as illustrated in the following examples:

  ```
  SET OVERWRITE OFF
  FI "FS:CBEVENT C.A-SRC;b1#4" -
  /USER_FILENAME="e:\test\cbevent.c" /NOEXPAND
  ```

would not allow `cbevent.c` to be overwritten if it existed and was not marked read only.

```
SET OVERWRITE ON
FI "FS:CBEVENT C.A-SRC;b1#4" -
/USER_FILENAME="e:\test\cbevent.c" /NOEXPAND
```

would overwrite `cbevent.c` if it existed, irrespective as to whether it was marked read only or not.

- `CMD_TRACE [ON]|OFF [/USER_FILENAME=<file-name>]`

  Executing
  `SET CMD_TRACE ON`

  creates a log file[1] in the DM_TMP directory of the format

  Windows:
  `%DM_TMP%dmappsrvcmd_<user-name>.log`

  UNIX:
  `$DM_TMP/dmappsrvcmd_<user-name>.log`

  which records all the commands and session details run through the user's session.

  You can explicitly specify the name of the log file created on the server by the use of the `/USER_FILENAME` qualifier.

  Executing
  `SET CMD_TRACE OFF`

  will switch off the command logging.

  For a sample of the information contained in this log file, refer to "Logging All Commands Run by All Users" on page 29.

- `INFO ON|OFF`

  Set information on or off.

- `TIMEZONE <timezone name>`

  Sets the timezone for the Dimensions CM Server. This setting determines how the server tracks time for scheduled operations, such as scheduling deployment. Enter a standard database TZ value, such as `America/Los_Angeles`, to represent Pacific time. To display a list of all possible timezon.e values, use the `HELP timezones` command. See "HELP – Help" on page 255

- `EOL WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW`
- `[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]`

  Sets or displays the end-of-line handling mode for the current connection that will be used when getting text files over the current connection. The options are:

WINDOWS	Ensures that gotten text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.

1. The log file is created on the Dimensions server, *not* the client.

UNIX	Ensures that gotten text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Restores the default Dimensions end-of-line handling mode, i.e. text files gotten to a Windows node will have each line terminated with a CR/LF pair, while text files gotten to a UNIX node will have each line terminated with a single LF character.
UNCHANGED	Ensures that text files are gotten as-is from the item library without any end-of-line processing at all.
SHOW	Ask Dimensions to display its current EOL setting.

See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

Constraints

This command sets the above Dimensions parameters only for the duration of the current Dimensions session.

SF - Set Favorites

```
[/PROJECT=<project-spec>]
[/STREAM=<stream-spec>]
[/USER_WORKSETLIST]
[/OFF]
```

Description

Adds the specified streams and projects to your favorites list.

Examples

```
SF /STREAM=PROD:STR_ID
```

```
SF /STREAM=PROD:STR_ID /OFF
```

Parameters and Qualifiers

- /PROJECT=<project-spec>

  Specifies a project.

- /STREAM=<stream-spec>

  Specifies a stream.

- /USER_WORKSETLIST

  Specifies a file that contains a list of streams and projects (each one on a new line).

- /OFF

  Removes all the specified streams and projects worksets from your favorites list.

SHELVE - Shelve Changes to a Personal Stream

```
/SHELF_NAME=<new stream name>
[/STREAM=<stream-id>] or [/WORKSET=<stream-id>]
[<file-spec> or /DIR=<directory-spec>] or [/USER_FILELIST=<filelist-
    file>]
[/[NO]RECURSIVE]
[/LOGFILE=<file-spec>]
[/ATTRIBUTES=(<name>=<value>, ...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/CODEPAGE=<code-page> or DEFAULT]
[/COMMENT=<text>]
[/DESCRIPTION=<description>]
[/SHELF_DESC=<description>]
[/DEFAULT_BRANCH=<branch_name>]
[/PART=<part-spec>]
[/CONTRIBUTER_STREAMS=(<stream-id>, ...)]
[/ALL]
[/USER_DIRECTORY=<directory-path>]
[/RELATIVE_LOCATION=<directory-spec>]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/CONTENT_ENCODING=<file-encoding>]
[/[NO]ADD]
[/[NO]UPDATE]
[/[NO]DELETE]
[/[NO]QUIET]
[/[NO]VERBOSE]
[/[NO]EXECUTE]
```

Description

Use the SHELVE command to backup work that is in progress in your local work area. You can then work on an unrelated issue. When you need to resume work, merge the shelved changes back into the local work area.

The SHELVE command:

- Creates a new personal stream that contains some, or all, of the changes in a local work area. The new stream is a branch of the stream that owns the work area.
- By default resets the work area to match the tip of the stream that owns it.

Examples

- `SHELVE /SHELF_NAME=DEF_A /USER_DIR=c:\work\Qlarius_trunk`

  This example scans the specified local work area. If there are changes to existing files a new personal stream called `DEF_A` is created that contains the contents of the local work area, including the changes. New files and folders are ignored and metadata is not updated in the work area. The personal stream is a branch of the stream that was used to create the local work area. The work area is automatically reset to the tip of the parent stream.

- `SHELVE /SHELF_NAME=DEF_B /USER_DIR=c:\work\Qlarius_trunk /ADD /NORESET`

  This example scans the specified local work area. If there are changes, including new files and folders, a new personal stream called `DEF_B` is created that contains the contents of the local work area, including the changes. Metadata is not updated in the work area. The personal stream is a branch of the stream that was used to create the local work area. `/NORESET` is specified therefore the work area is *not* automatically reset to the tip of the parent stream.

Parameters and Qualifiers

- `/SHELF_NAME=<new stream name>`

  Specifies a unique ID for the new personal stream.

- `/STREAM=<stream-id>] or [/WORKSET=<stream-id>]`

- `<file-spec>`

  Specifies the name of a file to be shelved. This path is relative to the work area root.

- `/DIRECTORY=<directory-spec>`

  Specifies a directory path to be shelved. This path is relative to the work area root.

- `/USER_FILELIST=<filelist-file>`

  Specifies a file containing a list of file names to be shelved. Each file name must be on a separate line. File names may be specified as either relative or absolute paths. If the path is absolute it is interpreted as a full stream path. If not, Dimensions obtains the stream path by mapping the file name to the operation root directory, which is the current working location as specified by the last SCWS command. If this a mapping is not possible, the file name is ignored.

- `/[NO]RECURSIVE`

  If `/DIRECTORY` is specified and this qualifier is not present, all files in all directories beneath the one specified are shelved. `/NORECURSIVE` specifies that only files at the specified directory level are shelved.

  Default: `/RECURSIVE`

- `/LOGFILE=<file-spec>`

  Generates a log file at the specified file location that contains the results of all the individual Dimensions CM operations executed during shelving with this command.

- `/ATTRIBUTES=(<name>=<value>, ...)`

  Specifies the user defined attributes to set on the newly created revisions. All attributes specified must be valid for the item types created.

- /CHANGE_DOC_IDS=(<request1>,<request2>,...)

  Specifies the requests for the shelved items to be related to. The originally fetched versions will be related as "Affected", and the newly created versions will be "In Response To".

- /CODEPAGE=<code-page>

  Specifies the code page to be associated with the shelved items.

- /COMMENT=<text>

  Specifies a comment to apply to all the shelved item revisions.

- /DESCRIPTION=<description>

  Describes the shelved items.

- /SHELF_DESC=<description>

  Describes the new personal stream.

- /DEFAULT_BRANCH=<branch_name>

  Specifies the default branch for the new personal stream.

- /PART=<part-spec>

  Specifies the design part specification to which the shelved items belong, in this format:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

- /CONTRIBUTER_STREAMS=(<stream-id>, ...)

  If the work area contains files that originated from other streams that need to be shelved, use this qualifier to specify which streams to add content from.

- [/ALL]

  Content originating from any stream is also included when shelving files.

- /USER_DIRECTORY=<directory-path>

  Specifies a directory other than the current working location. For example, the following command creates a stream from C:\temp regardless of the current working location:

  ```
  SHELVE /USER_DIRECTORY="C:\temp"
  ```

- /RELATIVE_LOCATION=<directory-spec>

  Specifies a project, stream, or baseline directory which is to be the "virtual" root directory for the duration of this command. If this parameter is given, the paths specified in <file-spec> or /DIRECTORY must be relative to the directory specified with /RELATIVE_LOCATION.

- /FILTER=<filter-name>

  Only shelves files that satisfy the criteria specified in the area filter <filter-name>.

  An area filter is a regular expression that used the same syntax as the Dimensions GREP command.

- `/USER_FILTER=<filter-file-spec>`

  Specifies the name of a local file containing the definition of a file filter to be used when shelving files. You can use the Dimensions `node::` syntax. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502.

  Only files matching the filter (and not excluded by the filter) are shelved when a user filter is specified.

- `/CONTENT_ENCODING=<file-encoding>`

  Specifies the content encoding for new item revisions being shelved. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- `/[NO]ADD`

  Shelves all new files and folders.

  Default: `NOADD`

NOTE If you specify /ADD, you may also need to specify /UPDATE if there are updates that need to be performed as well (specifically moves).

- `/[NO]UPDATE`

  Allows updating (and refactoring) of existing content in the repository.

  Default: `UPDATE`

- `/[NO]DELETE`

  Allows existing content to be deleted from the repository.

  Default: `NODELETE`

- `/[NO]QUIET`

  Only displays critical messages.

- `/[NO]VERBOSE`

  Prints additional information about the shelving process.

- `/[NO]EXECUTE`

  Forces the transfer of files while generating a script file containing the equivalent Dimensions commands.

Constraints

To create personal streams the PROJECT_PERSONAL_STREAM_CREATE privilege is required. To deliver to, and update from, the personal stream you require the usual combination of product-level privileges.

SHOW - Show Hidden Streams and Projects

```
[/STREAM=PROD:STREAM_ID]
[/PROJECT=PROD:PROJ_ID]
[/USER_WORKSETLIST]
[/NOUNLOCK]
```

Description

Makes visible streams and projects that are hidden (see the HIDE command on page 256) and unlocks them.

Example

```
SHOW /PROJECT=QLARIUS:MAINLINE_JAVA
```

Makes visible a project with the specification `QLARIUS:MAINLINE_JAVA`.

Parameters and Qualifiers

- /STREAM

  The specification of a stream to show.

- /PROJECT

  The specification of a project to show.

- /USER_WORKSETLIST

  The path of a local file that lists multiple streams and projects to show (specify each on a new line).

- /NOUNLOCK

  Does not unlock streams and projects after they are made visible.

SI – Suspend Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/WORKSET=<project-spec>]
```

Example

```
SI PROD:"QUERY RELEASE".AAAA-SRC;1
```

or, to suspend all revisions

SI PROD:"QUERY RELEASE".AAAA-SRC;*

Parameters and qualifiers

- `<item-spec>` comprises:

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

  `<item-id>` may be omitted if `<file-name>` is specified.

  `<variant>` may be omitted if only one exists.

  `<revision>` may be specified as * (asterisk) to suspend all revisions of the item that are in the project. If omitted, the latest revision is suspended (see note in About the Command-Line Interface on page 12).

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

NOTE Suspended items will be included in a revised baseline if they are identified as requiring updates based on a related request used in the revised baseline specification.

Constraints

This command can be run at any lifecycle state either by a user with the appropriate management privileges or by a user for whom the item is pending.

A suspended item may be 'unsuspended' by actioning it to a valid state in the lifecycle.

SPSP – Set Per-Stage Preservation Policy

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
/WORKSET=<project-spec>
/STAGE=<deployment-stage>
[/POLICY=<policy-id> | /RESET_POLICY]
```

Example

The command

```
SPSP /WORKSET="PAYROLL:EXEDLL 2.0" /STAGE="UNIT TEST" -
/POLICY="PAYROLL:DEFAULT_POLICY"
```

assigns a preservation policy to UNIT TEST builds of the PAYROLL:EXEDLL 2.0 project.

Parameters and qualifiers

- `/WORKSET=<project-spec>`

  Comprises `<product-id>:<project-id>` and specifies the project.

- `/STAGE=<deployment-stage>`

  Specifies the deployment stage.

- `/POLICY=<policy-spec>`

  `<policy-spec>` comprises `<product--id>:<policy-id>` and specifies the preservation rules policy to be applied at this stage for the specified project.

- `/RESET_POLICY`

  Specifies that the preservation policy is to be reset.

Description

The SPSP command specifies per-stage project build properties that apply to all the build areas defined for the specified build stage within a project.

SPV – Suspend Design Part Variant

```
<part-spec>
```

Example

```
SPV PROD:"RELEASE MANAGEMENT".IBM
```

Parameters and qualifiers

- `<part-spec>` comprises:

  `<product-id>:<part- id>.<variant>;<pcs>`

  `<variant>` may be omitted if only one exists.

  `<pcs>` is ignored; the current PCS is always used.

Description

This command enables a design part variant that is no longer required to be suspended at its current PCS. In the `SUSPENDED` state a design part variant can serve no useful purpose in the design process. Also, once set to this state it cannot be restored to active use at its current PCS. Only by creating a new PCS via the UP command can a design part variant be restored to active use.

Constraints

This command can be run only by a user with the appropriate management privileges for the selected design part. This design part must be in an `OPEN` state but not be referenced in an open request.

SRAV – Submit Rollback Area Version

```
<area_name>;<version>>
[/COMMENT=<userComment>]
/DEPLOY_START_TIME="DD-MON-YYYY HH24:MI:SS" | "YYYY-MM-
    DDTHH24:MI:[SS.sss]Z"
/[NO]FORCE
[/WORKSET=<projectName>]
```

Description

Roll backs a deployment from a specific area version.

Parameters and Qualifiers

- `<area_name>;<version>`

  Specifies the name and version of an area to roll back.

  Default: the latest area version

- `/COMMENT: <userComment>`

  Describes the rollback.

- `/DEPLOY_START_TIME`

  Specifies the start time for the roll back to start, use one of the following formats:

  - `"DD-MON-YYYY HH24:MI:SS"` (Dimensions date time)
  - `"YYYY-MM-DDTHH24:MI:[SS.sss]Z"` (ISO8601 date time)

  The following formats are not accepted:

  - "YYYY-MM-DDTHH24:MI:SSZ" (omission of milliseconds does not work)
  - "DD-MM-YYYY HH24:MI:SS"

- `/[NO]FORCE`

  Disables the area version. Item revisions referred to by the disabled area version are not updated or removed.

  If the area you specified includes an area version that belongs to a deleted project or stream, this qualifier enables the command to complete.

  Default: `/NOFORCE`

- `/WORKSET=<projectName>`

  The name of the project or stream affected by the rollback.

SSPM – Display Values in Symbol Tables

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
[ON [/USER_FILENAME="file"]]
[DBOTH]
[DPERSISTENT]
[DTEMP]
[NONE]
[OFF]
[LOAD]
[DUMP /USER_FILENAME="file"]
[DVAR <variableName>]
```

Parameters and qualifiers

- `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.

The SSPM command controls structured information return processing. For full details, see the *Developer's Reference*.

SUB – Subscribe to Notification Rule

```
<notification-id>
[/[NO]DIGEST]
[/USER_LIST=(user1,user2,...)]
[/AREA=<area-name>]
[/WORKSET=<project-name>]
[/ROLES=(role1,role2,...)]
```

Example

```
SUB <rule-id> /USER_LIST=Smith
```

Parameters and qualifiers

- `<notification-id>`

  Name of the notification rule. For a complete list of notification rules, see the *Process Configuration Guide*.

- `/DIGEST`

  Enables this subscription for digests (notification summaries).

- `/USER_LIST`

  Specifies users to subscribe to this notification rule.

- `/AREA=<area-name>`

  For any of the deployment related notification rules, including PROMOTED_ITEM_NOTIFICATION, PROMOTED_REQUEST_NOTIFICATION, PROMOTED_BASELINE_NOTIFICATION, ITEM_DEPLOYMENT_NOTIFICATION, REQUEST_DEPLOYMENT_NOTIFICATION, and BASELINE_DEPLOYMENT_NOTIFICATION, specifies the area against which this notification will apply. If the operation subscribed to occurs in another area, then no email will be generated.

- `/WORKSET=<project-name>`

  For any of the deployment related notification rules, including PROMOTED_ITEM_NOTIFICATION, PROMOTED_REQUEST_NOTIFICATION, PROMOTED_BASELINE_NOTIFICATION, ITEM_DEPLOYMENT_NOTIFICATION, REQUEST_DEPLOYMENT_NOTIFICATION, and BASELINE_DEPLOYMENT_NOTIFICATION, specifies the project or stream against which this notification will apply. If the operation subscribed to occurs in another project or stream, then no email will be generated.

- `/ROLES`

  Specifies roles to unsubscribe from this notification rule.

Description

Subscribe users to a notification rule. The /AREA and /WORKSET filters only apply to notifications that are deployment specific, such as promotion, deployment, and rollback notifications

SVBF – Set Version Branch Flags

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<branch-id>
[/DESCRIPTION=<description>]
[/[NO]LOCK]
[/OWNER]
```

Example

```
SVBF MAINT/LOCK /OWNER=LOCAL
```

Parameters and qualifiers

- `<branch-id>`

  unique branch identifier.

- `/DESCRIPTION=<description>`

  brief description of the purpose for the branch.

  If omitted, the description last entered (using DVB or SVBF) remains unchanged.

- `/LOCK`

  Optional flag to specify that the branch is locked and further development along it cannot take place.

- `/NOLOCK`

  Optional flag, negation of `LOCK` and is the default.

- `/OWNER=<site_id>`

  where `<site_id>` is either:

  - `LOCAL`, a keyword which can be used to set the ownership to the local base database, or
  - `<node_name>:<dbname>@@<sid>`

`<node_name>`	=	the node name
`<dbname>`	=	the base database
`<sid>`	=	the # sid

NOTE @@ is used because @ is the default Dimensions Escape character for the command line. The parameter `OWNER` enables change in branch ownership created by the Replicator product.

Description

This command modifies (sets) branch-id definitions that were defined using the DVB command i.e. the description or lock status.

Constraints

Only users with the appropriate management privileges can run this command.

SWF – Set Project File Name

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/WS_FILENAME=<ws-filename>
[/WORKSET=<project-spec>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
```

Example

```
SWF PROD:"QUERY RELEASE"-SRC /FILENAME=qr.c -
/WS_FILENAME=maintenance/src/system.c
```

This command is used to change the name of the file associated with an item in a project or stream. The new file name may include a directory path relative to the project (for example, `src/foo.c`.)

Parameters and qualifiers

- `<item-spec>` comprises:

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

`<item-id>`	identifies the item within the product.
`<variant>`	if omitted, the default (specified when the product was defined) is used.
`<revision>`	is ignored, all revisions within the project will be affected.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/WS_FILENAME=<ws_filename>`

  This is the new project file name for the item in the project.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command. If unspecified, the user's current project will be taken.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which this structural change to the project is to be related `In Response To`.

  Specify this optional qualifier if you want this structural change to the project to be recorded against the specified request(s). If path control has been enabled, this

qualifier is mandatory. If path control is not enabled, then the request(s) will be ignored.

NOTE A project file name (`<ws_filename>`) can also be assigned to an item when it is created (see Dimensions command CI on page 100). A particular Dimensions item can have different project file names in different projects.

Note on Areas

Whenever a new revision is added to a project, its stage is reset to DEVELOPMENT, and associated deployment areas and library cache areas are updated.

Constraints

This command can be run only on pending item revisions by a user with the appropriate management privileges for the project concerned, or by users whose privileges have been extended by a user with the appropriate management privileges because the users have a role on the product.

This constraint can, however, be relaxed using the Set Project Permissions (SWSP) command, as described on page 451.

SWS – Set Project/Stream Attributes

NOTE This command is deprecated. Use the UWA command instead.

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
[/BRANCH|/TRUNK]
[/[NO]AUTO_REV]
[/DESCRIPTION=<description>]
[/VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)]
[/DEFAULT_BRANCH=<branch-id>]
[/FILENAME=<report-filename>]
[/[NO]POPULATE]
[/[NO]PARALLEL_EXTRACT]
[/[NO]USE_LOCAL_STAGES]
```

Examples

```
SWS PROD:"WS MAIN DVL" /BRANCH

SWS PROD:"WS MAINT DVL" /VALID_BRANCHES=(maint,upgrade)
```

Parameters and qualifiers

- `<project-spec>` comprises:

  `<product-id>:<project-id>`

- `/BRANCH`

  Optional qualifier to adopt "branching" for the item revision scheme. This means that if an item-revision is at revision `maint#5`, and the users decide to stay on this `maint` branch, then subsequent revisions will be `maint#5.1`, `maint#5.2`, `maint#5.3` etc.

  This qualifier cannot be specified for streams.

- `/TRUNK`

  Optional qualifier to adopt "trunking" for the item revision scheme. This means that if an item-revision is at revision `maint#5`, and the users decide to stay on this `maint` branch, then subsequent revisions will be `maint#6`, `maint#7`, `maint#8` etc.

  This qualifier cannot be specified for streams.

- `/AUTO_REV`

  Optional qualifier to tell Dimensions to automatically generate a new revision each time an item-spec is edited/updated. If this is specified, Dimensions CM calculates revision strings automatically when you create a new item revision.

  This qualifier cannot be specified for streams

- `/NOAUTO_REV`

  Optional qualifier to tell Dimensions **not** to automatically generate a new revision each time an item-spec is edited/updated, and instead request the user to supply a revision.

  This qualifier cannot be specified for streams

- `/DESCRIPTION=<description>`

Optionally specify a new description to be attached to the project definition, thus replacing the one which was assigned when the project was created (with the DWS command).

- /VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)

  Identifies one or more branches–each previously defined in a Define (Item) Version Branches (DVB) command–that are to be valid for new item revisions created in this existing project. The list of valid branch-ids is added to the list (if any) specified previously for this project using DWS or SWS.

  To clear the list of valid branches, set /VALID_BRANCHES to ".".

  This list specifies the branches on which newly created item revisions can be placed.

  If the project attributes are set to have *one or more* valid branches, every *new* item revision in the project must use one of these branch-ids.

  If the project attributes are set to have *no* valid branches, new revisions with no branch-ids in them can continue to be used.

  This qualifier cannot be specified for streams

- /DEFAULT_BRANCH=<branch-id>

  selects, from the valid-list of branch-ids, the branch-id to be the default branch for the whole project. If a default branch-id is not defined, the first branch-id in the valid-list of branch-ids is taken as the default.

- /FILENAME=<report-filename>

  Specifies the output file name for a report.

- /[NO]POPULATE

  Populates the associated build areas.

- [/[NO]PARALLEL_EXTRACT]

  Stops you checking out (extracting) an item if a revision of that item is already checked out. This behaves in the same manner as "Allow Parallel Checkout" for item type options, but with respect to all item types on a per project basis. See "About Item Type Options" in the "Object Type Definitions" chapter of the *Process Configuration Guide* for a discussion of parallel check out, and other places in that guide for a discussion of parallel development in general.

- /[NO]USE_LOCAL_STAGES

  A deployment-related option.

  - (Default) /USE_LOCAL_STAGES

    Preserves an item revision's stage in the local project/stream. The stage is not affected even when stages in the GSL are associated with states in its lifecycle.

    NOTE: The same item revision can be at different stages in different projects/streams.

  - /NOUSE_LOCAL_STAGES

    Changing an item revision's stage in a project/stream also changes its stage in all projects/streams that do not use local stages. This is not a recommended best practice.

    Note: Not supported by Deployment Automation (DA).

Description

The SWS command is used to set (or reset) the attributes of an existing project or stream. Some qualifiers cannot be specified for streams

NOTE The /BRANCH, /TRUNK, /AUTO_REV and /NOAUTO_REV qualifiers may further be used to alter the options associated with the project. The permitted combinations of these qualifiers are:

```
SWS <project-spec> /BRANCH
SWS <project-spec> /TRUNK
SWS <project-spec> /AUTO_REV
SWS <project-spec> /NOAUTO_REV
SWS <project-spec> /BRANCH /AUTO_REV
SWS <project-spec> /BRANCH /NOAUTO_REV
SWS <project-spec> /TRUNK /AUTO_REV
SWS <project-spec> /TRUNK /NOAUTO_REV
```

Constraints

This command can be run only by a user with the PROJECT-STREAM-UPDATE privilege.

SWSP – Set Project Permissions

NOTE This command has been superseded by privileges for project operations.

Command deprecated.

TBI – Transfer Baseline In

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<tbo-id>
/PART=<part-spec>
/DEVICE=<device-id> or /DEVICE=NONE
/TAPE=<tape no.>
/VOLUME=<volume-id>
/CATEGORY=<replacement-category>
[/DIRECTORY=<directory>]
[/REPORT or /TOKEN]
[/CHANGE_DOC_IDS=(<request-type>,...) or /CHANGE_DOC_IDS=*]
[/WORKSET=<project-id>]
[/SOURCE_OS=WINDOWS or UNIX]
```

Example

```
TBI TB12AB /PART="PRODY:P123" -
/CATEGORY=MODULE /CHANGE_DOC_IDS=* /DEVICE="/dev/rst0" -
/TAPE="ta100" /VOLUME="tb100" -
/DIRECTORY="/usr/jones/work"
```

Example

```
TBI TB12AB /PART="PRODY:P123" /CATEGORY=MODULE -
/CHANGE_DOC_IDS=* /DIRECTORY="c:\usr\smith\work"
```

See the *System Administration Guide* for details.

TBO – Transfer Baseline Out

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<tbo-id>
/BASELINE=<baseline-spec>
/DEVICE=<device-id> or /DEVICE=NONE
/TAPE=<tape no.>
/VOLUME=<volume-id>
[/DESCRIPTION=<description>]
[/DIRECTORY=<directory>]
[/REPORT or /TOKEN]
[/CHANGE_DOC_IDS=(<request-type>,...) or /CHANGE_DOC_IDS=*]
```

Example

```
TBO TB12AB /BASELINE="PRODX:BL12AB" -
/DEVICE="/dev/rmt0h" /TAPE="ta100" /VOLUME="tb100" -
/DIRECTORY="/usr/smith/work" -
/CHANGE_DOC_IDS=(PR,CR) -
/DESC="12AB transfer - sources & requests"
```

Example

```
TBO TB12AB /BASELINE="PRODX:BL12AB" -
/DIRECTORY="c:\usr\smith\work" -
/CHANGE_DOC_IDS=(PR,CR) -
/DESC="12AB transfer - sources & requests"
```

See the *System Administration Guide* for details.

UA – Update Area

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
/NEW_NAME=<new-name>
[/DESCRIPTION=<area-description>]
[/NETWORK_NODE=<node-name>]
[/DIRECTORY=<HLQ/directory>]
[/TYPE=<area-type>]
[/STAGE=<stage-name>]
[/USER_LIST=(<user-or-group>,<another-user-or-group>,...)]
[/USER=<user-name or credential-set-name> [/PASSWORD=<password>]]
[/LIBRARY_CACHE_AREA=<area-name>]
[/[NO]FETCH_EXPANDED]
[/TRANSFER_SCRIPTS=<script-set>]
[/SCRIPT_PARAMETERS=(<name1=value1,name2=value2,...,)]
[/OWNER=<user-name> or <group-name>]
[/ADD] or [/DELETE]
[/STATUS=ONLINE or OFFLINE]
[/FILTER=<area-filter>]
```

Example

```
UA <area-name> /NETWORK_NODE=<host-machine> /DIRECTORY=<area-directory>
    /TYPE=WORK USER_LIST=(<user1>,<user2>,<user3>)
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area. Area names must be unique within the base database.

- `<new-name>`

  Specifies the new name for the area.

- `/DESCRIPTION=<description>`

  Specifies a description for the new area.

- `/NETWORK_NODE=<node-name>`

  Specifies the machine hosting the area.

- `/DIRECTORY=<HLQ/directory>`

  Specifies the directory, or PDS (partitioned data set), where the area is located.

  HLQ is a high-level qualifier; for example, MERVK.WORK. It is a common prefix for all data sets in the area such as MERVK.WORK.C, MERVK.WORK.CBL, and so forth.

- `/TYPE=<area-type>`

  Specifies the type of the area: WORK or DEPLOYMENT.

  Below is a mapping between Dimensions 9 and Dimensions 10 area types:

Dimensions 9	Dimensions 10
Development Area (working location)	Work Area
Managed Development Area	Deployment Area associated with stage DEVELOPMENT
Build Area associated with stage <XXX>	Deployment Area associated with stage <XXX>

- `/STAGE=<stage-name>`

  Applicable only to deployment areas. If the area type is DEPLOYMENT, this qualifier specifies the stage with which the deployment area is associated.

- `/USER_LIST=(<user-or-group>,<another-user-or-group>,...)`

  Specifies the list of users and groups that are granted the permission to work with this area. Applies only to areas of type WORK. If /ADD is specified, the specified users are appended to the area's user list. If /DELETE is specified, the specified users are deleted from the area's user list. If neither /ADD nor /DELETE is specified, the specified list of users replaces the area's user list.

- `/USER=<user-name or credential-set-name> [/PASSWORD=<password>]`

  Login information for the operating system user account or credential set that will own files transferred into the area. If you specify a credential set name you do not need to specify a password.

  For more information about credential sets see the *System Administration Guide*.

- `/LIBRARY_CACHE_AREA=<area-name>`

  Specifies a library cache area defined with the CLCA (Create Library Cache Area) command. During fetch operations (FI, FWI, FBI, FCDI, EI, EWI, EBI, ECDI, DOWNLOAD), Dimensions checks whether the library cache area associated with the current project/stream already contains a copy of of the requested file. If so, Dimensions copies the file from the library cache to the user file area instead of from the library itself, which improves performance when the connection between the item library node and the user's network is slow.

- `/[NO]FETCH_EXPANDED`

  Specifies that item header substitution variables will be expanded when item files are fetched to the area. Default is /FETCH_EXPANDED.

- `/TRANSFER_SCRIPTS=<script-set>`

  Applicable only to the DEPLOYMENT area type. Specifies the transfer script set. The script set contains a comma-separated list of the names of pre/post/fail transfer scripts in the following format:

(<pre-script>,<post-script>,<fail-script>)

If one of the scripts is undefined, CA uses $NONE as a placeholder. The pre-script is executed before items are transferred into an area, the post-script is executed after successful transfer of all items into an area, and the fail script is executed after a failed transfer of all items into an area.

- /OWNER=<user-name> or <group-name>

  Optional. Specifies the user or group that is to become the owner of the new area. If /OWNER is not specified, the user who created the area is set as its owner.

- /SCRIPT_PARAMETERS=(<name1=value1,name2=value2,...,nameN=valueN>)

  List of comma separated keyword and values to be passed as script parameters.

  Names in lowercase are converted to uppercase during execution. Names in templates must be written in uppercase, for example: %NAME1. %NAME2

  For details see the "Templating Language and Processor" chapter of the *Developer's Reference*.

  - To delete all script parameters:

    /SCRIPT_PARAMETERS=.

  - To specify an array of values:

    /SCRIPT_PARAMETERS=(...,A=[A1,"A2"," "])

- /STATUS=ONLINE or OFFLINE

  Applicable only to the DEPLOYMENT area type. Specifies the status of the area. If the area's status is ONLINE, the area may participate in file transfer operations. If the area's status is OFFLINE, the area is automatically excluded from any file transfer operations. If this qualifier is not specified, an area with status ONLINE is created.

- /FILTER=<area-filter>

  Applicable only to the DEPLOYMENT area type. Specifies the name of the area filter to be used when deploying files into this area.

Description

The UA command updates an area definition.

If an area is in use (that is, associated with a project), /NETWORK_NODE, /DIRECTORY, /TYPE may not be updated. If an area is not in use, changing /NETWORK_NODE or /DIRECTORY *does not* physically transfer files from the old location to the new location.

Constraints

To update a work area, you must have the Update Work Area Properties privilege. To update a deployment area, you must have the Update Deployment Area Properties privilege. These privileges are automatically granted to the owner of the area.

UBA – Update Build Area

NOTE This command is no longer available; use UA (Update Area) instead.

See the UA command.

UBDB – Update an Existing Base Database Entry

```
/BDB_NAME=<base_db_name>
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
[/SITE_NO=<site_no>]
```

This command enables you to edit registered base database entries in an installation's network administration tables. See the *System Administration Guide* for details.

UBLA – Update Baseline Attributes

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
[/ATTRIBUTES=(<attr1>=<value1>, <attr2>=<value2>,...)]
```

Example

```
UBLA PROD:"R M VERSION 2 FOR HP" -
/ATTRIBUTES=(TESTED_BY=GROUP5, AUTH_CODE=542)
```

Parameters and qualifiers

- `<baseline-spec>` comprises:

  `<product-id>:<baseline-id>`

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

  `<attrN>` is the Variable Name defined for one of the user-defined attributes for the baseline's type, which has also been declared as usable for this `<product-id>` and baseline's type.

  `<valueN>` is the substitution value to be given to this attribute.

  To add a new value to an existing multivalue attribute, use the following syntax:

  `/ATTRIBUTES=(<attr1>+=["<value1>"])`

  For example, to add "Charlie" to the multivalue attribute "NAME":

  `/ATTRIBUTES=(NAME+=["Charlie"])`

NOTE For full details about how to use the /ATTRIBUTES qualifier to append or prepend values to an existing multivalue attribute, see the UIA command on page 479.

Description

Subject to user authorization, each of the specified attributes is updated to the value given; or, if any of these attributes had no value previously set for this baseline, it is now set with the value given.

NOTE In the other commands (CBL, CMB and CRB) that assign values to user-defined baseline attributes, the `/ATTRIBUTES` qualifier is shown as optional. But it cannot be omitted if there exist any user-defined attributes, applicable to this baseline type, **whose Mandatory flag is Y, and for which there is no Default Value**.

Constraints

This command can be run in accordance with the attribute update rules defined by a user with the appropriate management privileges.

UBPROJ – Update a Dimensions Build Project

NOTE This command is no longer available. Use Dimensions Build to manage build projects.

```
Command no longer available.
```

UC – Update Request

```
<request-id>
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/DESCRIPTION=<desc-file>] and/or [/ADD_DESCRIPTION or
    /EDIT_ACTION_DESCRIPTIONS or /CANCEL_EDIT]
[/ATTACHMENTS=
([FILENAME=<file-id>, DESCRIPTION=<description-text>], ...)]
[/ADD_ATTACHMENTS=
([FILENAME=<file-id>, USER_FILE=<user-file>, DESCRIPTION=<description-
    text>], ...)]
[/DELETE_ATTACHMENTS=([FILENAME=<file-id>], ...)]
[/DETAILED_DESCRIPTION=<desc-file>]
[/EDIT_DETAILED_DESCRIPTION]
[/[NO]EXCLUSIVE_LOCK]
```

NOTE This command only functions for pending or held requests. Additionally, it cannot be run from Dimensions for z/OS.

Example

```
UC PROD_DR_28 -
/ATTRIB=(TITLE="QREL Subdir format problem")
/ATTACHMENTS=([/FILENAME=Figure3.jpg, DESCRIPTION="updated description
    text"])
/ADD_ATTACHMENTS=([/USER_FILE=c:\temp\newfile.jpg, DESCRIPTION="new
    page image"]
/FILENAME=Figure7.jpg])
/DELETE_ATTACHMENTS=([FILENAME=Figure1.jpg])
```

Parameters and qualifiers

NOTE The following qualifiers are mutually exclusive (you can only specify one of them):

- /ATTRIBUTES
- /DESCRIPTION
- /ADD_DESCRIPTION
- /EDIT_ACTION_DESCRIPTIONS
- /CANCEL_EDIT

- <request-id>

  The identity of the request to be modified.

- /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)

<attrN>	The Variable Name defined for one of the user-defined attributes for requests, which has also been declared as usable for the product and type specified in <request-id>.
<valueN>	The new **substitution** value to be given to this attribute.

- /ATTRIBUTES=(<attr1>+=<value1>,)

  Adds a new value to an existing multivalue attribute. For example, to add "Charlie" to the existing multivalue attribute "NAME":

  /ATTRIBUTES=(NAME+=["Charlie"])

- /ATTRIBUTES=(<attr1>++=<value1>,)

NOTE For details about using the /ATTRIBUTES qualifier to append or prepend values to an existing multivalue attribute, see the UIA command on page 479.

- /DESCRIPTION=<desc-file>

  Specifies a file containing the text body to be used as:

  - the detailed description of the request, if it is currently held; or
  - an action description if it has been saved (entered into system).

- /ADD_DESCRIPTION

  Calls an editor for the user to edit (or enter, if <desc-file> is omitted) the detailed description of the request (if it is held) or an action description (if it is saved).

NOTE Do not specify if you are running UC from Dimensions for z/OS.

- /EDIT_ACTION_DESCRIPTIONS

  Calls an editor to allow a user with a leader role to edit all the action descriptions entered since the request was last actioned.

NOTE Do not specify if you are running UC from Dimensions for z/OS.

- /CANCEL_EDIT

  Undoes the effects of a failed edit of a request.

- /ATTACHMENTS=([FILENAME=<file-id>, DESCRIPTION=<descriptiontext>],...)

  Changes the description of an existing attachment.

FILENAME=<file-id>	Specifies the name of the attachment.
DESCRIPTION=<descriptiontext>	The description of the attachment.

- `[/ADD_ATTACHMENTS=([FILENAME=<file-id>, USER_FILE=<user-file>, DESCRIPTION=<description-text>], ...)]`

  Adds a new attached file.

`FILENAME=<file-id>`	Specifies the file to be attached.
`USER_FILE=<user-file>`	Specifies the user file from where the attachment is to be loaded.
`DESCRIPTION=<descriptiontext>`	is the description of the attachment.

NOTE The `FILENAME` parameter must be unique on the request. The `DESCRIPTION` parameter is generated automatically if you omit it.

- `/DELETE_ATTACHMENTS=([FILENAME=<file-id>],...)]`

  Deletes an existing attached file. The FILENAME parameter must identify an existing attachment.

- `/DETAILED_DESCRIPTION=<desc-file>`

  Allows a user (subject to constraints below) to replace a request's detailed description with the contents of file <desc-file>. You cannot chose this option together with `/EDIT_DETAILED_DESCRIPTION` `i.e. they are mutually exclusive.`

- `/EDIT_DETAILED_DESCRIPTION`

  Calls an editor to allow a user (subject to constraints below) to edit the request's detailed description. In UNIX the interactive editor is specified by the setting of the symbol `DM_CHD_EDT` or `DM_CHD_EDT_SCRIPT`. You cannot chose this option together with `/DETAILED_DESCRIPTION=<desc-file>` i.e. they are mutually exclusive.

- `/EXCLUSIVE_LOCK`

  Specifies that the new request will be "locked" against any request (issue) replication "requests" from users located on other replication sites. A locked request is still available for users to work on normally if they are located on the "owning" replication site.

  There are primarily two conceptual working models that are used to provide issue replication—the delegation model and the request model. The delegation model works on the assumption that requests are created on one site and then "delegated" to another site to work on; while the request model follows the principle that a user on any site who sees a request that they want to work on can "request" that the responsibility for that request is handed over to them. See the *System Administration Guide* for details of issue replication.

  The default is `/NOEXCLUSIVE_LOCK` meaning that the new request can be "requested" from any authorized replication site.

Constraints

This command can be run only by a user with the appropriate management privileges or by users who have the minimum role to action the request for the current state to the next state. Your edit is, however, also subject to any update rules set by a user with the appropriate management privileges.

Requests that were created in a held state are not considered to have been "created" as far as Process Models where optional sensitive states or attributes have been set up ("electronic signatures") are concerned. The act of entering them into the system by actioning them out of the held state is considered the "authorization point" for such process models. This also applies to held requests that are updated at the held state (using the command UC) before being actioned on.

UCM – Update Code Metrics

```
[<itemSpec>|<projectPath;revision>]
[/WORKSET=<workset-spec> ]
[/ITEM_TYPE=<type-spec>]
[/USER_FILENAME=<listFile>]
[/PURGE]
[/REGENERATE]
```

Examples

`UCM`

updates code metrics for all latest revisions within the current project

`UCM "src/hello.cpp"`

updates code metrics for the last revision within the current or /WORKSET project/stream.

`UCM "src/hello.cpp;branch#2"`

updates code metrics for the specified revision.

`UCM /WORKSET=DMPROD:CM12_2`

updates code metrics for all latest revisions within the specified workset

`UCM /ITEM_TYPE=SRC`

updates code metrics for the last revision of the specified item type within the current or /WORKSET project/stream.

`UCM /USER_FILENAME="C:\Temp\file.lst"`

updates code metrics for revisions specified into the specified list file (containing <item-spec>s or/and <file-name><;revision>s separated by new-line)

Parameters and qualifiers

- `[<itemSpec>|<projectPath;revision>]`

  The specification of the item or the project pathname and revision number.

  If the revision part of <item-spec> is omitted, this means the latest revision within the current or /WORKSET project/stream.

  The revision may be specified as *, which means that all revisions of the item within the current or /WORKSET project/stream should have their code metrics updated.

  The variant part of the specification can be omitted if only one exists.

- `[/WORKSET=<projectName>]`

  If specified, only items in this project/stream will have their metrics updated.

- `[/ITEM_TYPE=<typespec>]`

  If specified, only items of this type will have their metrics updated.

  /ITEM_TYPE is ignored when one of the following parameters is used:

  - <item-spec>
  - <file-name>
  - /USER_FILENAME.

- `[/USER_FILENAME=<listFile>]`

A user specified file containing the list of items or files that are to have their line counts recalculated. Specifying this option allows you to process many items at once. This option is mutually exclusive to specifying `<itemSpec>|projectPath;revision>`.

- `[/PURGE]`

  If specified, the items will have their metrics values purged.

- `[/REGENERATE]`

  If specified, the metrics are regenerated (equivalent to UCM /PURGE followed by UCM.

Description

The UCM command recalculates the values of reporting metrics, such as the line count, for all eligible files in a project or stream, optionally restricted by an item type, and/or a list of items. This calculation is only made for text files and Unicode files.

From the current release of Dimensions CM, these metrics are updated whenever an item file is checked in or delivered to the repository. The UCM command enables you to calculate the metrics for existing items that have not yet been updated.

These metrics are used for reporting by the Serena ALM Dashboard. They are also included in the PCMS_ITEM_DATA published view; see the Reports Guide for details.

UCO – Update an Existing Contact

```
/CO_NAME=<contact_name>
```

This command enables you to update an installation codeset. See the *System Administration Guide* for details.

UCS – Update Credential Set

```
<credential-spec>
/USER =<userid> or
/PASSWORD=<password> or
/USER =<userid> /PASSWORD=<password>
```

This command enables you to update a credential set. See the *System Administration Guide* for details.

UCSJ – Unrelate Command from Schedule Job

```
<job-id>
[/CMD_UID]
```

Example: `UCSJ "MyJobName" /CMD_UID=4215769`

Parameters and qualifiers

- <job-id>

  Specifies the job-id.

- /CMD_UID.

  Specifies the command UID. Use the LSJ command with the parameter "/COMMANDS" to get the UID.

Description

Enables you to unrelate a command from a scheduled job.

Constraints

You must be the job originator, or have the privilege 'Manage Scheduled Jobs', to execute this command.

UCST – Update an Existing Codeset

```
/CDST_NUMBER=<codeset_number>
[DESCRIPTION=<description>]
```

This command enables you to update an installation's codeset. See the *System Administration Guide* for details.

UCU – Update Customer

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<name>
/LOCATION=<location>
/PROJECT=<project-spec>
[/COMMENT=<comment>]
[/CONTACT=<contact-details]
[/NEW_LOCATION=<location>]
[/NEW_NAME=<name]
[/NEW_PROJECT=<project-spec>]
```

Example

```
UCU "Brown Finances" /LOCATION="Manchester" -
/PROJECT="PAYROLL" /CONTACT="Mrs E Green" -
/NEW_LOCATION="Bristol"
```

Parameters and qualifiers

- `<name>`

  Specifies the **present** name for the customer details you wish to update.

- `/LOCATION=<location>`

  Specifies the **present** physical location for the customer details you wish to update.

- `/PROJECT=<project-spec>`

  Specifies the **present** project name for the customer details you wish to update.

- `/COMMENT=<comment>`

  for optionally customer contact details.

- `/CONTACT=<contact-details>`

  for optionally adding the new customer contact details.

- `/NEW_LOCATION=<location>`

  Specifies the customer's **updated** physical location.

- `/NEW_NAME=<name>`

  Specifies a **updated** name for the customer.

- `/NEW_PROJECT=<project-spec>`

  Specifies the **updated** project name for the customer details you wish to update.

Description

The Dimensions product allows you to maintain a list of customers and a record of which Dimensions releases have been sent to each customer.

The UCU command enables you to update a customer's details.

Constraints

The combination of customer name, location, and project-spec must be unique in the Dimensions database.

If any releases are related to a customer, you can only edit the contact and location information.

UFS – Edit an Existing File System

```
/FS_NAME=<file_system_name>
[DESCRIPTION=<description>]
```

This command enables you to edit specific file systems definitions for each registered installation operating system. See the *System Administration Guide* for details.

UGRP – Update Group

```
<group-name>
/DESCRIPTION="<description>"
```

Description

This command updates a group's properties. `<group-name>` is the name of the group, and `<description>` is the group's description to be updated.

Constraints

Only users with the appropriate management privileges can run this command.

UI – Revise Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE This command is not available for items that belong to a stream.

```
<item-spec>
[/ROOT_PROJECT=<project-spec>]
[/FILENAME=<file-name>]
[/USER_FILENAME=<user-filename>]
[/[NO]KEEP]
[/REVISION=<new-revision>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/STATUS=<status>]
[/COMMENT=<comment text>]
[/WORKSET=<project-spec>]
[/[NO]FORCE_UPDATE]
[/CODEPAGE=<code-page>|DEFAULT]
[/CONTENT_ENCODING=<file-encoding>]
[/NOMETADATA]
```

Example

```
UI PROD:"QUERY RELEASE".AAAA-SRC;1 /KEEP -
/CHANGE=(PROD_DC_16, PROD_DR_8) /STAT="UNDER TEST" -
/COMMENT="updated for ERB 58"
```

Parameters and qualifiers

- `<item-spec>` comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision in the project specified by /WORKSET. If /WORKSET is unspecified, the user's default project will be assumed.

- `/ROOT_PROJECT=<project-spec>`

  Comprises:

  `<product-id>:<project-id>`

  This optionally specifies the root project. Use this when the current project set via SCWS (or the project specified by the /WORKSET qualifier) occurs in more than one project tree.

- `/FILENAME=<file-name>`

Specifies the name of the project file name. If /ROOT_PROJECT is used to specify a the root project, /FILENAME is interpreted in the scope of that project.

The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

It may be omitted if `<item-id>` is specified.

- `/USER_FILENAME=<user-filename>`

  Specifies the name of the file holding the item in the user area.

  If omitted, it defaults to `<file-name>` – i.e. the file in the user area (the current directory) has the same name as that of the item's file in the item library.

- `/KEEP`

  Specifies that the user area file, which is normally deleted once its data has been placed under Dimensions control, is to be left intact.

- `/REVISION=<new-revision>`

  Specifies a new revision for the item. If `/WORKSET` is specified, the new revision will be placed in that project; otherwise, the new revision will be placed in the user's current default project.

  If new revision is omitted, Dimensions increments the current revision (the rightmost sub-field only), unless the item revision in `<item-spec>` is at its initial lifecycle state. In this case, the revision is unchanged.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

`<attrN>`	is 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 value to be given to this attribute.

- `</CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request to which the new item-revision is to be related **In Response To**.

- `/STATUS=<status>`

  Specifies the status of the new item-revision.

NOTE The status, if specified, must be one which would be valid if AI had been used separately. If omitted, the initial state (in the lifecycle defined for `<item-type>`) is assigned.

- `/COMMENT=<comment text>`

  comment text to explain the reason for the revision of this item revision. The comment text can be up to 1978 characters long, and can be made available within the item header.

- `/WORKSET=<project-spec> comprises:`

  `<product-id>:<project-id>`

and is optional.

If specified, the new item revision will be placed in that project.

If unspecified, Dimensions will place the new item revision in the user's current project.

- /FORCE_UPDATE

  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.

- CODEPAGE=<code-page>|DEFAULT

  Specifies the *code page* to be associated with the item. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The /CODEPAGE parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.
  /CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

  For further details concerning code pages and logical nodes see the *Network User's Guide*.

  You are advised to let the parameter default to the code page for the item type or the platform.

  The /CODEPAGE options available are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.

- <file-encoding>

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- /NOMETADATA

  This parameter disables creation and usage of metadata files in the local work area.

Description

You use the UI command when you want to create a new item revision using the contents of a file in your working directory. Revising an item is similar to using the Edit command in that you do not need to check out the item first. When you revise an item, the revision ID is changed according to your process model rules.

Your process model may require you to relate a request to the revised item.

If local metadata is present, it is used to revise the correct revision (the revision originally fetched by the user). If the user specifies an explicit revision to revise that differs from the revision originally fetched by the user, a warning is generated, but the operation succeeds. After a successful revise command in which /KEEP is specified, the local metadata is revised.

/NOKEEP causes the local metadata to be deleted as well as the real file.

If the original file was fetched with item header substitution turned on (and some substitutions performed), the UI command produces a warning message and fails.

If the original revision fetched does not exist in the current project, the UI operation fails with an error message.

NOTE UI results in the creation of a new revision in the project. If DEVELOPMENT deployment areas are in use, this command automatically updates the corresponding areas.

Constraints

This command can be run only by users who have one of the roles required to action the item from the initial lifecycle state to a new state.

UIA – Update Item Attributes

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
[/ATTRIBUTES=(<attr1>=<value1>, <attr2>=<value2>, ...) or
    =(<attr1>+=<value1>,) or
    =(<attr1>++=<value1>, ...)]
[/COMMENT=<comment text>]
[/WORKSET=<project-spec>]
[/FORMAT=<format>]
[/DESCRIPTION=<item-description>]
[/ORIGINATOR=<Dimensions-user>]
```

Examples

```
UIA PROD:"QUERY RELEASE".AAAA-SRC;1 -
    /ATTRIB=(DELIVERY_DATE=10-JUN-1998, AUTH_CODE=542) -
    /COMMENT="updated for ERB 58a"
UIA PAYROLL:"FORM1 FRM".AAAA-SRC;win2000#1 -
    /ATTRIB=(DEPLOY_ID+=["1.0"],)
UIA PAYROLL:"FORM1 FRM".AAAA-SRC;win2000#1 -
    /ATTRIB=(DEPLOY_ID+=["1.1"],["1.2"])
UIA PAYROLL:"FORM1 FRM".AAAA-SRC;win2000#1 -
    /ATTRIB=(DEPLOY_ID++=["1.0"],)
UIA PAYROLL:"FORM1 FRM".AAAA-SRC;win2000#1 -
    /ATTRIB=(DEPLOY_ID++=["1.2"],["1.1"])
```

Parameters and qualifiers

- `<item-spec>` comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see About the Command-Line Interface on page 12).

- `/FILENAME=<file-name>`

  Specifies the name of the file holding the item in the item-library. This qualifier cannot be used to rename the tiem, but rather is a method for identifying the item.

  It may be omitted if `<item-id>` is specified.

- /ATTRIBUTES=(<attrN>=<valueN>,<attrN>=<valueN>,...)

  <attrN> Specifies the Variable Name defined for one of the user-defined single-valued attributes for items. There are a total of 220 attributes that can be declared for all single-valued and multivalued attributes.

  <valueN> Specifies the substitution value to be given to this attribute.

  For example:
  /ATTRIB=(DELIVERY_DATE=10-JUN-2015,AUTH_CODE=542)

- /ATTRIBUTES=(<attrN>+=<valueN>,)

  <attrN>+ The '+' syntax enables you to *append* multi value attributes. For example:
  /ATTRIB=(DEPLOY_ID+=["1.0"],)
  /ATTRIB=(DEPLOY_ID+=["1.1"],["1.2"])
  The existing attribute DEPLOY_ID now has values in the following order:
  1.0
  1.1
  1.2
  If you use the '+' syntax on a single value attribute the following error message is displayed:
  Error: Attribute <ID> cannot be appended / prepended as it is not a Multi Value attribute.

- /ATTRIBUTES=(<attr1>++=<value1>,)

  <attrN>++ The '++' syntax enables you to *prepend* multi value attributes. For example:
  /ATTRIB=(DEPLOY_ID++=["1.0"],)
  /ATTRIB=(DEPLOY_ID++=["1.1"],["1.2"])
  The existing attribute DEPLOY_ID now has values in the following order:
  1.0
  1.1
  1.2
  If you use the '+' syntax on a single value attribute the following error message is displayed:
  Error: Attribute <ID> cannot be appended / prepended as it is not a Multi Value attribute.

- /ATTRIBUTES=(<attrN>=[+<valueN>,+<valueN>])

  <attrN>=[+<valueN>,] This syntax enables you to *append* to an existing multi value attribute rather than specify all the values again including the new ones. For example:
  /ATTRIB=(DEPLOY_ID=[+1,+2])

- /COMMENT=<comment text>

  Specifies comment text to explain the reason for the update of this item attributes. The comment text can be up to 1978 characters long.

- `/WORKSET=<project-spec> comprises:`

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

- `/FORMAT=<format>`

  Specifies the item data format. You use this qualifier to modify an item's format from, for example, that with which it was created. If item data formats have been assigned with the ADF command, the format specified must be one of those in the valid list of formats. You cannot update the format for an item revision at the same time as you update other user-defined attributes.

NOTE Only the PRODUCT-MANAGER can modify a FORMAT.

- `/DESCRIPTION=<item-description>`

  You cannot update the description for an item revision at the same time as you update other user-defined attributes.

NOTE descriptive name for itemOnly the PRODUCT-MANAGER can modify a DESCRIPTION.

- `/ORIGINATOR=<Dimensions-user>`

  Specifies a new Dimensions-user to be treated as the "originator" of the item. This qualifier is for use in scenarios where the historical originator (whose name will remain in the item's history log) is no longer a Dimensions user or is no longer actively involved in the project concerned. From now on, whenever the original originator would have had the item appear in their pending list or have had received e-mail, the "new" originator will become the originator as far as Dimensions is concerned. You cannot update the originator for an item revision at the same time as you update other user-defined attributes.

NOTE Only the PRODUCT-MANAGER can modify an ORIGINATOR.

Description

Subject to the constraints below, each of the specified attributes is updated to the value given; or, if any of these attributes had no value previously set for this item revision, it is now set with the value given. (It is not possible to unset an attribute, once it has been set for that revision.)

Attribute values for other revisions of the same item remain unaffected, except for any attribute(s) specified here whose All Revisions flag is **Y**. Such attributes are updated, or defined, with the value given here for all those other revisions as well.

NOTE In the other commands (CI, EI and UI) that assign values to user-defined item attributes, the /ATTRIBUTES qualifier is shown as optional. However, it cannot be omitted if the command is creating a new item or revision, and there exist any user-defined attributes, applicable to the item-type of the new revision, **whose Mandatory flag is Y, and for which there is no Default Value**. (Attributes with **Y** also for All Revisions do, in effect, always have a default value for all revisions after the first.)

Constraints

- UIA can—in all circumstances—be run only in accordance with the attribute update rules set by a user with the appropriate management privileges. With the above proviso, UIA can then be run by users who have, for each attribute specified, a role that is compatible with the value of the Role Check parameter. If the attribute update rules are defined for the item revision, UIA can be run by users who have, for each attribute specified, the role required by the update rule for that attribute. If there are no attribute update rules for the item revision, the item must be in the user's pending list or the user must have the appropriate management privileges.
- The value specified for each attribute must be consistent with its Data_type parameter.
- Each attribute must be one which has **I** for items as the Scope flag, and if a specific item-type is set for the attribute, the <item-type> specified in <item-spec> must match it.
- Any attribute whose Updateable flag is **N** cannot be specified in this UIA command. Values can be assigned to such attributes for a revision, only by the command which creates it (and only by the CI command if in addition the All Revisions flag is **Y**).
- If an attribute's Visible flag is **N**, no value can be assigned to it by this or any other standard Dimensions function.

UINS – Update an Existing Database Instance Entry

```
/DB_SERVICE=<base_db_instance>
/NN_NAME=<network_node_name>
```

This command enables you to edit registered database instance entries in an installation's network administration tables. See the *System Administration Guide* for details.

ULCA – Update Library Cache Area

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
/NEW_NAME=<new-name>
[/DESCRIPTION=<area-description>]
[/NETWORK_NODE=<node-name>]
[/DIRECTORY=<HLQ/directory>]
[/USER=<user-name or credential-set-name> [/PASSWORD=<password>]]
[/OWNER=<user-name> or <group-name>]
[/STATUS=ONLINE or OFFLINE]
```

Example

```
ULCA <area-name> /NETWORK_NODE=<host-machine> /DIRECTORY=<area-
    directory>
```

Parameters and qualifiers

- `<area-name>`

  Specifies the name of the area. Area names must be unique within the base database.

- `<new-name>`

  Specifies the new name for the area.

- `/DESCRIPTION=<description>`

  Specifies a description for the new area.

- `/NETWORK_NODE=<node-name>`

  Specifies the machine hosting the area.

- `/DIRECTORY=<HLQ/directory>`

  Specifies the directory, or PDS (partitioned data set), where the area is located.

  HLQ is a high-level qualifier; for example, MERVK.WORK. It is a common prefix for all data sets in the area such as MERVK.WORK.C, MERVK.WORK.CBL, and so forth.

- `/USER=<user-name or credential-set-name> [/PASSWORD=<password>]`

  Login information for the operating system user account or credential set that will own files transferred into the area. For more information about credential sets see the System Administration Guide.

- `/OWNER=<user-name> or <group-name>`

  Optional. Specifies the user or group that is to become the owner of the new area. If /OWNER is not specified, the user who created the area is set as its owner.

- `/STATUS=ONLINE or OFFLINE`

  Specifies the status of the area. If the area's status is ONLINE, the area may participate in file transfer operations. If the area's status is OFFLINE, the area is automatically excluded from any file transfer operations. If this qualifier is not specified, an area with status ONLINE is created.

Description

The ULCA command updates a library cache area definition.

If an area is in use (that is, associated with a project), /NETWORK_NODE and /DIRECTORY cannot be updated. If an area is not in use, changing /NETWORK_NODE or /DIRECTORY *does not* physically transfer files from the old location to the new location.

Constraints

To update a library cache area, you must have the Update Library Cache Area Properties privilege. This privilege is automatically granted to the owner of the library cache area.

ULCK – Unlock Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
workset <project-spec>
[/STREAM=<stream-id>]
```

Example

```
ULCK WORKSET PROD_X:TEST_WS
```

Parameters and qualifiers

- `<project-spec>` comprises:

  `<product-id>:<project-id>`

  The specified project must exist.

- `/STREAM=<stream-id>`

  This is the name of a stream for which you want to unlock a specific item.

Description

This command unlocks the project with project as a fixed parameter and `<project-spec>` a user-defined parameter. The unlocked state allows new Dimensions items to be added to the project.

Constraints

This command can be run only by a user with the appropriate management privileges for the project concerned.

UNC – Update an Existing Network Node Connection

```
/SERVER_NAME=<server_node_name>
/CLIENT_NAME=<client_node_name>
/CDST_NUMBER=<codeset_number>
/NWO_NAME=<network_object_name>
[/DIRECT_FILE_COPY]
[/FILE_COMPRESSION]
```

This command enables you to edit an existing installation network node connection. For details, see the *System Administration Guide*.

UNDO - Undo a Delivery

```
version
[/CHANGE_DOC_IDS=(<request1>,<2>)]
[/WORKSET=<product:spec>]
[/COMMENT=<text>]
```

Description

Rolls back the delivery changes to a stream or project. For example, you deliver changes but discover problems in the code and decide to remove the changes. UNDO creates a new changeset with the changes and preserves a full audit trail.

Assume that changeset 14 has this item revision:

```
f.txt revision #2
```

And that changeset 13 has the previous version of the same item:

```
f.txt revision #1
```

If you undo changeset 14, the revision in changeset 13 becomes the tip revision and a new changeset, 15, is created with the change:

```
f.txt revision #1
```

NOTE

- Requires the same privileges as DELIVER.
- The original request relationships are not undone.
- Only makes changes in a repository and does not update a local work area. To revert the changes in a work area, use the UPDATE command (see page 496).
- Undoes an entire changeset, not individual item revisions.
- After completing an undo, to synchronize your work area with the repository, run an update.
- To reverse an undo operation and return the original changes, run the UNDO command again.

Example

```
13
/CHANGE_DOC_IDS=(QLARIUS_CR_12,QLARIUS_CR_14)
/WORKSET=CONVERSE:CONVERSE_MAINLINE
/COMMENT=Last changes removed as bug was found in the Java code
```

Parameters and Qualifiers

- `version`

  Species the changeset ID (stream or project version) to undo.

  Default: the last changeset.

- `/CHANGE_DOC_IDS=(<request1>,<2>)`

  Specifies one or more requests to be related to the undo operation.

- `/WORKSET=<product:spec>`

  Specifies the stream or project to undo.

  Default: current stream or project.

- `/COMMENT=<text>`

  Add a comment.

Limitations

- Can only undo a single changeset each time you run the command.
- Cannot undo items from a project.
- If items in the changeset to be undone have more recent changes in a newer changeset, you cannot undo it.

UNN – Update an Existing Network Node

```
/NN_NAME=<network_node_name>
/OS_NAME=<operating-system-name>
/LOGICAL=<y|n>
[/PHYSICAL_NAME=<physical_node_name>]
[/CO_NAME=<contact-name>]
[/DESCRIPTION=<description>]
[/RSD_NAME=<resident_software_definition>]
```

This command enables you to edit an existing installation network node. See the *System Administration Guide* for details.

UNWO – Update an Existing Network Object

```
/PROTOCOL=<communication_protocol>
[/DESCRIPTION=<description>]
[/PROCESS=<network_object_process_name>]
/NWO_NAME=<network_object_name>
```

This command enables you to edit an existing installation network object. See the *System Administration Guide* for details.

UOS – Update an Existing Operating System

```
/OS_NAME=<operating_system_name>
```

This command enables you to edit an existing installation operating system. See the *System Administration Guide* for details.

UP – Update Design Part PCS

```
<part-spec>
/NEW_PCS=<new-pcs>
[/DESCRIPTION=<description>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
```

Example

```
UP PROD:"RELEASE MANAGEMENT".AAAA /NEW_PCS=1A -
/DESC="Release Support - Sun Test Version"
```

Parameters and qualifiers

- `<part-spec>`

  Comprises:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

  `<variant>` may be omitted if only one exists.

  `<pcs>` is ignored. On completion of this command, what is now the current PCS will become `CLOSED`.

- `/NEW_PCS=<new-pcs>`

  Specifies the new PCS of the design part, to be `OPENed` and become the current PCS.

- `/DESCRIPTION=<description>`

  This is a text description that applies to every PCS in the design part or design part variant. You can update the description for every PCS in the design part or design part variant; however you cannot define a unique description for each PCS.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

  `<attr1>` is the Variable Name defined for one of the user-defined attributes for design parts, which has also been declared as usable for this `<product-id>` and design part's category.

  `<valueN>` is the substitution value to be given to this attribute for the new PCS only.

Constraints

This command can be run only by the user who has the appropriate management privileges for the design part to which the variant being updated belongs.

UPA – Update Part Attributes

```
<part-spec>
[/ATTRIBUTES=(<attr1>=<value1>, <attr2>=<value2>,...)] or
    [/DESCRIPTION=<part-description>]
[/ORIGINATOR=<part-creator>]
```

Examples

```
UPA PROD:"ITEM OPS".AAAA;5
    /ATTRIBUTES=(TESTED_BY=GROUP5, AUTH_CODE=542)"

UPA PROD:"ITEM OPS".AAAA;5
    /DESCRIPTION="LIMITED TO GROUP 5 WITH CODE 542"
```

Parameters and qualifiers

- `<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored. Only the current PCS may be updated.

- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)`

`<attrN>`	is the Variable Name defined for one of the user-defined attributes for design parts, which has also been declared as usable for this `<product-id>` and design part's category.
`<valueN>`	is the substitution value to be given to this attribute.

- `/DESCRIPTION=<part-description>`

  If omitted, the original description is retained.

  /ATTRIBUTES and /DESCRIPTION are mutually exclusive.

- `/ORIGINATOR=<part-creator>`

  Changes who created the part.

Description

Subject to user authorization, each of the specified attributes is updated to the value given; or, if any of these attributes had no value previously set for this design part PCS, it is now set with the value given. (It is not possible to unset an attribute, once it has been set for that PCS.)

Attribute values for the earlier, closed PCSs of the same design part variant are never altered.

NOTE In the other commands (CP, CPV and UP) that assign values to user-defined design part attributes, the `/ATTRIBUTES` qualifier is shown as optional. However, it cannot be omitted if there exist any user-defined attributes, applicable to this design part's category, **whose Mandatory flag is Y, and for which there is no Default Value**.

Constraints

Only users with the appropriate management privileges can run this command.

UPDATE – Update Work Area

NOTE When a project is specified, or the user's current project or stream is a project, this command will behave in the same way as the DOWNLOAD command, for details see page 187.

```
[<file-spec> or /DIRECTORY=<directory-spec> or
    /USER_FILELIST=<filelist-file> or /USER_ITEMLIST=<itemlist-file>]
[/[NO]RECURSIVE]
[/[NO]EXPAND]
[/[NO]TOUCH]
[/[NO]OVERWRITE]
[/LOGFILE=<file-spec>
[/STREAM=<stream-id>;n]
[/USER_DIRECTORY=<directory-path>]
[/RELATIVE_LOCATION=<directory-spec>]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/BASELINE=<baseline-spec>]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/[NO]CANCEL_TRAVERSE]
[/CODEPAGE=<cp>]
[/[NO]QUIET]
[/[NO]VERBOSE]
[/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]
[/[NO]AUTO_MERGE]
[/ACCEPT=LOCAL | REPOSITORY]
[/ANCESTOR]
[/PERMS=KEEP|READONLY|WRITABLE]
[/VERSION=<changeset ID>]
[/UNDO]
```

Description

Use UPDATE to populate an empty work area or to incrementally update an existing work area with the content of a stream, project, or baseline. For projects, this command behaves the same as the DOWNLOAD command.

IMPORTANT! Dimensions 14.x and later: you can only use the UPDATE command to incrementally update an existing work area with the content of the stream that was originally used to populate the work area. Use the MERGE command to merge the contents of a stream or baseline into a work area owned by another stream. The MERGE command is not available for projects.

UPDATE compares the work area with the stream or baseline, and automatically applies any non-conflicting content and refactoring changes. The ability to detect and apply refactoring changes is the key difference from the DOWNLOAD command. Specifying /OVERWRITE will make UPDATE replace locally modified files with the corresponding versions from the stream or baseline.

UPDATE compares each item revision selected by the passed parameters with the corresponding on-disk files. If the disk file has been locally modified, or does not have Dimensions metadata, then the command issues a warning and skips the file. Otherwise, UPDATE overwrites the disk file with the content of the corresponding item revision. If an

update is made for an item that another user has locked, then this file will be made read-only by default.

Examples

- `UPDATE`

  Updates the associated work area with the tip of the current stream.

- `UPDATE /STREAM="build"`

  Copies the tip of stream "build" into the work area associated with that stream.

- ```
  UPDATE "C:\temp\build\build.mk" /TOUCH
      /BASELINE="PVCS:DM10 TIER1 FINAL"
  ```

  Assuming that the stream user work area is set to `C:\temp`, this command updates the file `C:\temp\build\build.mk` with the baseline item revision with file name `build\build.mk` from the PVCS:DM10 TIER1 FINAL baseline into the `C:\temp` directory. The modification time of the updated file is set to the current system time.

- ```
  UPDATE /DIRECTORY="build\include" /TOUCH
      /STREAM="PVCS:DM10"
  ```

  All files found in the stream directory `build\include` and in any directories below it are considered for update into the current working area. If the user has locally modified any matching files in the current working location, these files are not updated.

Parameters and Qualifiers

- `<file-spec>`

  Specifies the name of the file to be updated. The Dimensions `node::` syntax is also valid.

- `/DIRECTORY=<directory-spec>`

  Specifies a relative stream folder to be updated into the matching folder of the target work area.

- `/USER_FILELIST=<filelist-file>`

  Specifies a file containing a list of file names to be updated from the stream. Each file name must be on a separate line.

  File names may be specified as either absolute or relative paths. If the path is absolute, it is interpreted as a full work area path. If the path is relative, Dimensions obtains the stream path by mapping the file name to the operation root directory specified by one of the following:

  - The /USER_DIRECTORY qualifier.
  - The current working location specified by the last SCWS command.

  If such a mapping is not possible, the file name is ignored.

- `/USER_ITEMLIST=<itemlist-file>`

Specifies a file containing a list of item specifications to be updated from the stream. This qualifier allows you to efficiently update an arbitrary set of items from Dimensions CM using the complete item specifications. Each item specification must be on a separate line. There is no need to use double quotes with item specifications.

- /[NO]RECURSIVE

  If /DIRECTORY is specified and this qualifier is not present, all files that have not been modified in all directories beneath the one specified are copied to the work area. /NORECURSIVE specifies that only files at the specified directory level are updated.

  Default: /RECURSIVE

- /[NO]EXPAND

  Expands substitution variables.

  Default: /NOEXPAND

- /[NO]TOUCH

  Applies the system date/time to each file being transferred to the work area.

  Default: /TOUCH

- /[NO]OVERWRITE

  By default, UPDATE 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 fetched (files that have different <product>:<item-id>.<variant>-<type> pairs).

  If /OVERWRITE is specified, UPDATE overwrites such files with the content of the corresponding stream's item revisions.

  /OVERWRITE overrides the /NOADD qualifier. If /OVERWIRITE and /NOADD are both specified, /NOADD is ignored.

- /LOGFILE=<file-spec>

  Generates a log file at the specified file location that contains the results of all the individual Dimensions CM operations executed with this command.

- /STREAM=<stream-id>;n

  where:

  <stream> specifies a stream name.

  n specifies a version number.

  If you do not use this parameter, or specify a version number, the latest version of the stream is used.

- /USER_DIRECTORY=<directory-path>

  Specifies a destination work area root that is not the current working location. This directory must be empty or have been created by a previous invocation of the UPDATE command against the same stream. Using the UPDATE command to update work areas owned by other streams is no longer supported. You can specify the destination using the Dimensions node:: syntax. It can also be a path relative to the user working location.

  For example:

- The following command updates a stream into `C:\temp` regardless of what the current working location is:

  ```
  UPDATE /USER_DIRECTORY="C:\temp"
  ```

- The following command updates from a stream to the `/tmp` directory on the host "hostname":

  UPDATE /USER_DIRECTORY="hostname::/tmp"

- The following command updates from a stream into the `src` directory inside the `area_name` area:

  ```
  UPDATE /USER_DIRECTORY="area_name::src"
  ```

- `/RELATIVE_LOCATION=<directory-spec>`

  Specifies a project, stream, or baseline directory that is to be made the "virtual" root directory for the duration of this command. If this parameter is used the paths specified in <file-spec> or /DIRECTORY must be relative to the directory specified with /RELATIVE_LOCATION.

- `/FILTER=<filter-name>`

  Specifies a filter that will only retrieve files that satisfy the criteria specified in the area filter `<filter-name>`.

- `/USER_FILTER=<filter-file-spec>`

  Specifies the name of a local file containing the definition of a file filter to be used when getting files or checking in files. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502.

  Only files matching the filter (and not excluded by the filter) are copied to the work area when a user filter is specified.

- `/BASELINE=<baseline-spec>`

  Specifies a baseline from which to update the area. If specified, the files in the work area are updated from this baseline and not from the associated stream.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  Specifies that all content and refactoring changes associated with the related *In Response To* requests or child requests are applied to the target work area.

- `/CANCEL_TRAVERSE]`

  By default all requests related as dependent to the specified request are processed by this command. This qualifier forces the command to process only the specified request.

- `/CODEPAGE=<code-page>|DEFAULT`

  Specifies the code page to be associated with the items. The code page defines the method of encoding characters. It encompasses both the different ways characters are encoded on different platforms (EBCDIC on z/OS and ASCII on Windows and UNIX) and differences between human languages. Every item in Dimensions has a code page associated with it, this being defined or derived for the connection setting or an individual item.

  The `/CODEPAGE` parameter defaults to the code page specified when the connection between the database server and the logical node on which the user file resides was created. Whenever the item moves between platforms, for example, on a check-out

from the mainframe to a PC, if the code page for the target platform is different to the item's code page, Dimensions automatically converts the item.

/CODEPAGE is relevant only for text files, because whenever a text file is checked out or gotten, it must be in the right code page for the target platform, so that it displays correctly. Binary files are moved between platforms with no conversion.

For further details about code pages and logical nodes see Network Administration in the System Administration Guide.

You are advised to let the parameter default to the code page for the item type or the platform.

The /CODEPAGE options are:

<code-page>	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 installation directory. This file also provides more information about translation between code pages.
DEFAULT	Use the code page specified for the target node connection.

- /QUIET

  Only print critical messages.

- /VERBOSE

  Print additional information about the update process.

- [/EOL=WINDOWS|UNIX|DEFAULT|UNCHANGED|SHOW]

  Specifies the end-of-line handling that will be used when updating text files. The options are:

WINDOWS	Fetched text files follow the Windows convention for line termination, i.e. each line is terminated with a CR/LF character pair, regardless of the client operating system.
UNIX	Fetched text files follow the UNIX convention for line termination, i.e. each line is terminated with a single LF character, regardless of the client operating system.
DEFAULT	Uses the default Dimensions end-of-line handling mode, i.e. text files fetched to a Windows node have each line terminated with a CR/LF pair. Text files fetched to a UNIX node have each line terminated with a single LF character.
UNCHANGED	Text files are fetched as-is from the item library without any end-of-line processing.
SHOW	Display current EOL setting.

  See also "SET – Set DIR, PRINTER, OVERWRITE, CMD_TRACE, INFO, TIMEZONE, or EOL Environment" on page 428.

- /[NO]AUTO_MERGE

  If you specify this qualifier the UPDATE command tries to perform an automatic merge of conflicting file content when certain types of conflicts are detected. The automatic merge occurs in a temporary location and the result is copied to the work area if the merge completes without any conflicts.

For example, assume that the home work area of a stream contains revision 2 of a locally modified file, foo.c. The corresponding home stream contains revision 4 of foo.c. By default, the UPDATE command flags this as a conflict and leaves the locally modified file as is. If you specify /AUTO_MERGE the UPDATE command attempts to perform an automatic merge of the locally modified revision and the newer repository revision. If the merge succeeds the merged file is placed in the work area and its metadata updated to revision 4. The revision of foo.c in the work area is now the latest version and is the same as the repository.

- `/ACCEPT=LOCAL | REPOSITORY`

  If you specify this qualifier the UPDATE command uses the local or repository path of a file when resolving automatic merge conflicts that include file path renames or moves, in addition to file content conflicts.

  For example, assume that the home work area of a stream contains revision 2 of a locally modified file, foo.c. The corresponding home stream contains a renamed revision 4 of foo.c, that is now called bar.c. By default, the UPDATE command flags this as a conflict and leaves the locally modified file as is.

  If you specify /AUTO_MERGE the UPDATE command attempts to perform an automatic merge of the locally modified file and the newer repository revision. Because of the path conflict, if the merge succeeds the merged file is not placed in the work area unless you also specify the /ACCEPT qualifier:

  - If you specify /ACCEPT=LOCAL the merged file is copied to the work area under the local path of foo.c and a 'moved-from' property is added.
  - If you specify /ACCEPT=REPOSITORY the merged file is copied to the work area under the repository path of foo.c and the old work area file is deleted.

  If you do not specify /AUTO_MERGE the /ACCEPT qualifier is ignored.

- `/ANCESTOR`

  Explicitly specifies a stream version or a baseline to be used as the ancestor for a three-way merge. This is particularly useful when performing the initial merge of two unrelated streams or baselines or when redoing an erroneous merge.

  Syntax:

  `/ANCESTOR=<workset-spec.[;<stream version>]`

  or

  `/ANCESTOR=<baseline-spec>`

- `/PERMS=KEEP|READONLY|WRITABLE`

  Sets permissions for files that you update in a work area:

  `KEEP`: retains the current permissions

  `READONLY`: makes the files read only

  `WRITABLE`: makes the files writeable

  NOTE: If a file is not updated its permissions do not change.

- /UNDO /VERSION=<changeset ID>

  Removes the changes associated with the changeset specified by /VERSION and replaces them with the previous versions. For example, you have made changes locally and delivered them to a stream, but subsequently you need to revert the changes in the work area.

  - Only reverts the changes in a local work area, not the stream.
  - To synchronize the stream with the work area, deliver the changes after you have run this operation. This creates a new changeset.
  - Requires the same privileges as DELIVER.
  - Reverts an entire changeset, not individual item revisions.
  - The original request relationships are not removed.
  - Only works with streams, not projects.

  See also the UNDO command on page 488.

  Example

  Assume that you have the following modifications that you delivered, which created changeset CS_19:

  - webapphelp\help.css (moved from webapp\help.css)
  - contact_support.html (a new file)
  - config.dat revision #9 (new revision)

  After you revert to the previous changeset, CS_18, the work area content looks like this:

  - webapp\help.css (the previous path)
  - contact_support.html was deleted (a new file that was first delivered in CS_19)
  - config.dat revision #8 (the previous revision)

  To synchronize the stream with the work area, deliver the changes, which creates changeset CS_20 with this content:

  - webapp\help.css
  - config.dat revision #8

Inclusion/Exclusion Filters

Inclusion/exclusion filters can be specified via the /USER_FILTER qualifier to UPDATE or DELIVER. The structure of patterns matches the same as those used by area filters.

See the following xml schema:

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.serena.com/2008/filter">
 <xs:element name="filter">
   <xs:complexType>
    <xs:all>
      <xs:element name="includes" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="rule" maxOccurs="unbounded">
              <xs:complexType>
                <xs:all>
                  <xs:element name="file-pattern" type="xs:string"/>
                 <xs:element name="data-format"  type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="item-type"    type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="design-part"  type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="recurse"      type="xs:string"
                           minOccurs="0"/>
                </xs:all>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="excludes" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="rule" maxOccurs="unbounded">
              <xs:complexType>
                <xs:all>
                  <xs:element name="file-pattern" type="xs:string"/>
                 <xs:element name="data-format"  type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="item-type"    type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="design-part"  type="xs:string"
                           minOccurs="0"/>
                 <xs:element name="recurse"      type="xs:string"
                           minOccurs="0"/>
                </xs:all>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
 <filter>
   <includes>
     <rule>
       <file-pattern>**/*.jsp</file-pattern>
       <data-format>TXT</data-format>
       <item-type>QLARIUS:SRC</item-type>
       <design-part>QLARIUS:QLARIUS.A</design-part>
       <recurse>true</recurse>
     </rule>
   </includes>
   <excludes>
     <rule>
       <file-pattern>**/*.tmp</file-pattern>
     </rule>
     <rule>
       <file-pattern>**/*.bak</file-pattern>
     </rule>
   </excludes>
 </filter>
```

UPLOAD – Upload Local File or Directory

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

NOTE When issuing the DELIVER command, and a project is specified, or the user's current project/stream is a project, this command will invoked instead.

```
[<file-spec> or /DIRECTORY=<directory-spec>] or
    /USER_FILELIST=<filelist-file>]
[/BRANCH or /FORCE_TIP]
[/FORCE_CHECKIN]
[/CANCEL_UNCHANGED]
[/[NO]KEEP]
[/[NO]RECURSIVE]
[/[NO]RESTRICTED]
[/LOGFILE=<file-spec> or /SCRIPTFILE=<file-spec>]
[/ATTRIBUTES=(<name>=<value>, ...)]
[/CHANGE_DOC_IDS=(<request1>,<request2>,...)]
[/CODEPAGE=<code-page> or DEFAULT]
[/COMMENT=<text>]
[/DESCRIPTION=<description>]
[/PART=<part-spec>]
[/CONTRIBUTER_PROJECTS=(<project-id>, ...)]
[/ALL]
[/WORKSET=<project-spec>]
[/USER_DIRECTORY=<directory-path>]
[/RELATIVE_LOCATION=<directory-spec>]
[/[NO]CONFLICT_CHECK]
[/FILTER=<filter-name>]
[/USER_FILTER=<filter-file-spec>]
[/CONTENT_ENCODING=<file-encoding>]
[/NOIGNORE]
```

Examples

```
UPLOAD "C:\temp\work\FooBar.java" /COMMENT="Fixed a bug"
    /ATTRIBUTES=(Complexity="High") /NOKEEP
```

This command will upload the locally modified file `FooBar.java`. If any conflicting revisions are found in the repository, they will be reported and the upload will fail. The newly created revision will have the comment "Fixed a bug", and the revision's Complexity attribute will be set to "High".

```
UPLOAD /DIRECTORY="C:\temp\work" /COMMENT="Finished refactoring"
    /BRANCH /CHANGE_DOC_IDS=(PAYROLL_TDR_2)
```

All files found in the directory `C:\temp\work` and in any directories below it will be considered for upload. If the user has locally modified any file or explicitly checked out any file in that directory, it will be checked in. Checked-in revisions will be placed on a branch (no merge will occur) and be related to PAYROLL_TDR_2.

Parameters and qualifiers

- `<file-spec>`

  This parameter specifies the name of the file to be uploaded. (The Dimensions `node::` syntax is also valid.)

- `/DIRECTORY=<directory-spec>`

  This parameter specifies a directory path. The files in the directory are enumerated, and each one that has been been modified is uploaded.

  You can specify the directory using the Dimensions `node::` syntax.

- `/USER_FILELIST=<filelist-file>`

  This optional qualifier must specify a file containing a list of file names to be uploaded form the project or baseline. Each file name must be on a separate line.

  File names may be specified as either relative or absolute paths. If the path is absolute, it is interpreted as a full project or baseline file path; otherwise, Dimensions obtains the project or baseline path by mapping the file name to the operation root directory, which is the current working location as specified by the last SCWS command. If such a mapping is not possible, the file name is ignored.

- `/BRANCH`

  This qualifier specifies that if any conflicts occur, modified files will be uploaded to form a branch. No merge will take place.

- `/FORCE_TIP`

  This qualifier specifies that if any conflicts occur, modified files will be uploaded to form the "tip" (latest) version of the file. No merge will take place and there is the potential for other conflicting changes to be "hidden".

- `/FORCE_CHECKIN`

  This qualifier specifies that if there are any local files without Dimensions metadata that correspond to existing items in the repository, then these files will be upload to the latest version of the items. No merge will take place and there is potential for conflicting changes. By default, `/FORCE_CHECKIN` is not specified and these files are skipped. The `UPLOAD` command issues a warning for each such file, in this case.

- `/CANCEL_UNCHANGED`

  Specifies that a CIU (Cancel Item Update) command is performed if a user file does not differ from the base revision and Dimensions is configured to allow updates only if a real change is made.

- `/[NO]KEEP`

  If this qualifier is specified, the modified files will be removed from the local workspace after the upload completes. If /NOKEEP is not specified (or if /KEEP is specified), the local files will remain after the upload.

- `/[NO]RECURSIVE`

  If `/DIRECTORY` is specified and this qualifier is not present, all files that have been modified in all directories beneath the one specified are uploaded. `/NORECURSIVE` specifies that only files at the specified directory level are uploaded.

  Default: `/RECURSIVE`

- /[NO]RESTRICTED

  Specifies that UPLOAD will run in "restricted" mode and will not create new project items or project directories. Only checked out or locally modified item revisions will be updated. By default, UPLOAD runs in unrestricted mode, creating new directories and items as needed.

- /LOGFILE=<file-spec>

  This qualifier specifies that a log file be generated at the given file location containing the results of all the individual Dimensions operations executed through this command.

- /SCRIPTFILE=<file-spec>

  This qualifier specifies that a script file is generated at the given file location containing the individual Dimensions operations that would have been executed through this command. The script file contains commands that have the *same* affect as UPLOAD, thought the operations are not executed. The commands in the script file do not necessarily have the same qualifiers as the UPLOAD command.

- /ATTRIBUTES=(<name>=<value>, ...)

  The user defined attributes to set on the newly created revisions. All attributes specified must be valid for the item types created.

- CHANGE_DOC_IDS=(<request1>,<request2>,...)

  The requests for the items to be related to. The originally fetched versions will be related as "Affected", and the newly created versions will be "In Response To".

- /CODEPAGE=<code-page>

  The code page to be associated with the item.

- /COMMENT=<text>

  The comment to apply to all of the newly created item revisions.

- /DESCRIPTION=<description>

  The description to apply to all the newly created items.

- /PART=<part-spec>

  Design part specification in the form:

  ```
  <product-id>:<part-id>.<variant>;<pcs>
  ```

- /CONTRIBUTER_PROJECTS=(<project-id>, ...)

  If a working area contains files that originated from other projects that need to be added to the target project, this qualifier can be used to specify which projects to add content from.

- [/ALL]

  If this qualifier is specified, content originating from any project is also to be included when uploading files.

- /WORKSET=<project-spec>

  The project to which to upload the files. If this parameter is not specified, files are uploaded to the current session project.

- /USER_DIRECTORY=<directory-path>

  Use /USER_DIRECTORY=<directory-path> to specify an upload directory other than the current working location. For example, the following command uploads the project into C:\temp regardless of what the current working location is:

  ```
  UPLOAD /USER_DIRECTORY="C:\temp"
  ```

- /RELATIVE_LOCATION=<directory-spec>

  Specifies a project, stream, or baseline directory which is to be made the "virtual" project, stream, or baseline root directory for the duration of this command. If this parameter is given, then the paths given in <file-spec> or /DIRECTORY must be relative to the directory specified with /RELATIVE_LOCATION.

- /[NO]CONFLICT_CHECK

  By default, if neither /BRANCH nor /FORCE_TIP is specified, UPLOAD assumes that /CONFLICT_CHECK was specified and searches for unresolved merge conflicts that correspond to each item revision to be updated. If any unresolved merge conflicts are found, Dimensions issues a warning and does not update the corresponding item revision. This gives you a chance to resolve conflicts before the update.

  To not check for unresolved conflicts specify /NOCONFLICT_CHECK.

- /FILTER=<filter-name>

  Specifies that UPLOAD will create or update only files that satisfy the criteria specified in the <filter-name> area filter.

  An area filter is a regular expression following the same syntax as that used by the Dimensions GREP command.

- /USER_FILTER=<filter-file-spec>

  Specifies the name of a local file containing the definition of a file filter to be used when getting files or checking in files. The format of the filter file and a sample format definition is described in "Inclusion/Exclusion Filters" on page 502.

  Only files matching the filter (and not excluded by the filter) will be uploaded when a user filter is specified.

- /CONTENT_ENCODING=<file-encoding>

  Specifies the content encoding for new item revisions to be created. Supported encodings are the Microsoft codepages, the ISO-8859 variants (1–10), UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF32, UTF32BE, and UTF32LE.

- /NOIGNORE

  You can use ignore rules to exclude specific files, folders, and file types from uploads. To skip ignore rules and upload all files, specify /NOIGNORE. For details about using ignore rules see the *User's Guide*.

  Default: /IGNORE

Description

Uploads files and folders into a project in a repository. Cannot be used for streams. If the specified files have been locally modified (by the use of optimistic locking) or are checked out by the user who is invoking the command, the files are checked in.

If the original file was fetched with item header substitution turned on (and one or more substitutions performed), the UPLOAD command produces a warning message and fails.

UPNO – Update Part Numbers

```
<part-spec>
[/GENERIC_NO=<standard-no> [/NOCHECK]]
[/LOCAL_NO=<local-no>]
[/DESCRIPTION=<description>]
```

Example

```
UPNO PROD:"RELEASE MANAGEMENT".AAAB -
/LOCAL="HP 44" /DESC="Release Support HP Version"
```

Parameters and qualifiers

- `<part-spec>`

  Specifies the design part to be renumbered.

  It comprises: `<product-id>:<part-id>.<variant>;<pcs>`

  `<variant>` may be omitted if only one exists.

  `<pcs>` is ignored. A part number always applies to all PCSs.

- `/GENERIC_NO=<standard-no>`

  Specifies the modified standard part number to be allocated.

  It may be omitted, provided a `<local-no>` is specified.

- `/NOCHECK`

  Specifies the modified standard part number need not be in a range of numbers allocated to the product.

- `/LOCAL_NO=<local-no>`

  Specifies the modified local part-number to be allocated.

  It may be omitted, provided a `<standard-no>` is specified.

- `/DESCRIPTION=<description>`

  Specifies a new description to be given to the design part.

Constraints

Only users with the appropriate management privileges can run this command.

Each part category that is to use part numbers has to be enabled by the Process Modeler.

UPROD - Update a Dimensions Product

```
<product-id>
[/ATTRIBUTES=(<attribute_id>=<value>,...)]
[/DESCRIPTION=<description>]
[/SDA_APPLICATION=<DA-application-name>]
[/SDA_PROCESS=<DA-default-process>]
```

Example

```
UPROD
   PROD2
   /ATTRIBUTES=(site=dallas, priority=critical,country_orig=germany)
   /DESC="PROD Rel 2.0 Test Vehicle"
   /SDA_APPLICATION="Prod2App"
   /SDA_PROCESS="Vehicle.Auto"
```

Parameters and Qualifiers

- `<product-id>`

  Specifies the ID of the product to be updated.

- `/ATTRIBUTES=(<attribute_id>=<value>,...)`

  Specifies values for product level attributes.

- `/DESCRIPTION=<description>`

  The description of the product.

- `/SDA_APPLICATION=<DA-application-name>`

  Specifies the Deployment Automation (DA) application to be used for deployment during promotion and demotion.

  Specify an empty value ("") to use the Dimensions CM deployment model.

- `/SDA_PROCESS=<DA-default-process>`

  Specifies the default DA application process name to be executed when running a promotion.

UPROJ – Update a Dimensions Project

NOTE This command is no longer available. Please use UWA to update project attributes or Dimensions Build to manage build projects.

See the UWA command and the *Dimensions CM Build Tools User's Guide*.

UREG – Register User

```
<user-id>
[/WORKSET=<project-spec>]
[/[NO]PASSWORD_SAVE]
[/LOCALE=<locale>]
[/ATTRIBUTES=(site=<site>,
    group_id=<group-id>,Dept=<dept>,
    full_name=<full-name>,phone=<phone>,
    <attribute-id>=<value>,email_addr=<email-addr>)]
```

Description

This command is the same as CUSR.

You can use it to promote proxy or dormant users.

For details, see the *System Administration Guide*.

URP – Unrelate Design Part

```
<part-spec>
/FATHER_PART=<parent-part-spec>
```

Example

```
URP PROD:"LIBRARY CONTROL".AAAA -
/FATHER_PART=PROD:"RELEASE MANAGEMENT".AABB
```

Parameters and qualifiers

- `/FATHER_PART=<parent-part-spec>`[1]

  (both for the child design part and its obsolete `USAGE` parent part) comprises:

  `<prod-id>:<part-id>.<variant>;<pcs>`

  `<variant>` may be omitted if only one variant of that design part exists.

  `<pcs>` is ignored; the current PCS is always used.

Constraints

Only users with the appropriate management privileges can run this command.

URSD – Update an Existing Resident Software Definition

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
/RSD_NAME=<name_RSD>
```

This command enables you to edit an existing installation Resident Software Definition (RSD). See the *System Administration Guide* for details.

USUB – Unsubscribe from Notification Rule

```
<notification-id>
[/ROLES=(role1,role2,...)]
[/USER_LIST=(user1,user2,...)]
```

Example

```
USUB <rule-id> /USER_LIST=Smith
```

Parameters and qualifiers

- `<notification-id>`

  Name of the notification rule.

- `/ROLES`

  Specifies roles to unsubscribe from this notification rule.

- `/USER_LIST`

  Specifies users to unsubscribe from this notification rule.

Description

Unsubscribe users from a notification rule.

UUA – Update User Attributes

```
UUA
<user_name>
[/ATTRIBUTES=(<attribute-id>=<value>, <attribute-id>=<value>,...)]
[/[NO]PASSWORD_SAVE]
```

Description

This command enables you to update a user's attributes.

For details, see the *System Administration Guide*.

Constraints

Only users with the appropriate management privileges can run this command.

UWA – Update Project or Stream Attributes

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
[/BRANCH or /TRUNK]
[/[NO]AUTO_REV]
[/[NO]PARALLEL_EXTRACT]
[/DESCRIPTION=<description>]
[/VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)]
[/DEFAULT_BRANCH=<branch-id>]
[/ATTRIBUTES=(<attribute-value-list>)]
[/[NO]CM_RULES]
[/DEFAULT_CM_RULES]
[/[NO]PATH_CONTROL]
[/[NO]USE_LOCAL_STAGES]
```

Example

```
UWA PROD:"WS MAINT DVL" /VALID_BRANCHES=(maint,upgrade)
```

Parameters and qualifiers

- `<project-spec>`

  comprises `<product-id>:<project-id>`.

- `/BRANCH`

  Optional qualifier to adopt "branching" for the item revision scheme. This means that if an item revision is at revision `maint#5`, and the users decide to stay on this `maint` branch, subsequent revisions will be `maint#5.1`, `maint#5.2`, `maint#5.3`, and so forth.

  This qualifier cannot be specified for streams.

  /TRUNK

  Optional qualifier to adopt "trunking" for the item revision scheme. This means that if an item revision is at revision `maint#5`, and the users decide to stay on this `maint` branch, subsequent revisions will be `maint#6`, `maint#7`, `maint#8`, and so forth.

  This qualifier cannot be specified for streams.

- `/AUTO_REV or /NOAUTO_REV`

  Optional qualifier to tell Dimensions whether to automatically generate a new revision each time an item is edited/updated. If this is specified, Dimensions CM calculates revision strings automatically when you create a new item revision. If not, Dimensions requests you to supply a revision number.

  This qualifier cannot be specified for streams.

- `PARALLEL_EXTRACT`

  Determines whether you can check out (extract) an item if a revision of that item is already checked out. This behaves in the same manner as "Allow Parallel Checkout" for item type options, but with respect to all item types on a per project basis. See "About Item Type Options" in the "Object Type Definitions" chapter of the *Process Configuration Guide* for a discussion of parallel check out, and other places in that guide for a discussion of parallel development in general.

  This qualifier cannot be specified for streams.

- /DESCRIPTION=<description>

  An optional new description to be attached to the project definition, thus replacing the one which was assigned when the project was created (with the DWS command).

- /VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)

  Identifies one or more branches—each previously defined in a Define (Item) Version Branches (DVB) command—that are to be valid for new item revisions created in this existing project. The list of valid branch ids replaces the list (if any) specified previously for this project using DWS or UWA.

  To clear the list of valid branches, set /VALID_BRANCHES to ".".

  This list specifies the branches on which newly created item revisions can be placed.

  If the project attributes are set to have one or more valid branches, every new item revision in the project must use one of these branch ids.

  If the project attributes are set to have no valid branches, new revisions with no branch ids in them can continue to be used.

  This qualifier cannot be specified for streams.

- /DEFAULT_BRANCH=<branch-id>

  Selects, from the valid list of branch ids, the branch id to specify the default branch for the whole project. If a default branch id is not defined, the first branch id in the valid list of branch ids is used as the default.

  This qualifier cannot be specified for streams.

- /ATTRIBUTES=(<attribute-value-list>)

  Standard Dimensions user-defined attributes qualifier.

- /[NO]CM_RULES

  For a project, /CM_RULES specifies that CM rules are fully validated for the supplied request type and item type. For details of CM rules see *Process Configuration Guide* section.

  For a stream, /CM_RULES specifies that a request is required when creating new item revisions in the stream. Note that this option does not fully validate the CM rules for the item type and request type.

- /DEFAULT_CM_RULES

  For a stream, specifies that CM rules are fully validated for the supplied request type and item type. Specifying /NOCM_RULES turns this option off.

- /[NO]CM_RULES

  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_CM_RULES

  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. For details of CM rules see *Process Configuration Guide* section *About Change Management Rules*.

- /[NO]PATH_CONTROL

Specifies whether a request is required to perform refactoring operations in this project.

- /[NO]USE_LOCAL_STAGES

  A deployment-related option.

  - (Default) /USE_LOCAL_STAGES

    Preserves an item revision's stage in the local project/stream. The stage is not affected even when stages in the GSL are associated with states in its lifecycle.

    NOTE: The same item revision can be at different stages in different projects/streams.

  - /NOUSE_LOCAL_STAGES

    Changing an item revision's stage in a project/stream also changes its stage in all projects/streams that do not use local stages. This is not a recommended best practice.

    Note: Not supported by Deployment Automation (DA).

Description

This command or stream. Some qualifiers do not apply to streams.

it replaces SWS. It includes support for the /CHILD_ORDER qualifier.

Constraints

Only users with the PROJECT-STREAM-UPDATE privilege can run this command.

VLSJ – View Log of Schedule Job Execution

/HISTORY_UID

Example: `VLSJ /HISTORY_UID=4215941`

- /HISTORY_UID

  Specifies the history UID. Use the LSJ command with the parameter "/JOB_HIST" to get the history UID.

Description

Enables you to view the execution log for a scheduled job.

WRC – Withdraw a Release from a Customer

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<release-spec>
/CUSTOMER=<name>
/LOCATION=<location>
/PROJECT=<project-spec>
```

Example

```
WRC PROD:"R M 2.0 FOR HP" /CUSTOMER="Brown Finances -
/LOCATION="Bristol" /PROJECT="PAYROLL"
```

Parameters and qualifiers

- `<release-spec>`

  Specifies the releases-spec, which comprises:
  `<product-id.:<release-id>`

- `/CUSTOMER=<name>`

  Specifies the customer's name.

- `/LOCATION=<location>`

  Specifies the customer's physical location.

- `/PROJECT=<project-spec>`

  Specifies the project name.

Description

The Dimensions product allows you to maintain a list of customers and a record of which Dimensions releases have been sent to each customer.

The WRC command enables you to remove the record of the fact that a release has been supplied to a specific customer.

Constraints

The combination of customer name, location, and project-spec must be unique in the Dimensions database.

You cannot forward the same release to a customer twice.

XABC -Remove Area from Build Configuration

```
XABC <area-name>
/WORKSET=<project or stream spec>
/BUILD_CONFIG=<configuration spec>
```

Example

```
XABC area-component1
/WORKSET=build-project-component1
/BUILD_CONFIG=build-config-component1
```

- `<area-name>`

  Specifies the name of the build area to be removed.

- `/WORKSET=<project or stream spec>`

  Specifies the project or stream containing the build configuration.

- `/BUILD_CONFIG=<configuration-spec>`

  Specifies the build configuration containing the area to be removed.

Description

Removes a build area from a build configuration. The configuration must be checked out first, see:

- "ECFG – Extract (Check Out) Build Configuration" on page 224.
- "RCFG – Return (Check In) Build Configuration" on page 358.

For more information about using Dimensions Build see *Dimensions CM Build Tools User's Guide*.

XAWS – Unrelate Area from Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<area-name>
/WORKSET=<project-spec>
[/[NO]KEEP]
```

Example

- `XAWS DM10-WIN32 /WORKSET="PVCS:DM10"`

  Unrelates the `DM10-WIN32` area from the `PVCS:DM10` project. Does not delete any files.

Parameters and qualifiers

- `<area-name>`

  The name of the area that you want to unrelate from the specified project.

- `/WORKSET=<project-spec>`

  The project from which you want to unrelate the specified area.

- `/KEEP or /NOKEEP`

  Specifies whether to keep files corresponding to the previously deployed item revisions in the area or delete them. By default, existing area files will not be deleted (/KEEP is default). If /NOKEEP is specified, files corresponding to item revisions from the specified project (including item revisions in child collections) previously transferred into this area will be deleted.

Description

This command breaks the relationship between an area and a project.

Constraints

Only users with the "Assign Deployment Areas to Project" privilege can run this command.

XBBL – Unrelate Baseline from Baseline

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<child-baseline-spec>
/BASELINE=<parent-baseline-spec>
```

Example

```
XBBL <child-baseline-spec> /BASELINE=<parent-baseline-spec>
```

Parameters and qualifiers

- `<child-baseline-spec>`

  Specifies the child baseline in the parent-child relationship.

- `/BASELINE=<parent-baseline-spec>`

  Specifies the parent baseline in the parent-child relationship.

Description

This command breaks the relationship between a baseline and a parent baseline. The parent baseline must be at the initial lifecycle state.

Constraints

Only users with the appropriate management privileges can run this command.

XBCD – Unrelate Baselines from Requests

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/AFFECTED or /IN_RESPONSE_TO or /INFO]
```

Example

```
XBCD "PAYROLL:ACME_2.1" -
/CHANGE_DOC=("PAYROLL_TDR_1","PAYROLL_TDR_2") /INFO
```

Parameters and qualifiers

- `<baseline-spec>` comprises:

  `<product-id>:<baseline-id>`

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>` identifies a request from which the specified baseline is to be unrelated.

- `/AFFECTED or /IN_RESPONSE_TO or /INFO`

  Specifies the type of relation to be deleted between the given baseline and the associated requests. The qualifiers are mutually exclusive.

  The default is `/AFFECTED`.

Constraints

XBCD is restricted to merge, release, and revised baselines. If it is run with respect to an archive or design baseline, then an appropriate error will be returned.

XBCD will only work successfully if you have both the baseline and requests in your pending list. If you specify an `/INFO` relationship, however, then this pending list restriction is relaxed.

There is no support for phase rules or change management rule enhancements within the context of baseline to request relationships.

Only the three relationship types – Info, Affected, and In Response To – are supported. There is no support for user-defined relationship types.

XBWS – Unrelate Baseline from Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<baseline-spec>
/WORKSET=<project-spec>
```

Example

```
XBWS <child-baseline-spec> /WORKSET=<parent-project-spec>
```

Parameters and qualifiers

- `<baseline-spec>`

  Specifies the child baseline in the parent-child relationship.

- `/WORKSET=<project-spec>`

  Specifies the parent project in the parent-child relationship.

Description

This command breaks the relationship between a baseline and a project.

Constraints

Only users with the appropriate management privileges can run this command.

XCCD – Unrelate Requests from Request

```
<request-id>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/DEPENDENT or /INFO ]
```

Example

```
XCCD PROD_CN_4 /CHANGE=PROD_DR_25
```

Parameters and qualifiers

- `<request-id>`

  This is the identity of the request which is the parent in the relationship to be removed.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

  `<requestN>` is the identity of a request which is a child in the relationship to be removed.

- `/DEPENDENT or /INFO`

  Specifies the type of relation to be removed from between the given requests. The two qualifiers are mutually exclusive.

  The default is `/DEPENDENT`.

Constraints

This command can be run by users appropriate management privileges on the product or products owning the specific request concerned. Such users must have the parent request in their Pending List.

However, a user with the appropriate management privileges can set up the Process Model to specify that no request relationships can be modified for certain request types.

XICD – Unrelate Item from Requests

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/AFFECTED or /IN_RESPONSE_TO or /INFO]
[/WORKSET=<project-spec>]
```

Example

```
XICD PROD:"QUERY RELEASE".AAAA-SRC;1 -
/CHANGE_DOC=(PROD_DR_25, PROD_DC_16)
```

Parameters and qualifiers

- `<item-spec>` comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see About the Command-Line Interface on page 12).

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...)`

`<requestN>`	identifies a request from which the specified item is to be unrelated.

- `/AFFECTED or /IN_RESPONSE_TO or /INFO`

  Specifies the type of relation to be deleted between the given item and the requests. The qualifiers are mutually exclusive.

  The default is `/AFFECTED`.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Constraints

This command can be run only by users who have the request in their pending list or by a user with the appropriate management privileges.

XII – Unrelate Item from Item

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<src-item-spec>
<dst-item-spec>
/RELATIONSHIP=<relationship-id>
[/FILENAME=<src-filename>]
[/FILENAME=<dst-filename>]
[/WORKSET=<project-spec>]
```

Example

```
XII "PROD_X:INTERFACE_C.AAAA-C;1" -
"PROD_X:INTERFACE_H.AAAA-H;1" -
/RELATIONSHIP="INCLUDES"
```

Parameters and qualifiers

- `<src-item-spec>`

  Specifies the item from which the relationship instance will be removed.

- `<dst-item-spec>`

  Specifies the item at the other end of the relationship to be removed.

- `/RELATIONSHIP=<relationship-id>`

  Specifies the relationship type as defined by the DIR command.

  Specify the BLD_PREDICTED qualifier to remove a predicted build made-of relationship between items.

- `/FILENAME=<src-filename>`
  `/FILENAME=<dst-filename>`

  Optional qualifiers that further specify the source item and the destination item by giving their file names.

- `/WORKSET=<project-spec> comprises:`

  `<product-id>:<project-id>`

  This optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the project.

Description

The XII command is used to remove instances of relationships created by the RII command (on page 383). The source and destination item types must be consistent with the relationship definition.

This can also be used to delete a predicted build *madeof* relationship between items, when specifying the /RELATIONSHIP=BLD_PREDICTED qualifier. Note that only predicted relations can be removed, BLD_ACTUAL relationships cannot be removed.

XIP – Unrelate Item from Part

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<item-spec>
[/FILENAME=<file-name>]
/PART=<part-spec>
[/WORKSET=<project-spec>]
```

Example

```
XIP PROD:"QUERY RELEASE".AAAA-SRC -
/PART=PROD:"RELEASE MANAGEMENT".IBM
```

Parameters and qualifiers

- `<item-spec>` comprises:

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

`<item-id>`	may be omitted if `<file-name>` is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are unrelated from the specified design part.

- `/FILENAME=<file-name>`

  Specifies the name of the project file name.

  The project file name identifies the relative path (directory plus file name) from the working location to the item to be used from the current project. The project file name for the same item may *differ* between projects; for example, `src/hello.c`, `hello.c`, or `src/build/hello.c`.

  It may be omitted if `<item-id>` is specified.

- `/PART=<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/WORKSET=<project-spec>` comprises:

  `<product-id>:<project-id>`

  this optionally specifies the project to be used for this command: failing this, the user's current project will be taken.

  All Item revisions, regardless of whether they are in the project, will be affected.

Constraints

This command can be run only by users who have the item revision in their pending list or have the appropriate management privileges.

XPCD – Unrelate Part from Requests

```
<part-spec>
/CHANGE_DOC_IDS=(<request1>,<request2>,...)
[/PENDING]
```

Example

```
XPCD PROD:"RELEASE MANAGEMENT".AAAA -
/CHANGE_DOC=PROD_DR_26
```

Parameters and qualifiers

- `<part-spec>` comprises:

  `<product-id>:<part-id>.<variant>;<pcs>`

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/CHANGE_DOC_IDS=(<request1>,<request2>,...`

`<requestN>`	is the identity of a request from which the specified design part is to be unrelated.

- `/PENDING`

  This option runs the PEND command after the completion of the XPCD command.

Constraints

- This command can be run only by the user who has the request in their Pending list or by a user with the appropriate management privileges.
- This command cannot be used with the secondary catalog list.
- This command cannot be run if the request is at its end (closed) lifecycle state – even by a change-manager; however, a change-manager can action it back to an earlier lifecycle state perform the unrelate operation, and then action the request back to its closed state.

XRCD – Unrelate Requirement from Request

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
XRCD /CHANGE_DOC_ID=<request_id> /REQUIREMENT_ID=<requirement>
/CONTAINER_NAME=<container_name> /RM_PROJECTNAME=<project_name>
/RM_DBNAME=<dbname> /RM_URL=<url>
```

Example

```
XRCD /CHANGE_DOC_ID=REPX_CR_114
   /REQUIREMENT_ID= Marketing_Requirements.MRKT_000020;79
   /CONTAINER_NAME="Engineering Requirements"
   /RM_PROJECTNAME=QLARIUS_RM /RM_DBNAME=RM10
   /RM_URL="http://hostname/rtmBrowser"
```

Description

This command breaks the link between a requirement and a request.

- `<request-id>`

  Specifies the request to be unrelated from the requirement.

- `<requirement>`

  Specifies the requirement to be unrelated from the request and comprises:

  ```
  <class_name>.<puid>;objId
  ```

  For example:

  ```
  Marketing_Requirements.MRTK_000020;4
  ```

- `<container_name>`

  Specifies the name of the originating Dimensions RM baseline, collection, document, or snapshot for the requirement (multiple "versions" of a requirement cannot be related to a single Dimensions CM request).

- `<project_name>`

  Specifies the Dimensions RM project name.

- `<dbname>`

  Specifies the Dimensions RM database name.

- `<url>`

  Specifies the Dimensions RM Browser URL

NOTE You can only specify Dimensions RM requirements if you have installed the Dimensions RM integration, have associated Dimensions CM projects with Dimensions RM containers, and have associated Dimensions CM products with Dimensions RM projects. See the "Miscellaneous Functions" chapter in the Dimensions CM *User's Guide*, the *Dimensions CM-Dimensions RM ALM Integration Guide*, and the Dimensions RM documentation for details.

Constraints

Only users with the privilege "Perform Requirement Related Operation" (PRODUCT_REQUIREMENTMAN) can run this command.

XREG – Unregister User

```
<user-id>
[/[NO]KEEP]
```

Description

This command is the same as DUSR.

For details, see the *System Administration Guide*.

XWCD – Unrelate Project from Request

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<project-spec>
/CHANGE_DOC_IDS=(<request1,<request2>,request3>,...)
```

Example `XWCD PAYROLL:PRJ_INITIAL /CHANGE_DOC_IDS=(PAYROLL_CR_21)`

Description

The example above unrelates the PAYROLL_CR_21 request from the PAYROLL:PRJ_INITIAL project.

Constraints

Only users with the appropriate management privileges can run this command.

XWWS – Unrelate Project from Project

NOTE This command is not available in the Issue Management–only licensed version of Dimensions.

```
<child-project-spec>
/WORKSET=<parent-project-spec>
```

Example

```
XWWS <child-project-spec> /WORKSET=<parent-project-spec>
```

Parameters and qualifiers

- `<child-project-spec>`

  Specifies the child project in the parent-child relationship.

- `/WORKSET=<parent-project-spec>`

  Specifies the parent project in the parent-child relationship.

Description

This command breaks the relationship between two projects

Constraints

Only users with the appropriate management privileges can run this command.

Chapter 3

Standalone Dimensions Utilities

Introduction	542
General Information	543
Metadata Utility	544
Actioning Requests by Date or Attribute Value	551
Sending Reminders of Pending Lists	552
Encryption	554
PRCS–RCS-like Front End to Dimensions	556
PSCCS–SCCS-like Front End to Dimensions	565

Introduction

This chapter identifies various miscellaneous Dimensions® CM standalone utilities, some of which are described here and some of which are described in referenced related documents. Certain of these utilities are intended for use by users with the role of CHANGE-MANAGER, PRODUCT-MANAGER or PART-MANAGER only – these will be identified as such when discussed and for ease of reference will be referred to as change-managers, product-managers and part-managers respectively. Also, certain of the utilities are only available for specific operating systems – these too will be identified where appropriate.

The utilities discussed in this chapter are:

- **dmmeta** This utility enables you to view and edit the metadata in a work area that is associated with a project or stream. You can also use it to update metadata from releases prior to Dimensions CM 12.2 to the current format.
- **dm_auto_action** This utility is used to action requests once a date (attribute value) has passed.
- **dm_full_mail** and **dm_incremental_mail** These utilities are used to send users periodic reminders of requests in their pending lists.
- **dmpasswd** This utility ensures that Dimensions base database names, connection strings, and passwords are encrypted before being written to disk (they are stored in the Dimensions file `registery.dat` located in the `dfs` sub-directory).
- **prcs** [UNIX only] This utility provides a RCS-like front end to the version control commands of Dimensions.
- **psccs** [UNIX only] This utility provides an SCCS-like front end to the version control commands of Dimensions.

The auto-action and the two mail utilities have been designed to be executable as time-triggered automatic jobs, and some details on the use of UNIX's `crontab` are included.

The following standalone utilities are discussed in the indicated related documents.

- **download** This utility performs a download of a list of specified files under Dimensions into a target directory. See the *System Administration Guide*.
- **dmdba** This is the Dimensions interactive DBA Tool. This tool is used to create and administer base databases and Dimensions published views. See the related document *System Administration Guide*.
- **dm_mtu** This utility is used by Dimensions ART to read and write Archive and Transfer Baseline Out volumes. See the *System Administration Guide*.
- **pdiff** (only for use by change managers and product-managers). This utility is used to import/export data via the Dimensions Data Interchange File Format into/from a specified Dimensions product. See the *System Administration Guide*.
- **pm_label_branch** (only for use by tool-managers for the base databases concerned). This Dimensions Replicator utility enables tool managers to move items that are currently on a nameless branch to be placed on a named branch. See the *System Administration Guide*.
- **replicator** (only for use by tool managers for the base databases concerned). This utility enables the tool manager to initiate the Dimensions replication process at the command prompt. It can be executed manually or automatically via a scheduler such as UNIX `cron`. See the *System Administration Guide*.

- **upload** This utility performs an upload of a list of specified files in a user directory into Dimensions. See the *System Administration Guide*.

General Information

All these utilities are run as independent programs from the operating system prompt (or from a command script file). Some require the user to have performed a standard Dimensions user's login, which sets up the environment required for all Dimensions processing. The utilities all reside in the directory specified by the environment variable DM_PROG – which the standard login will include in the directory search path.

The syntax of each utility is explained under separate headings below, but the following general points are best explained in detail now:

Case Translation

Lower-case letters can be included in the values of parameters, and will automatically be interpreted as the equivalent in upper-case. This applies to all parameters, except those for which it is specifically stated that they are case-sensitive.

Wild Card Characters

In several parameters a range of possible values can be indicated by including a % (percent) character, which is interpreted as matching any zero or more characters. *This applies only to parameters for which it is stated that wild card % may be used*.

In other parameters a *null string* can be used to imply all possible values; a null string is specified as "" (two consecutive double-quote characters). This applies only to parameters for which it is stated that a null string may be used.

Execution Authority: Change-Manager or Tool-Manager

If the parameters are specified so that a utility is required to process the requests of several different products in the same execution, then the user must either have the CHANGE-MANAGER role for every product concerned, or else have the TOOL_MANAGER role for the database.

Metadata Utility

Overview

Dimensions CM uses metadata to record file and folder changes in a work area that is associated with a project or stream. In releases prior to Dimensions CM 12.2 these metadata files were located in a folder called `.metadata` corresponding to each folder in the work area. This contained a file for each file and subfolder and the files were in text format.

From Dimensions CM release 12.2, these folders are called `.dm`, and contain:

- One file called `.items.dmdb` for all the files in a folder
- One file called `.dirs.dmdb` for all the sub folders in a folder
- One file called `.workareaconfig.dmdb` in the top-level `.dm` folder corresponding to the work area root folder.

These files are in binary format, so cannot be edited with text-based editors. These changes were made for performance improvements.

The command-line utility `dmmeta` enables you to manage this metadata. The functions it performs are to:

- Convert pre-Dimensions CM 12.2 metadata to the current format.
- Display metadata for a file or folder
- list the metadata for all the files or subfolders in a folder
- Update metadata for a file or folder
- Delete metadata for a file or folder

CAUTION! Micro Focus recommends that you do not change values in the new binary format metadata files unless expressly directed to do so by a Support representative.

Syntax

The format of the command is:

```
dmmeta [ACTION] [PATHNAME] [OPTION]
```

where

ACTION can have the values:

- **convert**: Converts the metadata for the specified path from the pre-Dimensions CM 12.2 format to the current format.
- **get**: Displays the metadata for a specified file or folder
- **set**: Updates the metadata for a specified file or folder
- **del**: Deletes the metadata for a specified file or folder
- **list**: lists all the metadata for a specified folder

PATHNAME Specifies the path of a file or folder whose metadata is to be processed.

If a relative path is specified then:

- If an area root is supplied using the --area option (see below) then the path relative to the area root is used.
- If no area root is specified, the path relative to the current working folder is used.

If there is a space in the pathname you will need to enclose the pathname with quotes.

OPTION can have the values:

- **--type <metadata-type>**

  This specifies the type of metadata to be processed. The possible values are:

  - item
  - dir
  - itemprops
  - dirprops
  - workareaconfig

  The default is item.

- **--area <area root>**

  This specifies the local work area root folder associated with the project or stream. When a relative path is specified for the pathname above, it is interpreted as being relative to this folder, other wise it is interpreted as relative to the current working folder.

- **--file <filepath>**

  This specifies the pathname of a file that contains input or output information.

- **-dry-run**

  This runs a convert in dry-run mode. It verifies the existing old metadata in the specified work area, but does not perform the actual convert.s

- **-verbose**

  This specifies verbose mode when performing the conversion of metadata. It provides more information, but may cause the processing to be slower.

Displaying Help

You can view help for the syntax of *dmmeta* using the following command:

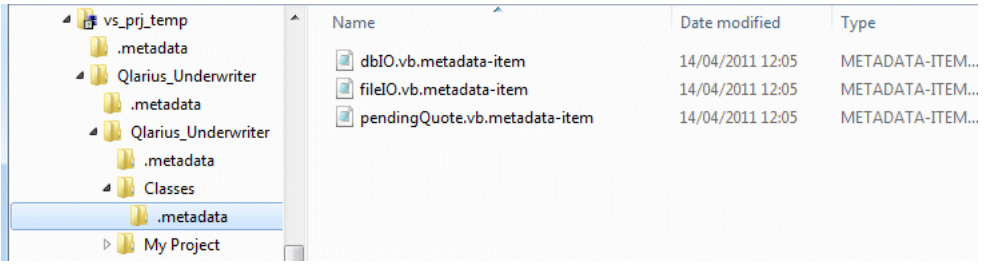
```
dmmeta --help
```

Converting Metadata

The format using the **convert** action is:

```
dmmeta convert <pathname> [-v] [--area <area root>]
   [--file <output filepath>] [--dry-run] [--verbose]
```

For example, if we have the following folder structure for the root folder `vs_prj_temp`:



The command:

```
dmmeta convert qlarius\vs_prj_temp --dry-run
```

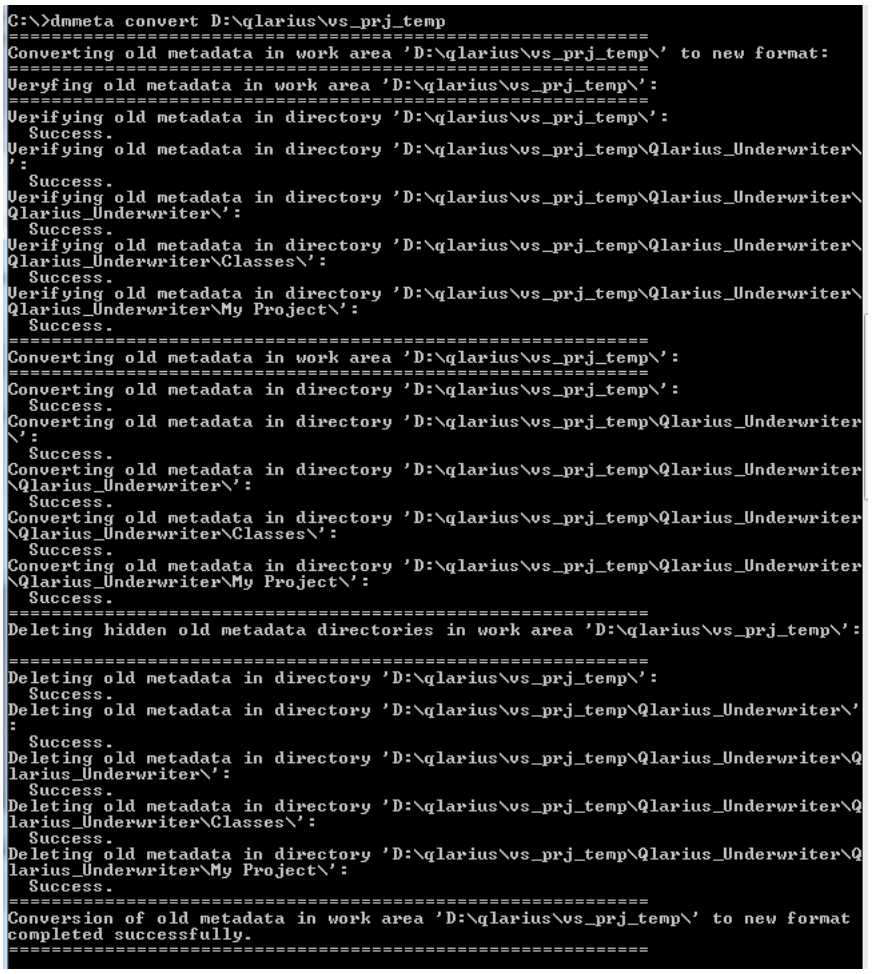
will display a message verifying the old metadata in the work area, but not perform the conversion.

```
C:\>dmmeta convert D:\qlarius\vs_prj_temp --dry-run
============================================================
Converting old metadata in work area 'D:\qlarius\vs_prj_temp\' to n
============================================================
Veryfing old metadata in work area 'D:\qlarius\vs_prj_temp\':
============================================================
Verifying old metadata in directory 'D:\qlarius\vs_prj_temp\':
  Success.
Verifying old metadata in directory 'D:\qlarius\vs_prj_temp\Qlarius
':
  Success.
Verifying old metadata in directory 'D:\qlarius\vs_prj_temp\Qlarius
Qlarius_Underwriter\':
  Success.
Verifying old metadata in directory 'D:\qlarius\vs_prj_temp\Qlarius
Qlarius_Underwriter\Classes\':
  Success.
Verifying old metadata in directory 'D:\qlarius\vs_prj_temp\Qlarius
Qlarius_Underwriter\My Project\':
  Success.
============================================================
Successfully verified old metadata in work area 'D:\qlarius\vs_prj_
============================================================
```

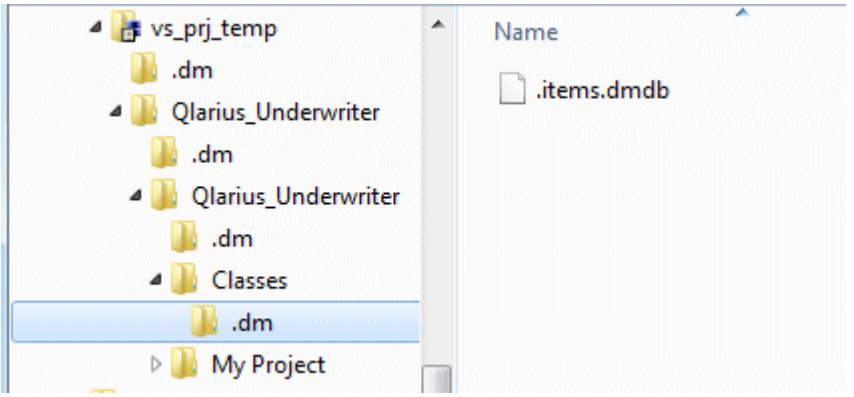
The following command:

```
dmmeta convert qlarius\vs_prj_temp
```

will perform the conversion and display confirmation messages.



and the metadata for all the files and folders beneath this area root folder will be converted to the new format, for example:



Displaying Metadata

The format using the **get** action is:

```
dmmeta get <pathname> [--type <metadata-type>] [--area <area-root>]
   [--file <input-filepath>]
```

For example the following command:

```
dmmeta get "Qlarius Underwriter\qlarius\DBIO.java" --area
    C:\Qlarius\java_typical_1.0
```

will produce the following output:

```
============================================================
C:\qlarius\java_typical_1.0\Qlarius Underwriter\qlarius\DBIO.java
============================================================
version=10.2
item-uid=4215096
item-spec=514C41524955533A344B483443204442494F204A4156412E412D535243
    3B31
file-version=1
checksum=4bd72ca03acc3413048534cd3f4e10fe
fetch-size=375
mtime=1241627044
is-extracted=0
is-headersubst=0
item-status=444556454C4F504552
relpath=Qlarius Underwriter/qlarius/DBIO.java
project=514C41524955533A4A4156415F5459504943414C5F312E30
project-uid=4214672
```

And the following command:

```
dmmeta get "Qlarius Underwriter\qlarius" --area
    C:\Qlarius\java_typical_1.0 --file tmp.txt
```

will produce the following output to the file `C:\tmp.txt`:

```
============================================================
C:\qlarius\java_typical_1.0\Qlarius Underwriter\qlarius\
============================================================
version=10.2
relpath=Qlarius Underwriter/qlarius
project=514C41524955533A4A4156415F5459504943414C5F312E30
project-uid=4214672
```

The following command:

```
dmmeta get C:\qlarius\STREAM_A --type workareaconfig
```

Will produce the following output for the work area root folder

```
============================================================
C:\qlarius\STREAM_A\
============================================================
version=10.2
project=514C41524955533A53545245414D5F41
project-parent=514C41524955533A4D41494E4C494E455F4A415641
project-home=514C41524955533A53545245414D5F41
perms-after-download=writable
perms-after-upload=keep
```

Updating Metadata

The format using the **set** action is:

```
dmmeta set <pathname> [--type <metadata-type>] [--area <area-root>]
   [--file <input-filepath>]
```

CAUTION! Micro Focus recommends that you do not change values in the new binary format metadata files unless expressly directed to do so by a Support representative.

For example to update the file version of the file DBIO.java to 2:

1. First run the command:

   ```
   dmmeta get "Qlarius Underwriter\qlarius\DBIO.java" --area
       C:\Qlarius\java_typical_1.0 --file tmp.txt
   ```

2. Edit the content of the file `tmp.txt` to remove the header section and update the value of `file-version` to 2:

```
version=10.2
item-uid=4215096
item-spec=514C41524955533A344B483443204442494F204A4156412E412D535243
    3B31
file-version=2
checksum=4bd72ca03acc3413048534cd3f4e10fe
fetch-size=375
mtime=1241627044
is-extracted=0
is-headersubst=0
item-status=444556454C4F504552
relpath=Qlarius Underwriter/qlarius/DBIO.java
project=514C41524955533A4A4156415F5459504943414C5F312E30
project-uid=4214672
```

3. Then run the following command:

   ```
   dmmeta set "Qlarius Underwriter\qlarius\DBIO.java" --area
       C:\Qlarius\java_typical_1.0 --file tmp.txt
   ```

This will update the metadata for the file `DBIO.java` with the content of `tmp.txt`.

Deleting Metadata

The format using the **del** action is:

```
dmmeta del <pathname> [--type <metadata-type>] [--area <area-root>]
   [--file <input-filepath>]
```

CAUTION! Micro Focus recommends that you do not change values in the new binary format metadata files unless expressly directed to do so by a Support representative.

For example the following command:

```
dmmeta del "Qlarius Underwriter\qlarius\utilities\DBIO.java" --area
    C:\Qlarius\STREAM_A
```

will produce the following output:

```
C:\>dmmeta del "Qlarius Underwriter\qlarius\utilities\FileIO.java" --area C:\qla
rius\STREAM_A
Deleted metadata of type 'item' for 'C:\qlarius\STREAM_A\Qlarius Underwriter\qla
rius\utilities\FileIO.java'.
```

and delete the metadata for the file FileIO.java from the `.items.dmdb` file in the `.dm` folder corresponding to the folder `Qlarius Underwriter\qlarius\utilities`.

The following command:

```
dmmeta del C:\qlarius\STREAM_A --type workareaconfig
```

Will remove the work area root folder `C:\qlarius\STREAM_A` from its association with Dimensions CM.

Listing Metadata

The format using the **list** action is:

```
dmmeta list <pathname> [--type <metadata-type>]
   [--area <area-root>] [--file <output-filepath>]
```

For example the command:

```
C:\>dmmeta list "Qlarius Underwriter\qlarius" --type dir --area
    C:\qlarius\temp_java_str
```

will produce the following output:

```
List of all metadata of type 'dir' in directory
    'C:\qlarius\temp_java_str\Qlarius Underwriter\qlarius\':
============================================================
C:\qlarius\temp_java_str\Qlarius Underwriter\qlarius\sampledata
============================================================
version=10.2
deliver-uid=4221402
relpath=Qlarius Underwriter/qlarius/sampledata
project=514C41524955533A4D41494E4C494E455F4A4156415F535452
project-uid=4217051
============================================================
C:\qlarius\temp_java_str\Qlarius Underwriter\qlarius\interfaces
============================================================
version=10.2
deliver-uid=4221402
relpath=Qlarius Underwriter/qlarius/interfaces
project=514C41524955533A4D41494E4C494E455F4A4156415F535452
project-uid=4217051
============================================================
C:\qlarius\temp_java_str\Qlarius Underwriter\qlarius\utilities
============================================================
version=10.2
deliver-uid=4221402
relpath=Qlarius Underwriter/qlarius/utilities
project=514C41524955533A4D41494E4C494E455F4A4156415F535452
project-uid=4217051
```

The command:

```
C:\>dmmeta list QLARIUS\VS_TYPICAL_1.0\Qlarius_Underwriter\
      Qlarius_Underwriter\properties
```

will display the metadata for all the files in :

```
   C:\QLARIUS\VS_TYPICAL_1.0\Qlarius_Underwriter\Qlarius_Underwriter
       \properties
```

Actioning Requests by Date or Attribute Value

IMPORTANT! If the *Dimensions CM* "electronic signatures" facility is enabled (authentication required for sensitive changes to a request's or an item's lifecycle state and attributes), attempting automatic actioning to sensitive lifecycle states will always fail.

The dm_auto_action utility is available to all users and supports two different syntaxes:

Syntax (1st form)
```
dm_auto_action <product-id> <request-type> <holding-state>
   <preferred-state> <fallback-state> <date-attribute-name>
```

Example (1st form)
```
dm_auto_action PCMS CR HELD "CCB PENDING" \
   OUTSTANDING HOLD_UNTIL
```

This form of the utility actions each request in <product-id> of type <request-type>, whose current status is <holding-state>, to either <preferred-state> or <fallback-state>, provided that the value of <date-attribute-name> is less than the current system date (i.e. provided the value lies in the past).

<date-attribute-name> is the Variable Name parameter of a user-defined request attribute whose Data-Type is Date for date format (for details, refer to the Process Modeler description (Configuration Object Management | Object Type Definitions | Requests | <request-type> | Attributes) in the *Process Configuration Guide* or the associated online help).

The action will be to <holding-state> provided that this will allow the request to be placed in at least one user's pending list. This means that: <holding-state> must not be a final state, and of the role(s) to handle the lifecycle transition(s) onwards from <holding-state>, at least one must be currently assigned to at least one user.

If this criterion cannot be met, the action will be to <pending-state> (regardless of what the pending-list position may be at this state: no requests that meet the specified criteria are left at <holding-state>).

Syntax (2nd form)
```
dm_auto_action <product-id> <request-type> <request-status>
   <preferred-state> <fallback-state> <attribute-name>=<value>
```

Example (2nd form)
```
dm_auto_action PCMS PR IN_PROGRESS CLOSED \
   COMPLETED WORK_STATUS=COMPLETED
```

In this form of the utility, if any requests of type <change-type> with status <request-status> have the value <value> in the attribute <attribute-name>, then an action check is performed for the preferred state <preferred-state>. If the action-check succeeds, the requests are actioned to the preferred state <preferred-state>, otherwise they are actioned to the fallback state <fallback-state>.

Sending Reminders of Pending Lists

Two utilities, available to all users, are provided to mail users with details on their pending requests:

- **dm_full mail**, which lists all relevant requests pending for each user; and
- **dm_incremental mail**, which lists only relevant requests last actioned (or, optionally, last modified) within the preceding few days.

Syntax
```
dm_full_mail [-v] <product-id> <request-type>
dm_incremental_mail [-u] [-v] <product-id> <request-type> <days>
```

-v specifies a verbose format for each request in the mail message (see below for details). The default is a brief format.

`-u` is valid only for `dm_incremental_mail`, and it specifies that the time of last modification is to be used rather than the time of last action to determine whether a request falls within the designated period.

`<product-id>` is the identity of the product whose requests are to be reported. A null string may be used to specify all products in the database.

`<request-type>` is the request type for the requests to be reported. A null string may be used to specify all request types.

`<days>` is the number of days to be covered by the messages generated by `dm_incremental_mail`. This period ends at midnight immediately before dm_incremental_mail is started.

For example, if three days are specified here, and dm_incremental_mail is started at 2 a.m. on Monday, requests last actioned at 1 a.m. the previous Friday or at 11 p.m. on Sunday will be reported, while requests last actioned at 11 p.m. on Thursday or at 1 a.m. today (Monday) will not be reported.

Separate mail messages are sent to each user listing all requests that meet the given selection criteria and that are pending for the user in a Leader, Primary, or Secondary role. The following are given for each request in both brief and verbose formats:

Request Identity

Current Status

Title (i.e. Attr-no 1): the first 70 characters

The following are given for each request in verbose format only:

The relevant user roles

Related Design Parts

Related Requests and their Current Status

Originator

These utilities are intended to provide regular reports to users on requests needing to be dealt with. They will normally be run as automatic jobs, as described below.

Automatic Job Triggering: Using crontab

If the following entries are placed in the crontab of the change-manager for product PRODX:

```
0 2 * * 2-5 /usr/local/bin/do_incmail PRODX PR 1
0 2 * * 1   /usr/local/bin/do_incmail PRODX PR 3
0 3 1 * *   /usr/local/bin/do_fullmail PRODX PR
```

then `dm_full_mail` will run at 3 a.m. on the first day of every month, while `dm_incremental_mail` will run at 2 a.m. on Mondays to Fridays (to cover just the preceding day, except on Monday when 3 days are included to cover the weekend). All reports are specified to cover requests of type `PR` in the `PRODX` product.

The entries refer to simple scripts that must be set up to invoke the utilities. For example, a C-shell script for `do_incmail` could be:

```
#!/bin/csh
source .login
```

```
dm_incremental_mail $1 $2 $3
exit
```

Similar `crontab` entries can be made to run `dm_auto_action`.

Encryption

In a Dimensions installation the following data is encrypted before being written to disk:

- User names
- Base database names
- Connection strings

  The above are collectively referred to as *User Ids* (`<UserId>`).

- Passwords

The encrypted information is stored in the Dimensions file `registry.dat` located in the `dfs` sub-directory of the installation home directory. The header in `registry.dat` contains the encryption version in use. For additional security, Serena recommends that only the Dimensions System Administrator user account (by default, `dmsys`) can read, and write to, `registry.dat`.

NOTE The User Ids are mapped to upper case before being encrypted. The passwords remain case sensitive.

The `dmpasswd` program enables `<UserId>` entries to be added and deleted, and for passwords to be changed for existing entries.

Constraints

Only the Dimensions System Administrator user account and the installation owner have permissions to run `dmpasswd` and access the `registry.dat` file. No Dimensions roles are required.

This feature is supported on UNIX and Windows. It is not supported on z/OS, since the feature is not required on that platform.

Syntax

```
dmpasswd <UserId> -add [ -pwd <Password> ] | -mod | -del | -help
```

where:

- `<UserId>`

  Specifies the user id entry for a particular user. This argument is mandatory and is always the first argument specified. Examples of `<UserId>` are:

  - Oracle: `<base_db_name>@<connect_string>`
  - Dimensions installation owner: `dmsys`

- `-add`

  Interactively adds a new `<UserID>` entry for a particular user. The program prompts for a password followed by confirmation of that password. The `-pwd <password>` argument can be used to avoid `dmpasswd` prompting for the password.

  Appropriate message texts are logged, for example, `user added`, `user already exists`, and `passwords don't match`. The `-add`, `-mod`, and `-del` arguments are mutually exclusive.

- `-pwd <Password>`

  Optional argument (can only be used with the `-add` option) that specifies the case sensitive password to be used.

- `-mod`

  Interactively modifies the password of an existing `<UserId>`. The program prompts for the old password followed by the new password, and then prompts for confirmation of the new password.

  Appropriate message texts are logged, for example, `password changed`, `user does not exist`, `invalid password`, and `new passwords don't match`. The `-add`, `-mod`, and `-del` arguments are mutually exclusive.

- `-del`

  Deletes a `<UserId>`. The program prompts for the password, and then prompts for confirmation of the deletion.

  Appropriate message texts are logged, for example, `user deleted`, `user does not exist`, and `invalid password`. The `-add`, `-mod`, and `-del` arguments are mutually exclusive.

- `-help`

  Displays program usage.

Examples

Command	Description
`dmpasswd intermediate@dim9 -add`	Registers the `intermediate` base database for an Oracle instance `dim9`.
`dmpasswd dmsys -add -pwd Not-Telling`	Registers the password of the pool owner user account on Windows platforms.
`dmpasswd pcms_sys -mod`	Modifies the password associated with the `pcms_sys` user-id.
`dmpasswd intermediate@dim9 -del`	Deletes the `intermediate@dim9` user-id.
`dmpasswd -help`	Lists a usage summary.
`dmpasswd dim9 -add -pwd dmsys/foobar`	Registers the user-name/password credential required to access a remote IBM UDB database pointed to by the ODBC alias `dim9`.

PRCS–RCS-like Front End to Dimensions

```
prcs subcommand [ option…] [ filename…]
   [ Dimensions option…]
```

NOTE prcs cannot be run from Windows or Dimensions for z/OS.

Examples

1 To create a Dimensions item for file `program.c`, ensure that the desired project is set and that you are in a sub-directory within the scope of the working location, then use *'prcs ci':*

```
example% prcs ci program.c +P fs:fs.aaaa +T src
PROD/program.c <-- program.c
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> cryptic description
>> .
initial revision: 1.1
done
```

2 To check out a copy of `program.c` for editing. Edit it, and then check it back in:

```
example% prcs co -l program.c
PROD/program.c --> program.c
revision 1.1 (locked)
done
example% vi program.c
your editing session
example% prcs ci program.c
PROD/program.c <-- program.c
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file:
>> clarified cryptic diagnostic
>> .
done
```

3 To check out with lock all files under Dimensions in the current directory:

```
example% prcs co -l PROD
```

4 To check in all files currently checked-out to you:

```
example% prcs ci 'prcs rlog -R -L PROD'
```

5 To search for `printf` in all files in the current project directory held in Dimensions.

```
example% prcs grep printf PROD
```

6 To run `wc` on all files in the current project directory held in Dimensions.

```
example% prcs eval wc -- PROD
```

(See page 562 for **Description**.)

Subcommands

The following prcs subcommands invoke programs that provide functionality with similar semantics to the standard RCS commands. See Constraints for non-supported RCS options on page 564.

- co [-l[rev] -u[rev] -p[rev] -q[rev] -r[rev] -sstate -w[login] -k -ddate-time] filename….

  Gets a revision from Dimensions. By default, this is a read-only working copy of the most recent revision on the trunk. The specified revision can be locked so that other users of prcs cannot deposit new revisions with this as a direct predecessor. Dimensions Item header substitutions are performed if enabled unless the revision is being locked or the -k option is used. Refer to co(1).

  -r<rev> Retrieves the latest revision whose number is less than or equal to rev. If rev indicates a branch rather than a revision, the latest revision on that branch is retrieved. If rev is omitted, the latest revision on the trunk will be retrieved.

  -l<rev> Retrieves a revision for editing. The checked out revision will be writable and prcs will not allow other users to lock the specified revision. A repeated lock on the same revision will be allowed. The -l option overrides the -u option.

  -p<rev> Prints the retrieved revision on the standard output.

  -q<rev> Quiet mode: diagnostics are not printed. Check out will be aborted if there is a writable copy of the file in the working directory.

  -u<rev> Same as -r except it unlocks the retrieved revision if it was locked by the caller. If rev is omitted, -u retrieves the revision locked by the caller if there was one, indicates errors if more than one is locked or else it retrieves the tip trunk revision.

  -f<rev> Forces overwriting of the working file. This is useful with -q.

  -ddate Retrieves the latest revision on the selected branch whose check in time is less than or equal to present date/time. Date may take the form: mm/dd/yyyy [hhmm] or dd-mon-yyyy [hh:mm]. Omitted units that default to zero values e.g. 08/23/1996 are equivalent to **any** setting of the locale that affects *date/time* formats, will be honored.

  -k Does not perform Dimensions item header substitution. This is the same as the /NOEXPAND option to the Dimensions FI command. This is provided for compatibility with the psccs command.

  -sstate Retrieves the latest revision on the selected branch whose state is set to state.

  -w<login> Retrieves the latest revision on the selected branch which was checked in by the user with login name login. If the user name is omitted, the caller's login is assumed.

- ci [-l[rev] -u[rev] -q[rev] -r[rev] -sstate -i -j -mmsg -tdesc] filename…

  Checks in new revisions. The effective user ID must be the same as the ID of the person who has the predecessor revision locked or must have a role to update the item. Refer to ci(1).

  Unless the -f option is given, ci will check if the working file differs from the preceding one. If there are no differences, ci will not perform the check in. Check-summing must be turned on for the item type if this mode of operation is to be

enabled. If check-summing is off or the files differ, a new revision will be created. For each revision deposited a log message is requested. If multiple files are to be operated on, `ci` will ask if the log message is to be reused. If a log message is given with `-m`, this is used for all check ins.

If there is no equivalent Dimensions item, a new item is created with default revision `1.1`. The `+P`, `+T` and optionally `+F`, together with any desired `+C` or `+A` options must be provided. The following steps will be performed:

- Creates a Dimensions item with the following parameters:

  Item-id derived from the file name by replacing all "**_ .$ # ~ ; : -**" characters with a space character and truncating to 25 characters. If this is not unique, the user must use Dimensions to create the item.

  Item type given by the `+T` Dimensions Option.

  Revision defaults to `1.1` unless a revision is given using the `-r` option. If just the release component is given, the initial revision will then be `<release>.1`.

  Owned by the design part given by the `+P` Dimensions specific option.

  The project file name will consist of a project directory that matches the difference between the current working directory and working location and the file name specified to the create command.

  The `+F` Dimensions specific option must be used to specify the format if the file name has no extension.

- Performs a `prcs co` or `prcs co -l` on the item to retrieve a read-only or editable copy of the initial revision if the `-u` or `-l` options were given.

The revision given to any new Dimensions revision will be determined in the following way.

- If `rev` is a full revision number, it must be higher than the latest one on the branch to which the revision is being appended.
- If `rev` is a branch number, the revision is appended to an existing branch by incrementing the sequence component.
- If the branch given is non-existent and the branch point exists, then a new branch is created with the initial revision being `Rev.1`.
- If `rev` is omitted, the new revision will be derived from existing locks on the file.
- If a tip revision was locked, then a new revision will be appended to the relevant trunk or branch. The new revision number will be given by incrementing the branch or sequence component.
- If a non-tip revision was locked, then a new distinct branch will be started at that point by incrementing the branch component and starting with a sequence component of `1`. Only one revision must be locked by the caller if the new revision is to be omitted from the command.

`-r<rev>` Checks in revision `rev`. The working file is deleted.

`-r` With no revision specified to the `-r` option removes the working file and unlocks the predecessor revision. This overrides any `-l` or `-u` options given to the command.

`-l<rev>` Works like `-r` except a `co -l` is performed. Any requests or attributes required for the check out must be specified.

`-u<rev>` Works like `-l` except a `co` is performed.

`-f<rev>` Forces deposition of the new revision even if it does not differ from the predecessor.

`-q<rev>` Diagnostics are not printed. Revisions identical to the predecessor are not deposited unless `-f` is specified or item check-summing is not enabled.

`-mmsg` Uses the string `msg` for all deposited revisions.

`-I` Initializes: error indicated if file already exists in Dimensions.

`-j` Just checks in: error indicated if file does not exist in Dimensions.

`-sstate` Actions the deposited revision to state. If state is omitted, the next normal lifecycle state is assumed.

`-tstring` Uses string as description when creating a new item.

`-t-string` Uses string as description when creating a new item. The '`-`' character is stripped from the string.

- `rcs [-l<rev>] [-u<rev>] [-o<rev>] [-i] [-sstate[:rev] filename….`

  The `rcs` command performs administrative functions.

  `-l<rev>` Locks the specified revision. A check out is performed to a temporary file that is then discarded.

  `-u<rev>` Unlocks the specified revision. Only the locker of a revision can unlock it.

  `-o<rev>` Deletes the specified revision.

  `-I` Creates a new empty item in Dimensions. The revision given is `1.1`. The `+P`, `+T`, `+F` and any `+C` or `+A` Dimensions specific options must be supplied. This differs from RCS usage which does not create an initial revision for the `rcs -i` command until a `co` is performed. See `ci` (on page 557) for a description of the operations performed on initialization.

  `-s<state><:rev>` Action the revision specified by `rev` to state `state`. If state is omitted, assume the next normal lifecycle state.

- `rlog [-L] [-R] [-h] [-t] [-N] [-b] [-ddates] [-l<lockers>] [-r<revisions>] [-sstates] [-w<logins>] filename…`

  Prints rcs-style information about Dimensions item. Refer to `rlog(1)`. The intersection of the revisions selected with `-d -l -s` and `-w` intersected with the union of revisions selected by `-r` and `-b` are printed.

  `-L` Ignores files that have no locks set.

  `-R` Prints the name of the file prefixed by `Dimensions/`.

  `-h` Prints **only** the file name, head revision, locks and symbolic names.

  `-t` As `-h` plus description.

  `-N` Does not print symbolic names.

  `-b` Only prints information for the highest branch on the trunk.

-ddates
: Prints information about revisions with a check in within the specified range of date/times. The *date/time* values are as described for co (on page 557). The '**<**', '**>**' and '**=**' characters in conjunction with one or two dates specify the range.

- d1<d2 or d2>d1 between d1 and d2 (exclusive)
- <d or d> revisions earlier than d
- d< or >d revisions later than d
- If '=' follows the '<' or '>' characters the comparisons are inclusive.
- d single revision dated d or earlier

-l<lockers>
: Prints information about locked revisions only. Lockers is a comma-separated list of user names that restrict the selection to only those items locked by the specified users.

-r<revisions>
: This specifies a range of revisions to report on.

- :rev specifies revisions from the beginning of the branch to rev.
- rev: specifies revisions from rev to the end of the specified branch.

A branch name means all revisions on that branch. A range of branches means all revisions on the specified branches and a bare -r means the tip of the trunk. Only one range can be specified. A separator of ' **-** ' is also supported.

-sstates
: Only print information for revisions that have a status matching any one of the comma-separated list states.

-wlogins
: Only report on revision checked in by the comma-separated list of users logins.

- rcsdiff [diffoptions] [-k] [-q] [-rrev1] [-rrev2] filename…

  Compares two revisions corresponding to the specified revisions using diff. Refer to rcsdiff(1). All options except -k are passed to diff**,** see diff(1)**.** The -k option turns off item header substitution. If -q is specified, diagnostics are not printed.

  - If both rev1 and rev2 are omitted, rcsdiff compares the latest revision on the trunk with the contents of the corresponding working file. That is, the file with the same leaf file name in the current working directory.
  - If rev1 is given but rev2 is omitted, rcsdiff compares revision rev1 with the contents of the corresponding working file.
  - If both rev1 and rev2 are given, rcsdiff compares revisions rev1 and rev2 of the corresponding Dimensions item.
  - rev1 can be given numerically or symbolically, rev2 must be given explicitly.

- help

  Displays a list of the supported commands.

- ls [-l] [-R]

  Displays the list of files that are in the matching directory in the current project. The -l and -R options cannot be combined.

  l
  : Appends latest revision to file name.

  R
  : Performs List recursively.

- grep [-r<rev>] [grep_options] [grep_string] filename…

Performs the grep command on the tip trunk revision of the specified files in the current project. The -r option can be used to specify a particular revision or named branch. The grep options are as documented in grep (1). The grep string should be enclosed in single quotation characters (') if the search string contains spaces or shell meta-characters.

- eval [-r<rev>] [command] [command_options] -- filename…

  Performs the specified command on the tip trunk revision of the specified files in the current project. The -r option can be used specify a revision or named branch. The options are as documented for the specified **Any** parameters that contain spaces or shell meta-characters which should be enclosed in single-quotation characters ('). The -- separator is required after the last command parameter.

Dimensions Options

The prcs command takes a number of subcommand specific options and, additionally, Dimensions specific options may be specified. Dimensions-specific options for prcs must appear after the subcommand and optional fIlename arguments. Options for a given subcommand must appear after the subcommand argument. These options are specific to each subcommand, and are described with the subcommands themselves (see Subcommands on page 557).

- +V

  Displays the Dimensions commands and messages generated by the prcs operation. The project selected by prcs will also be displayed.

- +W <project>

  Defines the Dimensions project to be used for the prcs command. This option is required if prcs is unable to determine the project from the current directory. The working location is set to the current working directory unless a +D option is also specified.

- +D <working location>

  Defines the working location to be used for the project given in the +W Dimensions Option. This cannot be specified without the +W option.

- +C <Request List>

  This option allows a list of requests to be supplied to the Dimensions command invoked by prcs where this is appropriate. This applies to checking-out with lock creation (rcs -i, ci) and check in commands that force an update. The format of the option is:

  ```
  +C 'CHANGE_DOC_ID_1[, CHANGE_DOC_ID_2,…]'
  ```

- +A <Attribute List>

  This option allows a list of Dimensions item attribute values to be supplied to the Dimensions command invoked by prcs where this is appropriate. This applies to checking-out with lock creation (rcs -i, ci) and check in commands that force an update. The format of the option is:

  ```
  +A 'ATTRIBUTE_VARIABLE1=value[, ATTRIBUTE_VARIABLE2=value,…]@
  ```

- +N <New Revision>

This option allows a specific new revision to be specified instead of that determined by prcs.

- +P <Design Part>

  This option allows a Dimensions design part to be specified for the prcs commands that create new Dimensions items. The format of the option is:

  ```
  +P 'PRODUCT_ID:PART_ID.VARIANT[;PCS]'
  ```

- +T <Item Type>

  This option allows a Dimensions item type to be specified for the prcs commands that create new Dimensions items.

- +F <Item Format>

  This option allows a Dimensions item format to be specified for the prcs create and prcs enter commands. This is required only if there is no extension on the file being entered into Dimensions. Otherwise the format is determined from the extension.

  All values to be passed to Dimensions incorporating embedded spaces must be protected by double-quotation characters (""). The whole parameter should be enclosed in single- quotation characters (') to protect it from interpretation by the shell.

Description

The prcs command is a RCS-like front end to the version control commands of Dimensions.

prcs applies the indicated *subcommand* to the Dimensions 'item' associated with each of the specified files.

The mapping between the file name(s) specified the prcs command and the associated Dimensions 'item' is derived from the file name by searching for a matching project file name in the current project.

The prcs command expects these 'items' to reside in a project directory that when appended to the current project root equates to the current working directory. Thus, when you invoke prcs from directory "/usr/work" which is marked as being the working location for project WS1 with a file name argument, the subcommand will be applied to a Dimensions item in the root directory of project WS1 with project file name equivalent to that specified to the command. If the command is issued in a subdirectory of "/usr/work", then the file name is appended to the offset from the working location to the current directory and this name is used to find the 'item'. If the file name is given as PCMS or PCMS/<wildcard>**,** then the command is applied to all Dimensions items which are in the same relative sub-directory of the project and which match the wildcard. The wildcard is in standard Dimensions format. Thus executing the prcs command in the working location prcs co program.c would apply the co (check out) subcommand to an 'item' with project file name program.c**.** However, executing the same prcs command in the project directory "src" would apply the co subcommand to an item with project file name src/program.c**,** while the command prcs co PCMS executed in a directory "/usr/work/src" in project WS1 with root directory "/usr/work/" would apply the co subcommand to every item in Dimensions with the project directory "src".

prcs uses the same mechanism as Dimensions Make to determine the correct project for operation. If it is unable to determine a project, then a diagnostic message is issued and

the +W and +D Dimensions-specific options must be used (see Dimensions Options on page 561).

Revisions

prcs accepts revision specifications using the following syntax:

```
-option<revision_specification>
```

where option depending on the prcs subcommand is one of:

```
l, p, q, r, f, u
```

There should be no space between the -option and the revision specification. The revision specification can contain Dimensions named branches, for example:

```
-rmaint#1
```

prcs will also accept a symbolic name for a revision. prcs will use the revision of the relevant Dimensions item contained in the Dimensions baseline identified by the symbolic name. The Dimensions product-id can be omitted, in which case, the product-id is taken from the project that prcs is operating in.

```
prcs co -rfs:source_1.0 foo.c
```

Will check out the revision of foo.c contained in baseline fs:source_1.0.

prcs by default uses the RCS style of revisioning. This is composed of the following components.

```
Release.Level[.Branch.Sequence.[ Branch.Sequence…]]
```

By convention RCS revisions start at 1.1 and the main trunk consists of revisions with just Release and Level components. The default action of prcs is to operate on the tip trunk revision. This is the revision with the highest Release and Level numbers. The default next revision for the tip is to increment the level component. Branches are identified by adding branch and sequence components to the trunk revision specifier. A branch is initiated by checking in a new revision following a check out with lock on a non-tip trunk revision or by explicit use of a branch revision specifier. The first branch from Revision 1.2 would be 1.2.1.1. Subsequent branches from Revision 1.2 would have the Branch component incremented. A branch from a branch is identified by appending 1.1 to the to the current branch revision.

Dimensions revisions do not follow the same convention**.** prcs **treats Dimensions revisions with a single numerical field as belonging to Release '0'**. A branch revision e.g. 1.1 will be interpreted by prcs as a trunk revision in Release 1. Dimensions style revisions can be accessed by prcs commands by using default action or explicit specification. If default operation is used, the latest revision with a style revision or a RCS style trunk revision would be used. Specifying a release of '0' to a prcs command restricts the search to Dimensions style single numeric field revisions. Dimensions style revisions can be accessed explicitly by prefixing their revision with the non-existent Release 0. A branch created from a style revision will have the new revision 0.revision.1.1 so that it will appear as a branch in the RCS style of revisioning.

prcs prints the revisions that will be operated on by default. These values should be checked to ensure that the desired revision is being accessed.

Dimensions Named Branches

prcs supports Dimensions named branches in the following way. A named branch is specified by using the branch name in the prcs revision specification. Specifying a branch name without the branch revision will be interpreted as the highest revision on the branch. Specifying a branch name and a revision will access that revision. Editing a non-tip named branch revision causes the new revision to be an unnamed branch from the named branch, for example:

editing foo.c;fix23#1.1 gives new revision: foo.c;fix23#1.1.1.1

prcs will use the default project branch if the branch name is omitted e.g. prcs co -r# will checkout the tip trunk revision on the default branch, and prcs co -r#2.3.1.4 will check out the specified revision on the default branch.

Environment

All environment variables required for correct Dimensions functioning must be present.

prcs Dimensions Files

$DM_PROG/prcs	prcs program

Constraints

This command is available to all users who have the roles to perform the associated Dimensions operations, but not from Windows or z/OS systems.

Currently only emulations of the co, ci, rcs, rlog, and rcsdiff RCS commands are supported. The additional non-rcs commands ls, grep and eval are supported to provide additional functionality.

The following RCS commands are not supported.

- rcsmerge
- ident

The following command options for prcs subcommands are not supported:

```
co
       -I -M -j -V -x –z
ci
       -k -d -M -n/N -I -T -w -V -x -z

rcs
       -a/A -e -b -c -k -L -U -I -m -M -n/N -T -V

rlog
       -V -x -z
```

Refer to the UNIX manual pages for the equivalent RCS commands for further information.

NOTE

There is a conflict between the Forced Revision Generation flag on Dimensions projects and the new revisions generated by `prcs`. Do not set this flag if the project is going to be used for `prcs` operations.

The use of automatic Item-Id generation is detected and a null Item-Id will be used for the creation of items using the `ci` and `rcs -i` subcommands. The implicit get or check out when the `-u` or `-l` option is supplied to `ci` will not be performed. An explicit `co` or `co -l` should be performed.

The calling syntax for `prcs` is different from the common usage of RCS in which the individual subcommands are called directly from the shell prompt. Users of shells with an alias capability can alias the `prcs` subcommands to their common form. For example, the following syntax can be used in the `csh`:

```
% alias co prcs co.
```

PSCCS–SCCS-like Front End to Dimensions

```
psccs subcommand [ option…] [ filename…]
   [ Dimensions option…]
```

NOTE psccs cannot be run from Windows or Dimensions for z/OS.

Examples

1 To create a Dimensions item for file `program.c`, ensure that the desired project is set and that you are in a subdirectory within the scope of the working location. Then use `psccs create`:

```
example% psccs create program.c +P PAYROLL:PAYROLL.A +T src
program.c:
1.1
```

2 To check out a copy of `program.c` for editing, edit it, and then check it back in:

```
example% psccs edit program.c
1.1
new delta 1.2
25 lines
example% vi program.c
your editing session
example% psccs delget program.c
comments? clarified cryptic diagnostic 1.2
1.2
```

3 To check out all files under Dimensions in the current directory:

```
example% psccs edit PROD
```

4 To check in all files currently checked out to you:

```
example% psccs delta 'psccs tell -u'
```

5 To search for `printf` in all files in the current project directory held in Dimensions.

```
example% psccs grep printf PROD
```

6 To run `wc` on all files in the current project directory held in Dimensions.

```
example% psccs eval wc -- PROD
```

(See page 571 for **Description**.)

Subcommands

The following `psccs` subcommands invoke programs that provide functionality with similar semantics to the standard UNIX SCCS commands. Many of these subcommands accept additional arguments that are documented in the reference page for the equivalent SCCS utility. (See **Constraints** on page 572 for non-supported SCCS options.)

- `check [-b] [-u[<user-name>]]`

  Checks for files currently being edited. Like `info` and `tell`, but returns an exit code, in addition to producing a listing of files. `check` returns a non-zero exit status if anything is being edited.

  `-b` Ignores branches.

  `-u[<user-name>]` Only checks files being edited by you. When `<user-name>` is specified, only check files being edited by that user.

- `create [-r<release>] filename…`

  Creates (initializes) Dimensions item for specified file(s). `create` performs the following steps:

  - Create a Dimensions item with the following parameters.
    - Item-Id derived from the file name by replacing all "_ . $ # ~" characters with a space character and truncating to 25 characters.
    - Item type given by the `+T` Dimensions specific option.
    - Revision defaults to `1.1` unless a release is given using the `-r` option. The initial revision will be `<release>.1`.
    - Owned by the design part given by the `+P` Dimensions specific option.
    - This project file name consists of a project directory that matches the difference between the current working directory and working location and the file name specified to the create command.
    - The `+F` Dimensions specific option must be used to specify the format if the file name has no extension.
  - Performs a `psccs get` on the specified file to retrieve a read-only copy of the initial revision.

- `deledit [-s] [-y[comment]]filename…`

Equivalent to a `psccs delta` and then a `psccs edit`. `deledit` checks in a new revision, and checks the file back out again.

`-s` Silent. Does not report revision numbers or statistics.

`-y[comment]` Supplies a comment for the revision commentary. If `-y` is omitted, the command will prompt for a comment. A NULL *comment* results in an empty comment for the checked in revision.

- `delget [-s] [-y[comment]]filename…`

  Performs a `psccs delta` and then a `psccs get` to check in the new revision and retrieve a read-only copy of the resulting new revision. See the `deleted` subcommand for a description of `-s` and `-y`. `psccs` performs a `delta` on all the files specified in the argument list, and then a `get` on all the files. If an error occurs during the `delta`, the `get` is not performed.

- `delta [-s] [-n] [-y[comment]]filename…`

  Checks-in pending changes. The effective user ID must be the same as the ID of the person who has the file checked out. Refer to `sccs-delta(1)`. See the `deledit` subcommand for a description of `-s` and `-y`. The `-n` option keeps the working file.

- `diffs [-C] [-cdate-time] [-rrevision] filename…`

  Compares (in `diff(1)` format) the working copy of a file that is checked out for editing, with a revision from the Dimensions history. Use the most recent checked-in revision by default. The `diffs` subcommand does not accept any `diff`, options with the exception of the `-c` option to `diff` which must be specified as `-C`.

  `-C` Passes the `-c` option to `diff`.

  `-cdate-time` Uses the most recent version checked in before the indicated date and time for comparison. `date-time` takes the form: `mm/dd/yyyy [hh:mm] dd-mon-yyyy [hh:mm]` Omitted units default to zero values; that is `-c08/23/1996` is equivalent to `-c08/23/1996 00:00`.

  `-r<revision>` Uses the revision specified for comparison.

- `edit`

  Retrieves a revision of the file for editing. `psccs edit` checks out a version of the file that is writable by the user and marks the new revision as *checked out* in Dimensions, so that no one else can check that revision in or out. Item header substitutions are not performed. `edit` accepts the same options as `get`, below.

- `enter`

  Similar to `create`, but omits the final `psccs get`. This may be used if an `psccs edit` is to be performed immediately after creation.

- `get [-e-k-p-s] [-cdate-time] [-rrevision] filename…`

  Gets a revision from Dimensions. By default, this is a read-only working copy of the most recent revision on the trunk; Dimensions item header substitutions are performed if enabled. Refer to `sccs-get(1)`.

  `-e` Retrieves a revision for editing. Same as `psccs edit`.

  `-k` Does not perform Dimensions **item header substitution**. This is the same as the `/NOEXPAND` option to the `FI` command.

  `-p` Produces the retrieved revision on the standard output. Reports that would normally go to the standard output (revisions, statistics) are directed to the standard error.

  `-s` Silent. Does not report revision numbers or statistics.

  `-cdate-time` Use the most recent version checked in before the indicated date and time for comparison. `date-time` takes the form: `mm/dd/yyyy [hh:mm] dd-mon-yyyy [hh:mm]` Omitted units default to zero values; that is `-c08/23/1996` is equivalent to `-c08/23/1996 00:00`. Any setting of the locale that affects date/time formats will be honored.

  `-r<revision>` Retrieves the specified revision.

- `help`

  Displays a list of the supported commands.

- `info [-b] [-u[<user-name>]]`

  Displays a list of files being edited, including the revision checked out, the revision to be checked in, the name of the user who holds the lock, and the date and time the file was checked out.

`-b` Ignores branches.

`-u[<user-name>]` Only lists files checked out by you. When `<user-name>` is specified, only list files checked out by that user.

- `print`

  Prints the entire history of each named file. Equivalent to an `psccs prs -e`

- `prs [-el] [-cdate-time] [-rrevision] filename…`

  Peruses (displays) the history of a Dimensions item. Refer to `sccs-prs(1)`.

  `-e` Displays history information for all revisions earlier than the one specified with `-r` (or all revisions if none is specified).

  `-l` Displays information for all revisions later than, and including, that specified by `-c` or `-r`.

  `-cdate-time` Specifies the latest revision checked in before the indicated date and time. The *date-time* argument takes the form:
  `mm/dd/yyyy [hh:mm] dd-mon-yyyy [hh:mm]`

  `-rrevision` Specifies a given revision

- `sccsdiff -rold-revision -rnew-revision -C filename…`

  Compares two revisions corresponding to those specified using `diff`. Refer to `sccs-sccsdiff(1)`. Only the `-c` option to `diff` is supported which must be specified as `-C`.

- `tell [-b] [-u[<user-name>]]`

  Displays the list of files that are currently checked out, one file name per line.

  `-b` Ignores branches.

  `-u[<user-name>]` Only lists files checked-out to you. When `<user-name>` is specified, only list files check out to that user.

- `unedit filename…`

  "Undoes" all pending `edits` or `get -e`, and checks in the working copy to its previous condition. `unedit` backs out all pending changes made since the file was checked out.

- `unget [-n]`

  Same as `unedit`. Refer to `sccs-unget(1)`. The `-n` option keeps the working file.

- `ls [-l] [-R]`

  Displays the list of files that are in the matching directory in the current project. The `-l` and `-R` options cannot be combined.

  `-l` Appends revision to file name.

  `-R` Performs List recursively.

- `grep [grep_options] [grep_string] filename…`

  Performs the `grep` command on the tip trunk revision of the specified files in the current project. The grep options are as documented in `grep (1)`. The grep string

should be enclosed in single quotation characters (') if the search string contains spaces or shell meta-characters.

- `eval [command] [command_options] -- filename…`

  Performs the specified command on the tip trunk revision of the specified files in the current project. The options are as documented for the specified command. Any parameters that contain spaces or shell meta-characters should be enclosed in single quotation characters ('). The -- separator is required after the last command parameter.

Dimensions Options

As well as the `subcommand` specific options, the `psccs` command can take a number of Dimensions specific options. Such-specific options for `psccs` must appear after the `subcommand` and optional file name arguments.

(Options for a given subcommand must appear after the `subcommand` argument. These options are specific to each subcommand, and are described along with the subcommands themselves, see `Subcommands` on page 566.)

- `+V`

  Displays the Dimensions commands and messages generated by the `psccs` operation. The project selected by `psccs` will also be displayed.

- `+W <project>`

  Defines the Dimensions project to be used for the `psccs` command. This option is required if `psccs` is unable to determine the project from the current directory.

- `+D <working location>`

  Defines the working location to be used for the project given in the `+W` Dimensions Option.

- `+C <Request List>`

  This option allows a list of requests to be supplied to the Dimensions command invoked by `psccs` where this is appropriate. This applies to edit and creation commands only. The format of the option is:
  `+C ''CHANGE_DOC_ID_1[, CHANGE_DOC_ID_2,…]'`

- `+A <Attribute List>`

  This option allows a list of Dimensions item attribute values to be supplied to the Dimensions command invoked by `psccs` where this is appropriate. This applies to edit and creation commands only. The format of the option is:
  `+A 'ATTRIBUTE_VARIABLE1=value`
  `[,ATTRIBUTE_VARIABLE2=value,…]'`

- `+N <New Revision>`

  This option allows a specific new revision to be specified instead of that determined by `psccs`. This applies to the `edit` and `get -e` commands only.

- `+P <Design Part>`

  This option allows a Dimensions design part to be specified for the `psccs create` and `psccs enter` commands. The format of the option is:
  `+P 'PRODUCT_ID:PART_ID.VARIANT[;PCS]'`

- +T <Item Type>

  This option allows a Dimensions item type to be specified for the psccs create and psccs enter commands.

- +F <Item Format>

  This option allows a Dimensions item format to be specified for the psccs create and psccs enter commands. This is only required if there is no extension on the file being entered into Dimensions. Otherwise the format is determined from the extension.

  All values to be passed to Dimensions incorporating embedded spaces must be protected by double-quotation characters ("). The whole parameter should be enclosed in single-quotation characters (') to protect it from interpretation by the shell.

Description

The psccs command is a SCCS-like front end to the version control commands of Dimensions.

psccs applies the indicated subcommand to the Dimensions item associated with each of the specified files.

The mapping between the file name(s) specified to the psccs command and the associated Dimensions item is derived from the file name by searching for a matching project file name in the current project.

The psccs command expects these 'items' to reside in a project directory that when appended to the current project root equates to the current working directory. Thus, when you invoke psccs from directory "/usr/work" which is marked as being the working location for project WS1 with a file name argument, the subcommand will be applied to an item in the root directory of project WS1 with project file name equivalent to that specified to the command. If the command is issued in a sub-directory of "/usr/work", then the file name is appended to the offset from the working location to the current directory and this name is used to find the Dimensions item.

If the file name is given as PCMS or PCMS/<wildcard>, then the command is applied to all Dimensions items which are in the same relative sub-directory of the project and which match the wildcard. The wildcard is in standard Dimensions format. Thus, executing the psccs command in the working location psccs get program.c would apply the get subcommand to an item with project file name program.c

However, executing the same psccs command in the project directory "src" would apply the get subcommand to an 'item' with project file name src/program.c while the command psccs get PCMS executed in a directory "/usr/work/src" in project WS1 with root directory "/usr/work/" would apply the get command to every item in Dimensions with the project directory "src".

psccs uses the same mechanism as Dimensions Make to determine the correct project for operation. If it is unable to determine a project, then a diagnostic message is issued and the +W and +D Dimensions specific options (see Dimensions Options on page 570) must be used.

Revisions

psccs uses the SCCS style of revisioning. This is composed of the following components.

```
Release.Level.Branch.Sequence
```

By convention SCCS revisions start at 1.1 and the main trunk consists of revisions with just Release and Level components. The default action of psccs is to use the tip trunk revision. This is the revision with the highest Release and Level numbers. A branch is initiated by editing a non-tip trunk revision. The first branch from revision 1.2 would be 1.2.1.1. Subsequent branches from revision 1.2 would have the branch component incremented. A branch from a branch is created by incrementing the branch component and setting the sequence component to 1. If this branch was already used the branch component is incremented until a non-used branch number is found.

Dimensions revisions do not follow the same convention. **psccs treats Dimensions revisions with a single numerical field as belonging to Release '0'**. A branch revision e.g. 1.1 will be interpreted by psccs as a trunk revision in Release '1'. Dimensions style revisions can be accessed by psccs commands by using default action or explicit specification. If default operation is used then the latest revision with a style revision or a SCCS style trunk revision would be used. Specifying a release of 0 to a psccs command restricts the search to Dimensions style single numeric field revisions. Dimensions style revisions can be accessed explicitly by prefixing their revision with the non-existent Release '0'. A branch created from a style revision will have the new revision 0.revision.1.1 so that it will appear as a branch in the SCCS style of revisioning.

psccs prints the revisions that will be operated on by default. These values should be checked to ensure that the desired revision is being accessed.

Dimensions Named Branches

psccs supports Dimensions named branches in the following way. A named branch is specified by using the branch name in the psccs -r option. Specifying a branch name without the branch revision will be interpreted as the highest revision on the branch. Specifying a branch name and a revision will access that revision. Editing a *non-tip* named branch revision causes the new revision to be an unnamed branch from the named branch e.g. editing foo.c;fix23#1.1 gives new revision: foo.c;fix23#1.1.1.1

psccs Dimensions Files

$DM_PROG/psccs	psccs program

Constraints

Currently only emulations of the check, create, deledit, delget, delta, diffs, edit, enter, get, help, info, prs, sccsdiff, tell, unedit and unget SCCS commands are supported. The additional non-sccs commands eval, grep, and ls are supported to provide additional functionality.

The following SCCS commands are not supported:

- admin
- cdc
- clean
- comb
- fix
- print
- rmdel
- val
- what

The following command options for psccs subcommands are not supported:

- admin
  -h -z -a -d -e -f -l -m
- get
  -m -n -l[p] -a -i -x
- delta
  -p -g -m

Refer to the UNIX manual pages for the equivalent SCCS commands for further information.

Chapter 4

The Developer Command-Line Interface

Introduction	576
Working with DM – Some Typical Development Scenarios	578
What are the Commands I Can Perform?	586
Alphabetic List of Commands	587
Command List	589
Configuring the Developer Command-Line	631

Introduction

This chapter describes the Serena® Dimensions® CM Developer Command-Line, a simplified command interface designed for developers working exclusively with streams. The purpose is to:

- Provide a utility that is easy to learn and use
- Enable users to work with a simpler syntax than that of the dmcli client
- Only provide those commands that are required for working with streams

What Can I Do with the Developer Command-Line?

The Developer Command-Line provides a set of functions designed to manage streams in a command-line format. Streams are a category of project that is better suited to collaborative development and for using agile development techniques. For an overview of streams, see "About Streams" in the User's Guide.

This utility allows you to:

- Perform simple administration of streams
- Switch between streams and work area locations
- Update your work area with the latest files in a stream and automerge any conflicts
- Commit changes from your work area to a stream
- View information about streams, baselines, and file history

How Do I Use the Developer Command-Line Interface?

To use the Developer Command-Line interface, you need to invoke an executable called *DM*. This command interfaces with a daemon, a process that manages its connection to the database. The DM executable will connect to the Dimensions CM repository that you specify in the standard way that all Dimensions CM clients use. Dimensions CM makes use of a connection cache, which means that once a connection has been made to a specified repository, that connection will be automatically reused unless:

- Different connection details are specified
- The cache is cleared via the logout command
- The cache times out and is automatically deleted

You need to understand how the commands are structured in order to use them. Each DM command consists of the following components:

- A command name, which may have one or more aliases, or alternative names
- Parameters that may be mandatory or optional
- A set of command-specific options
- A set of global options, specifying the connection details, that can be applied to any command. See "How do I Connect to the Database?" on page 577.

The format for any given command is generally:

```
dm command parameter_1 parameter_2 ...
   [--option-1
   --option-22
   ... ]
```

How do I Invoke the Developer Command-Line Interface?

To invoke the Developer Command-Line interface, invoke the `DM` command. This runs the utility and connects to the Dimensions repository and server that you have specified. There are a number of ways that you can specify your desired repository. For more details, see "How do I Connect to the Database?" on page 577.

How can I Display Help Information?

DM comes with a help system that you can access via the *help* command. To access general help on DM, you can specify

```
dm --help
```

This will show the list of commands DM provides and a summary of what these commands do. To access specific help on a command, you can do the following

```
dm help <command>
```

For example:

```
$ dm help sw
```

will display the help text describing what the command does and the paramaters available.

How do I Connect to the Database?

There are a number of global options that can be used with any DM command that allow you to specify your repository connection details. These are:

```
--user <user_name>
--password <password>
--database <database>
--server <server_name:port>
--card
```

Where:

- `<user_name>`

  Specifies a Dimensions CM user name.

  If omitted, the DM command will use the name of the OS user that you are currently logged in as.

- `<password>`

  Specifies the password for the user.

If omitted, the DM command will use the last password that you supplied for the user specified by `<user>`, if cached by the connection daemon.

- `<database>`

  Specifies the database name and connection string for the database to which you are connecting.

  If omitted, the DM command will connect to the last database that you connected to as the user specified by `<user>`.

- `<server_name[:port]>`

  Specifies the name of the server and optionally the port number to connect to. For example, `devserver` or `devserver:699`

  - SDP protocol: `<server name[<:port>]`
  - HTTP protocol: `http://<server name[<:port>]`
  - HTTPS protocol: `https://<server name[<:port>]`

  If omitted, the DM command will connect to the last server and port number that you connected to as the user specified by `<user>`.

- `--card`

  Specify this parameter to use a Smart Card to login (only applicable if configured).

  You will be prompted for your PIN if you have not previously supplied it. You will then be presented with a dialog box asking you to choose a certificate. These credentials will be used instead of your user name and password.

Example

```
dm liststreams --user user1 --password alpha --database
 qlarius_cm@dim2009 --server winxbox1:8080
```

Working with DM – Some Typical Development Scenarios

This section will walk you through a number of common development scenarios and tell you how to use the Developer Command-Line in those scenarios. The following common tasks will be covered:

- Creating a stream to contain your code base. See "Creating and Deleting Streams" on page 579.
- Importing your initial code base into a stream. See "Importing Your Code Into the Stream" on page 579.
- Getting a working copy of the code base. See "Obtaining a Working Copy of the Code for Modification" on page 579.
- Committing your changes back to the repository. See "Making Changes and Committing them Back to the Repository" on page 580.
- Automatically merging changes from the repository into your working copy and dealing with any conflicts. See "Handling Conflicts" on page 581.
- Using requests to control your change sets. See "Using Requests to Control Change Sets" on page 584.

Creating and Deleting Streams

Before you start to work with your code, you must first put a copy of it into the Dimensions CM repository. To do this you need to create a development stream that will manage that code. This can be done with the `createstream` or `cs` command.

In this example, we will be using a stream called MESSENGER to manage a simple MSN and YAHOO IM application.

To create this empty stream we would use a command such as:

```
% dm cs messenger -m "A simple IM application"
Stream 'messenger' created
```

If we wished to later delete this stream, we could do this using the following command:

```
% dm ds messenger
Stream 'messenger' deleted
```

Our stream is now created and ready for use. The next step is to import our initial code base into this stream for development use.

Importing Your Code Into the Stream

Once we have created our stream for managing the code, we need to import the code base that we want into that stream. The simplest way of doing this is to use the *import* command. This command will take all the uncontrolled files from a specified area on disk and recursively add them to a stream.

In our example, the messenger application is located in the directory `d:\messenger`. At present, this directory contains just the sources, makefiles and help text that we want to put under control. So, we now do this with a series of commands such as the following:

```
% cd d:\messenger
% dm sw messenger .
% dm import -m "Initial code commit to the repository"
```

Which displays the following:

```
Processing files...
Adding          ChatSessions.cpp
Adding          ChatSessions.h
```

Our code is now under control and ready for use.

Obtaining a Working Copy of the Code for Modification

Once our messenger application has been imported into the repository, we want to be able to obtain a working copy of our code on our development machine for building and modification. We can do this by the use of the *get*, or – more likely – the *update* command.

The *get* command will copy the code from the repository into our working area and abort if any conflicts are detected, whilst the *update* command will automatically merge any

conflicts unless a line-level conflict is detected. As we are more likely to want to automatically merge our code, we will focus here on using *update*.

After creating a working area on disk called `D:\messenger1_0\`, we set that directory as our default and run the *update* command

```
% mkdir D:\messenger1_0\
% cd D:\messenger1_0\
% dm sw messenger D:\messenger1_0\
```

Which displays the following:

```
Stream 'messenger' is now using 'D:\messenger1_0\'

% dm update
```

Which displays the following:

```
Processing files...
A       ChatSessions.cpp;messenger#1
A       ChatSessions.h;messenger#1
```

As we have just created our work area, all the code is added from the repository without any conflicts.

Making Changes and Committing them Back to the Repository

Once we have retrieved the code we can now build it and start adding new features.

Let's assume that we modify the MSN support to include the ability to use VOIP and Webcam protocols. In doing this we modify existing source code and make files plus add 1 new source file. Once we have finished coding and unit tested our changes we want to commit them back to the repository.

To check what the state of our work area is in comparison with the repository, we first run the *status* command to see what we have changed.

```
% dm status
```

Which displays the list of files and their status:

```
Processing files...
M       makefile
M       MsnChatSessions.cpp
M       MsnChatSessions.h
?       vc80.pdb
?       WebVoip.cpp
?       win32x86_debug/ChatSessions.obj
```

This shows us that we have modified 3 controlled files (tagged M) and have a number of uncontrolled files that are not currently in the repository (tagged "?").

As we have both uncontrolled new source files and the results of our build in the working area, we need to decide what we want to add to the repository so that we can commit a change set that we are happy with. As we are not interested in controlling the build results, we decide we are only interested in adding the new file `WebVoip.cpp` to our commit. We can do this by the use of the *add* command.

```
% dm add WebVoip.cpp
```

Which displays the following:

```
Processing files...
A       webvoip.cpp
add complete
```

This will schedule the new file `WebVoip.cpp` to be included when we next commit our change set, but exclude all the built targets, which we are not interested in controlling.

When we are finally ready to commit our changes, we simply run *commit* as follows

```
% dm commit -m "Add initial prototype VOIP support"
```

Which displays the following:

```
Processing files...
Adding          WebVoip.cpp
Adding          makefile
Adding          MsnChatSessions.h
Adding          MsnChatSessions.cpp
```

This has now committed our VOIP support to the repository. If we use the status command again we can now see that the only difference is the built files that we decided not to control.

```
% dm stat
```

Which displays the following:

```
Processing files...
?       vc80.pdb
?       win32x86_debug/ChatSessions.obj
?       win32x86_debug/FileTransferRequests.obj
```

We have now successfully performed a simple development cycle!

Handling Conflicts

Although in our small example it is unlikely that more than one developer will be working on the application, in the real world, conflicting changes to the same file is something that most developers need to deal with on a regular basis. The *update* command helps to manage such conflicts.

The *update* command will not only populate a work area with the latest contents of the repository, but will also perform an automatic merge if any conflicts occur between files. There are two main types of conflict that *update* can handle. These can be resolved in the following ways:

- Automatic merges – where the file on disk is merged cleanly with the latest version in the repository
- Manual merges – where the file on disk has line-level conflicts with the latest version in the repository and requires the developer to manually merge the file

All merges are performed through the use of a DIFF3 script and conflicts are tagged using standard DIFF3 formats (DIFF3 is a standard comparison utility; see the DIFF3 documentation for more details).

Ways to Identify conflicts

In our example, we will assume that two developers have been working on our code and have introduced conflicts that one developer later needs to resolve.

A conflict can be found in several ways. The main ones however are

- using the *status* command to notify you of conflicts
- attempting a *commit* and having it fail due to conflicts

Examples of these are shown below:

- Using status:

```
% dm st
```

Which displays the following:

```
Processing files...
C       UtilityFuncs.cpp;messenger#2 : File is modified locally and
    in the repository
?       vc80.pdb
C      WebVoip.cpp;messenger#2 : File is modified locally and in the
    repository
?       win32x86_debug/ChatSessions.obj
?       win32x86_debug/FileTransferRequests.obj
?       win32x86_debug/messappcmd.obj
?       win32x86_debug/MessengerApps.obj
```

- Using commit:

```
% dm ci -m "Show a conflict"
```

Which displays the following:

```
Processing files...

Conflicts have been detected between your area and the repository.
C               WebVoip.cpp : Repository file has been updated
COR3200234E Error: Delivery aborted.
```

If this situation occurs, then the developer will need to merge his or her code with the latest tip before they can do a successful commit.

Useful Merging Commands

There are several commands that the developer doing the merge may find useful during this process.

diff command

The first of these is *diff*. This command allows the developer to show what the differences between their local version and the latest in the repository are. An example of this might be:

```
% dm diff WebVoip.cpp
C       WebVoip.cpp
```

Which displays the following:

```
== Differences detected between files were ====================
Local file     : d:/messenger1_1/WebVoip.cpp
```

```
Repository file: d:/messenger1_1/WebVoip.cpp.cm.latest

18a19,22
> /// \namespace WebUtils
> namespace WebUtils
> {
>
32,33d35
< /// This is a comment which I have added
< /// For the purpose of showing a conflict
47a50
> }

=============================================================
```

As you can see, the difference between the two versions is shown using the standard diff style output. (Note – It is possible to configure the Developer Command-Line to use different tools other than `dm diff`).

update command

As well as using `dm diff` to show the code differences, it is also possible to run *update* in a dry run mode to show what it would do, but not actually do it. An example of this is shown below:

```
% dm up --dry-run
```

Which displays the following:

```
Processing files...

G       WebVoip.cpp: Local file would have been automatically merged
    with the repository version
C       UtilityFuncs.cpp : Automatic merge would have resulted in
    conflicts

Dry run complete - no files modified
```

The "G" code indicates that a file would have been merged cleanly, while the "C" code indicates that manual intervention would have been required.

Performing the actual update would give our developer the following results:

```
% dm up
```

Which displays the following:

```
Processing files...

C       UtilityFuncs.cpp : Line level conflicts have been detected

=============================================
Please select one of the following options -
     (i)gnore the conflict for the moment,
     (a)ccept the repository version,
     (u)se your local version,
     (s)how differences,
     (m)erge the files and invoke an editor to resolve any conflicts
     (g)o ahead with the merge and resolve conflicts later
     (q)uit
   i
```

```
G       WebVoip.cpp: Local file was automatically merged with the
    repository version
S       UtilityFuncs.cpp: Conflict was skipped
```

Resolving Conflicts

When line-level conflicts are detected, our developer is presented with a number of options that they can apply to that conflict. Being unsure of the details, our developer decides to ignore the conflict for the momen and talk to the person who did the original change later on. However, the developer could have used the (s) option to show the conflicts, decide what to do, and then use (m) to manually edit the conflicts.

Let's assume that our developer, after consultation with the author of the other change, decides that the repository version is the correct one and overrides their local changes. They therefore re-run the update command specifying that it should automatically accept the repository version for that file.

```
% dm up UtilityFuncs.cpp --accept repository
```

Which displays the following:

```
Processing files...

A       UtilityFuncs.cpp;messenger#2: Overwriting locally modified
    file
1 file transfer operation succeeded, 0 failed.
```

When they re-run *status*, our developer now sees that only the automatically merged source file is now due to be committed, and they commit that file.

```
% dm st WebVoip.cpp UtilityFuncs.cpp
```

Which displays the following:

```
Processing files...
M       WebVoip.cpp
```

They run the commit:

```
% dm commit WebVoip.cpp -m "Commit merged change"
```

Which results in the following:

```
Processing files...
Adding          WebVoip.cpp
```

The conflicts are now successfully resolved.

Using Requests to Control Change Sets

Change sets – or sequences of file changes – can be controlled with the use of requests. Most of the Developer Command-Line commands support the notion of using a request or list of requests to control the file versions that are processed by that command. For example, doing a command such as:

```
% dm get --requestid product_def_1
```

would only get the file versions that were related as in-response-to PRODUCT_DEF_1 and in the stream that you are currently working with. The same would also apply to a command like *status* where the files compared against an area would be those files related *In-Response-To* to the requests that you specified, plus any that were also related as dependent. This means that you could take a change set that had been controlled by a request – say a patch – and apply it to your area by using a command such as:

```
% dm up --requestid patch_def_20
```

and have that patch set automatically merged into your working copy.

When committing changes to the repository from your working area, you can specify one or more requests that you would like these changes to be logged against. Your changes will be related as *In-Response-To* those specified requests, for example:

```
% dm commit include/sys/*.h --requestid patch_def_15 -m "Changes for
    supporting AIX"
```

All the .h files committed will be related to request `patch_def_15`.

With regards to our example, let's say that one of the developers writing our IM application adds Doxygen comments to some of the header files in the work area. The developer then commits this change set against a specific request as follows:

```
% dm ci --requestId minako_scr_212 -m "Doxygen comments for
    standards"
```

Which results in the following:

```
Processing files...
Adding          Msnlocale.h
Adding          Mutex.h
Adding          NetworkOpsSSL.h
Adding          MsnChatSessions.h
```

Another developer then wants to pick up these changes and merge them into their local working copy. However, they only want to pick up these changes, and not the latest copy.

To do this they would run an update command specifying the request against which those changes were made so that only those files are updated in the work area. The developer first performs a dry run of the update, as it would be advisable to check the results:

```
% dm up --requestId minako_scr_212 --dry-run
```

Which displays the following:

```
Processing files...

C       Msnlocale.h : Automatic merge would have resulted in
    conflicts
C       Mutex.h : Automatic merge would have resulted in conflicts
C       MsnChatSessions.h : Automatic merge would have resulted in
    conflicts
G       FileTransferRequests.h: Local file would have been
    automatically merged with the repository version
C       MessengerApps.h : Automatic merge would have resulted in
    conflicts
C       Msn.h : Automatic merge would have resulted in conflicts
C       NetworkOps.h : Automatic merge would have resulted in
```

```
    conflicts
U       NetworkOpsSSL.h;messenger#2
U       Threads.h;messenger#2
U       UtilityFuncs.h;messenger#2

Dry run complete - no files modified
```

They can check the changes and then run the *update* for real.

What are the Commands I Can Perform?

This section consists of an overview of the commands you can perform with the DM command-line client. More details are given for each command in the "Command List" on page 589.

Managing Streams

The commands for creating and deleting streams are:

- The **createstream** command creates a new stream and defines its associated work area. You can create an empty stream or use an existing stream or baseline to populate it with its initial content. A version branch name for the items in the stream is also created. See "createstream – Create a new stream in the repository" on page 595
- The **deletestream** command can be used to delete a stream if it has not had any versioned content created in it. See "deletestream – Delete a stream" on page 598

These commands are for locking a stream to prevent users from updating its content in the repository. See

- The **lockstream** command locks a stream. See "lockstream – Lock a stream" on page 617
- The **unlockstream** command unlocks a stream. See "unlockstream – Unlock a stream" on page 627.

Working with Streams

To manage your stream settings, the following commands are available.

- the **switchstream** command enables you to set your default stream and work area. See "switchstream – Switch the working stream" on page 625.
- The **getinfo** command enables you to find out what your current stream and work area are. See "getinfo – Get current stream and work area details" on page 608.

To commit content to a stream, or to populate a work area from a stream, the following commands are available:

- The **update** command updates the files in your work area with the current contents of a stream, resolving any conflicts as it does so. See "update – Update a local work area" on page 628.

- The **deliver** command updates the stream with the files in your work area. See "deliver – Deliver content to a stream" on page 599.
- The **export** command copies the latest files in a stream into an area without creating any metadata. This is a useful feature if you want to create a copy of the code for release purposes. See "export – Exports a non-versioned copy of a stream to a work area" on page 604.
- The **import** command allows you to import new files into a stream that have not been previously controlled. See "import – Import uncontrolled content into a stream" on page 609.
- The **add** command schedules previously uncontrolled files to be added to the stream on the next commit or deliver command. See "add – Schedule a file or directory to be added to the repository" on page 589.
- The **commit** (or deliver) command updates the stream in the repository with changes from the local work area. See "commit – Commit content to a stream" on page 593.
- The **revert** command resets the files in your work area to the latest state of the files in the stream, overwriting any local content or refactoring changes where possible. See "revert – Revert local changes made to a work area" on page 621.

Commands to help you resolve conflicts between a stream and your work area are:

- The **diff** command displays any differences in file content between the files in the work area and the same files in the stream. See "diff – Display the code differences between a stream and a local work area" on page 602
- The **update** command updates a work area with the latest content of the stream, automatically resolving conflicts where possible, or assisting you to resolve them when line-level conflicts occur. See "update – Update a local work area" on page 628.

Alphabetic List of Commands

An alphabetical list of the commands available are listed in the table below. Some commands have one or more aliases, which are alternative names that can be used for when specifying the command.

Command	See the following ...	Alias
add	"add – Schedule a file or directory to be added to the repository" on page 589	
annotate	"annotate – Add a comment to a file" on page 591	ann
cat	"cat – Display the contents of a file" on page 592	
commit	"commit – Commit content to a stream" on page 593	checkin ci
createstream	"createstream – Create a new stream in the repository" on page 595	create cs
delete	"delete – Schedule deletions" on page 597	delete rm, del erase

Command	See the following ...	Alias
deletestream	"deletestream – Delete a stream" on page 598	ds
deliver	"deliver – Deliver content to a stream" on page 599	de
diff	"diff – Display the code differences between a stream and a local work area" on page 602	di
export	"export – Exports a non-versioned copy of a stream to a work area" on page 604	export
get	"get – Get the contents of a stream to a local work area" on page 606	checkout co
getinfo	"getinfo – Get current stream and work area details" on page 608	
import	"import – Import uncontrolled content into a stream" on page 609	
list	"list – List the contents of a stream" on page 611	ls
listbaselines	"listbaselines – List the baselines in the repository" on page 612	lsb
liststreams	"liststreams – List the streams in the repository" on page 614	lss
lockfile	"lockfile – Lock a file in the repository" on page 616	lock
lockstream	"lockstream – Lock a stream" on page 617	locks
log	"log – Display the repository history for a file" on page 618	
logout	"logout – Clear the Dimensions login credentials" on page 619	lo
move	"move – Move (rename) files" on page 620	mv ren
revert	"revert – Revert local changes made to a work area" on page 621	
status	"status – Report on changes to a local work area" on page 623	stat st
switchstream	"switchstream – Switch the working stream" on page 625	switch sw
unlockfile	"unlockfile – Unlock a file in the repository" on page 626	unlock
unlockstream	"unlockstream – Unlock a stream" on page 627	unlocks
update	"update – Update a local work area" on page 628	up

Command List

add – Schedule a file or directory to be added to the repository

This command will schedule any uncontrolled files or directories in a work area to be added to the repository the next time you issue a *commit* command.

Note that when adding a directory, everything underneath the specified directory will be scheduled for addition.

Files that are up to date with the repository will be ignored.

Alias None

Format
```
dm add <file1> <file2>...
   [--directory <directory>, --area <directory>
   --recursive, -R
   --non-recursive, -N
   --quiet, -q
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Optionally specifies the names of files or directories/folders to be added.

  You can use "*" as a wildcard in these names to represent one or more characters.

- `--directory <directory> or --area <directory>`

  Specifies the name of the folder/directory identifying the work area from which files are to be added.

  If it is not specified, then the default work area associated with the stream will be used.

- `--recursive or -R`

  This option will add files and subdirectories of the specified directories. This is the default.

- `--non-recursive or -N`

  This option will only process the files and directories for the specified path. It will not include subdirectories.

- --quiet or -q

  Only print critical messages.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm add insurance_v2 c:\work\insurance\`

annotate – Add a comment to a file

This command adds a comment to a file, or set of files.

Alias ann

Format
```
dm annotate <file1[;rev]> <file2[;rev]> ...
   [--stream <stream_name>
   --comment <comment>, -m <comment>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file1[;rev]> <file2[;rev]> ...`

  specifies the name(s) of the file(s) to which you want the comment to be added.

  If specified, `rev` determines which revision to process. The default is the revision in the work area, otherwise the latest in the repository is used.

- `--stream <stream_name>`

  Specifies the name of the stream for which you want to annotate files.

  If it is not specified, then your current stream will be used.

- --comment <comment> or -m <comment>

  The comment that you wish to add to the files.

  This can be a maximum of 2k.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm annotate DBIO.java FileIO.java;2 --stream PATCH_1 --comment Updated
for patch 1
```

cat – Display the contents of a file

This command will display the contents of one or more files to standard out.

The format is:

Format

```
dm cat <file1[;rev]> <file2[;rev]> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file1[;rev]> <file2[;rev]> ...`

  specifies the name(s) of the file(s) you wish to be deploy.

  If specified, `rev` determines which revision to process. The default is the latest revision in the repository.

- `--stream <stream_name>`

  Specifies the name of the stream to which the file(s) belong.

  If a stream is not specified, then the current stream will be used.

- `--directory <directory> or --area <directory>`

  Optionally, specifies the folder/directory of a local work area from which the specified files are to be listed. The default is the current directory.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm cat DBIO.java FileIO.java --stream MAINLINE --area c:\work\devarea\
```

commit – Commit content to a stream

This command will commit any changed content from a local work area to a stream. If there are no differences, then the command will do nothing.

Note that the commit functionality is a subset of the deliver functionality.

The status of the file being committed is represented by the following keys:

- Adding – A file/directory in your work area has been added to the stream.
- Importing – A file already in the repository has been imported into the stream.
- Renaming – A file/directory has been renamed/moved in the stream to reflect changes in your work area.
- Deleting – A file/directory that has been deleted from your local work area has also been deleted from the stream.
- Skipping – A file/directory in your local work area has been skipped because it was not scheduled for addition or deletion. Note this status will only be shown if you have specified the `verbose` parameter.

Files that are up to date with the stream will be ignored.

Aliases ci

Format
```
dm commit <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --recursive, -R
   --non-recursive, -N
   --contributors <contributorstream1>, <contributorstream2>,...
   --comment <comment>, -m <comment>
   --quiet, q
   --verbose, -v
   --requestId <request> ...
   --remove <stream_name>, --remove <revision>
   --changes-only
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file1> <file2> ...`

  Optionally, specifies the names of files or directories/folders to commit to the stream.

  You can use "*" as a wildcard in these names to represent one or more characters.

- `--stream <stream_name>`

  Specifies the name of the stream to commit the content to.

  If it is not specified, then the current stream will be used.

- `--directory <directory> or --area <directory>`

Specifies the name of the folder/directory of the local work area whose content is to be committed.

If it is not specified, then the current directory will be used.

- `--recursive or -R`

  Include the subfolders and files of the specified directories.

  This is the default.

- `--non-recursive or -N`

  Only process the files and directories at the specified path. (Do not include subfolders and files).

- `--contributors <contributorstream1>, <contributorstream2>, ...` or `--contributors all`

  Allow content from other streams that might be in your work area to be committed as well as files owned by the target stream. `contributorstreams` can either be a comma-separated list of streams or the value "`all`", which means consider content from all streams.

- `--comment <comment> or -m <comment>`

  Use the specified comment when creating new item revisions. If no comment is specified, then Dimensions CM will generate a default.

- `--quiet or -q`

  Only print critical messages.

- `--verbose or -v`

  Print additional information about the update process.

- `--requests <request-id> ...`

  Relate the changes that are committed as *In-Response-To* the requests specified.

- `--remove <stream_name> or --remove <revision>`

  If comitting deletions, then specify the scope of that deletion.

  - If `stream_name` is specified, then all revisions of the deleted file will be removed from the stream. This is the default.
  - If `revision` is specified, then only the file revision that was deleted from your local area will be removed from the stream.

- `--changes-only`

  Only files which have been changed or moved will be processed. Any scheduled additions, deletions or import of content of contributing streams will be ignored.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm commit --stream MAINLINE --area c:\work\devarea\`

createstream – Create a new stream in the repository

This command will create a new stream in the repository.

Aliases cs, create

Format
```
dm createstream <stream_name> <area>
   [--product <productName>
   --from-stream <fromStreamName>
   --from-baseline <fromBaselineName>
   --branch  <branchName>
   --description <description>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

Where:

- `<stream_name>`

  Is the name of the new stream.

- `<area>`

  Is the name of the folder/directory of the work area to be associated with the stream.

- --product <`productName`>

  Specifies the name of the product that will own this stream.

  If this is omitted, then the owning product will default to the product owning the baseline or stream specified in the `--from-stream` or `--from-baseline` option.

  If neither of those options are specified, then the owning product will be the product owning your default stream.

- `--from-stream <fromStreamName>`

  If specified, is the name of an already existing stream from which item revisions will be used to initially populate the new stream.

- `--from-baseline <fromBaselineName>`

  If specified, is the name of an existing baseline whose item revisions will be used to initially populate the new stream.

- `--branch <branchName>`

  Specifies the name of the branch to be used for new item revisions created in this stream.

  This must either be a new branch in the repository, or an existing branch that has not been used for any existing item revisions.

  If omitted, defaults to the value of *stream_name*.

- `--description <description>`

  Is a description for the stream.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm create STREAM_B c:\work\stream_b\ --from-stream MAINLINE
```

delete – Schedule deletions

This command will delete the controlled files or directories from the local work area and schedule those files or directories to be removed from the repository when you next perform a *commit*. It will also remove a file or directory scheduled to be added to the repository.

In the case of uncontrolled files that have been scheduled for addition, this command will leave those files or directories in the local work area, but will remove them from the scheduled additions when you next commit.

Note that when deleting a directory, everything underneath the specified directory will be deleted and scheduled for removal. Removals are non-recursive by default.

Alias del, rm, erase

Format
```
dm delete <file> ...
   [--directory <directory>, --area <directory>
   --recursive, -R
   --non-recursive, -N
   --quiet, -q
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file> ...`

  Optionally, specifies the names of files or directories/folders to be deleted.

  You can use "*" as a wildcard in these names to represent one or more characters.

- `--directory <directory> or --area <directory>`

  Specifies the folder/directory for the work area to which the deletions are to be scheduled.

  If it is not specified, then the current directory will be used.

- `--recursive or -R`

  Include the subfolders and files of the specified directories.

- `--non-recursive or -N`

  Only process the files and directories at the specified path. (Do not include subfolders and files.)

  This is the default.

- --quiet or -q

  Only print critical messages.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm delete include/calcs.h --stream STREAM_A --area c:\work\stream_a\
```

deletestream – Delete a stream

This command will delete the specified stream from the repository.

Alias ds

Format
```
dm ds <stream_name>
   [--user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<stream_name>`

  specifies the name of the stream you want to delete.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm ds STREAM_C
```

deliver – Deliver content to a stream

This command will deliver any changed content, or any new content scheduled for delivery, from a local work area to a stream in the repository. If there are no changes, then the command will do nothing.

The status of the file being delivered is represented by the following keys:

- Adding – A file/directory in your work area has been added to the stream.
- Importing – A file already in the repository has been imported into the stream.
- Renaming – A file/directory has been renamed/moved in the stream to reflect the changes in your work area.
- Deleting – A file/directory that has been deleted from your local work area has also been deleted from the stream.
- Skipping – A file/directory in your local work area has been skipped in the process because it was not scheduled for addition or deletion.

  Note that this status will only be displayed if you have specified the `verbose` parameter.

Files that are up to date with the stream will be ignored.

Alias de

Format dm deliver <file1> <file2> ...

```
[--stream <stream_name>
--directory <directory>, --area <directory>
--recursive, -R
--non-recursive, -N
--contributors <contributorStream1> <contributorStream2> ...
--comment <comment>, -m <comment>
--quiet, q
--verbose, -v
--requestId <request-id1> <request-id2> ...
--add
--del
--remove <stream_name>, --remove <revision>
--changes-only
--removal_scope <revision> or <stream>
--user <user_name>
--password <password>
--database <database>
--server <server_name:port>
--card]
```

where

- <file1> <file2> ...

  Optionally, specifies the names of files or directories/folders to deliver to the stream.

  You can use "*" as a wildcard in these names to represent one or more characters.

- `--stream <stream_name>`

  Specifies the name of the stream to update the work area from.

  If it is not specified, then the current stream will be used.

- `--directory <directory>`

  Specifies the name of the folder/directory of the work area to be updated from the stream.

  If it is not specified, then the default work area associated with the stream will be used.

- `--recursive or -R`

  Include the subfolders and files of the specified directories.

  This is the default.

- `--non-recursive or -N`

  Only process the files and directories at the specified path. (Do not include subfolders and files.)

- --contributors <contributorStream1>, <contributorStream2>, ... or --contributors all

  Allow content from other streams that might be in your work area to be committed as well as files owned by the target stream. contributorstreams can either be a comma-separated list of streams or "all" which means consider content from all streams.

- `--comment <comment> or-m <comment>`

  Use the specified comment when creating new item revisions. If no comment is specified, then a default will be used.

- --quiet or -q

  Only print critical messages.

- `--verbose or -v`

  Print additional information about the update process.

- `--requests <request-id1> <request-id2> ...`

  Relate changes in the stream as *In-Response-To* the requests specified.

- `--add`

  Allow the delivery of both changed content in your work area and new content previously unscheduled for delivery to a stream.

- `--del`

  Allow the delivery of both changed content in your work area and deleted content previously unscheduled for removal from a stream.

- `--remove <stream_name> or --remove <revision>`

  If delivering deletions, then specify the scope of that deletion.

  - If `stream_name` is specified, then all revisions of the deleted file will be removed from the stream. This is the default.
  - If `revision` is specified, then only the file revision that was deleted from your local area will be removed from the stream.

- `--changes-only`

  Only files which have been changed or moved will be processed. Any scheduled additions, deletions or import of content of contributing streams will be ignored. This option is mutually exclusive to --add or --del.

- `--removal_scope [revision|stream]`

  If delivering deletions, specify the scope of the deletions. If `revision` is specified, then only the file revision that was deleted from your local work area will be removed from the stream. If `stream` is specified, then all revisions of the deleted file will be removed from the stream. This is the default.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm create STREAM_B c:\work\insurance\`

diff – Display the code differences between a stream and a local work area

This command will display the code differences between the local work area and a stream. If there are no code differences, then the command will do nothing.

Aliases di

Format
```
dm diff <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --diff-cmd <diffcmd>
   --recursive, -R
   --non-recursive, -N
   --requestId <request1> <request2> ...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Specifies the names of files and/or folders to be compared.

- `--stream <stream_name>`

  Specifies the name of the stream whose content in being compared.

  If it is not specified, then the current stream will be used.

- `--directory <directory> or --area <directory>`

  Specifies the name of the folder in the work area to be compared with the stream.

  If it is not specified, then the current folder will be used.

- `--diff-cmd <diffcmd>`

  Override the default diff command used with the one specified in `diffcmd`.

- `--recursive or -R`

  Include the subfolders and files of the specified folders.

  This is the default.

- `--non-recursive or -N`

  Only process the files and directories at the specified path. (Do not include subfolders and files).

- `--requestId <request1> <request2> ...`

  Only compare the files that are related as *In-Response-To* the requests specified.

  Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used.

  This option cannot be used with the <file> parameter.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm diff --stream STREAM_A --area c:\work\insurance\
```

export – Exports a non-versioned copy of a stream to a work area

This command will populate a local work area with a non-versioned copy (i.e. a copy of the files in the stream without any associated metadata) of the latest contents of a stream. If the specified work area is already defined and in use, then the export will fail and you will need to use a clean work area.

Aliases export

Format
```
dm export <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --verbose, -v
   --force
   --recursive, -R
   --non-recursive, -N
   --expand
   --requestId <request-id1> <request-id2> ...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Optionally, specifies the names of files or directories/folders to retrieve from the stream.

- `--stream <stream_name>`

  Specifies the name of the stream to export the files from.

  If it is not specified, then the current stream will be used.

- `--directory <directory> or --area <directory>`

  Specifies the name of the folder/directory of the work area to be populated from the stream.

  If it is not specified, then the current directory will be used.

- `--verbose or -v`

  Print additional information about the update process.

- --force

  Force export to use the specified work area, even if it is already populated. Existing files will be overwritten.

- `--recursive or -R`

  Include the subdirectories and files of the specified directories.

  This is the default.

- `--expand`

Perform file header substitution on exported files. The default is not to perform any header expansion.

- `--non-recursive or -N`

  Only process the files and directories for the specified path (do not include subfolders and files).

- `--requestId <request-id1> <request-id2> ...`

  Only export the files that are related as *In-Response-To* the requests specified.

  Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used.

  This option cannot be used with the <file> parameter.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm export --stream STREAM_A --directory c:\work\testarea\`

get – Get the contents of a stream to a local work area

This command will refresh a local work area with a working copy of the latest contents of a stream. If a conflict is found between files that have been modified locally and those in the stream, then the get will fail and you will need to run the update command to process these conflicts.

Aliases checkout, co

Format
```
dm get <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --verbose, -v
   --recursive, -R
   --non-recursive, -N
   --requestId <request1> <request2>...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Optionally, specifies the names of files or directories/folders to retrieve from the stream.

- `--stream <stream_name>`

  Specifies the name of the stream to update the work area from.

  If it is not specified, then the current stream will be used.

- `directory <directory> or --area <directory>`

  Specifies the name of the folder/directory for the work area to be refreshed from the stream.

  If it is not specified, then the default work area associated with the stream will be used.

- `--verbose or -v`

  This option prints additional information about the export process.

- `--recursive or -R`

  This option will update the subdirectories and files of the specified directories.

  This is the default.

- `--non-recursive or -N`

  This option only processes the files and directories for the specified path. It does not include subdirectories.

- `requestId <request1> <request2> ...`

  Only retrieve the files that are related as *In-Response-To* the requests specified.

Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used.

This option cannot be used with the <file> parameter.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm get reports\src --stream STREAM_A --directory c:\work\insurance\
```

getinfo – Get current stream and work area details

This command will display the user's current working stream, work area, and default request (if any) details. It has no parameters.

Format

```
getinfo
```

Example

```
dm getinfo
   [--user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm getinfo
```

will display:

```
Current stream is set to 'STR1' using the working location 'c:\patches\'
```

import – Import uncontrolled content into a stream

This command will import any uncontrolled content from a work area into a stream in the repository. If there are no differences, then the command will do nothing.

The status of the file being imported is represented by the following keys:

'Adding' – A file/directory in your work area has been added to the stream.

'Importing' – A file already in the repository has been imported into the stream.

Files that are up to date with the repository will be ignored.

Alias import

Format
```
dm import <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --recursive, -R
   --non-recursive, -N
   --contributors <contributorStream> ...
   --comment <comment>, -m <comment>
   --verbose, -v
   --quiet, q
   --requestId <request1> <request2> ...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Specifies the names of files or directories/folders to be imported.

  You can use "*" as a wildcard in these names to represent one or more characters.

- `--stream <stream_name>`

  Specifies the name of the stream to update the work area from.

  If it is not specified, then the current stream will be used.

- `directory <directory> or--area <directory>`

  Specifies the name of the folder/directory of the work area from which the files are to be imported.

  If it is not specified, then the current directory will be used.

- `-recursive or -R`

  Include the subdirectories and files of the specified directories..

  This is the default.

- `--non-recursive or -N`

  Only process the files and directories at the specified path (do not include subfolders and files).

- --contributors <contributorstream1>, <contributorstream2>, ... or --contributors all

Allow content from other streams that might be in your work area to be imported as well as files owned by the target stream. `contributorstreams` can either be a comma-separated list of streams orthe value "`all`", which means consider content from all streams.

- --comment <comment> or -m <comment>

  Use the specified comment when creating new item revisions. If no comment is specified, then a default will be used.

- --quiet, -q

  Only print critical messages.

- `--verbose or -v`

  Print additional information about the import process.

- `--requestId <request1> <request2> ...`

  Relate files that are imported as *In-Response-To* the requests specified.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm import --stream STREAM_A --directory c:\work\stream_a\
```

list – List the contents of a stream

This command lists information about the contents of a stream.

Aliases ls

Format
```
dm list [<directory1> <directory2> ...]
   [--stream <stream_name>
   --verbose, -v
   --recursive, -R
   --non-recursive, -N
   -l
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <directory1> <directory2> ...

  Specifies the name(s) of the directory(s) you want to include in the listing.

  If this is not specified, all files in the stream will be listed.

- `--stream <stream_name>`

  Specifies the name of the stream whose content you want to list.

  If this is not specified, then your current default stream will be used.

- `--verbose or -v`

  Specifying this option includes additional information about the files, such as item specification, creator and status.

- `-recursive or -R`

  Include the subdirectories and files of the specified directories.

- `--non-recursive or -N`

  Only process the files in the specified directories. Do not include subfolders and files.

  This is the default.

- `-l`

  This prints a long listing that includes additional file information.

  If a file has been locked in a stream, this will be indicated with the use of a '*' after the filename.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm list qlarius\interfaces\ --stream STREAM_A --verbose
```

listbaselines – List the baselines in the repository

This command will list the release baselines that are present in the repository based on user defined selection criteria.

Baselines are listed in order of creation date.

Aliases lsb

Format
```
dm listbaselines <pattern> ...
   [--status <status>
   --date <dateRange>
   --show <number>
   -l
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<pattern> ...`

  Specifies a regular expression pattern matching filter for the baseline names, for example `*rel`, that can be used to refine the list of baselines.

- `--status <status>`

  Specifies a regular expression pattern matching filter for the status, For example `*PEN`, that can be used to refine the list of baselines.

  This filter can also be used to display all active and inactive baselines by specifying the following keywords:

  - `ACTIVE` —This will list all baselines that are at an *open* state.
  - `INACTIVE` —This will list all baselines that are at a *closed* or *rejected* state.

- `--date <dateRange>`

  Specifies a date filter to be applied to the creation date of the baselines that are displayed.

  You can use >, < and = operators. You cannot use multiple combinations of these operators, but you must specify only one of them in the expression.

  The format used for the date must be one of the following:

  DD/MON/YYYY, DD-MON-YYYY, DD MON YYYY, MM/DD/YYYY, MM-DD-YYYY,

  MM DD YYYY, DD-Month-YYYY, DD/Month/YYYY, DD Month YYYY and HH24:MI:SS.

  Example date expressions are:

  ```
  > 20-jun-2009

  > 4 feb 2009 01:00
  ```

- `--show <number>`

  An integer specifying the maximum number of baselines to be displayed.

- `-l`

This prints a long listing that includes additional baseline information.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm listbaselines --date > 12-may-2009 --show 30
```

liststreams – List the streams in the repository

This command will list the streams that are present in the repository based on user-defined selection criteria. Streams are listed in order of creation date.

Aliases lss

Format
```
dm liststreams <pattern> ...
   [--status <status>
   --date <dateRange>
   --show <number>
   -l
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<pattern> ...`

  Specifies a regular expression pattern matching filter for the stream names, such as `*REL`, that can be used to refine the list of streams.

- `--status <status>`

  Specifies a regular expression pattern matching filter for the status, such as `*PEN` that can be used to refine the list of streams.

  This filter can also be used to display all active and inactive baselines by specifying the following keywords -

  - `ACTIVE` —This will list all baselines that are at an *open* state.
  - `INACTIVE` —This will list all baselines that are at a *closed* or *rejected* state.

- `--date <dateRange>`

  Specifies a date filter to be applied to the creation date of the streams that are displayed.

  You can use >, < and = operators. You cannot use multiple combinations of these operators, but you must specify only one of them in the expression.

  The format used for the date must be one of the following:

  DD/MON/YYYY, DD-MON-YYYY, DD MON YYYY, MM/DD/YYYY, MM-DD-YYYY,

  MM DD YYYY, DD-Month-YYYY, DD/Month/YYYY, DD Month YYYY and HH24:MI:SS.

  Example date expressions are:

  ```
  > 20-jun-2009

  > 4 feb 2009 01:00
  ```

- `--show <number>`

  An integer specifying the maximum number of streams to be displayed.

- `-l`

  This prints a long listing that includes additional stream information.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm lss --status ACTIVE
```

lockfile – Lock a file in the repository

This command will lock one or more files in the repository. Locking files prevents other users from delivering those files to the stream in the repository.

Alias lock

Format
```
lockfile <file>
   [--stream <stream_name>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file> ...`

  Specifies the names of the files to lock. Only the latest revisions in the target stream can be locked, any other revisions specified will be rejected

- `--stream <stream_name>`

  Specifies the name of the stream to lock the files in. If it is not specified, then the current stream will be used.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm lock include/calcs.h --stream STREAMA
```

lockstream – Lock a stream

This command will lock the specified stream in the repository.

Alias locks

Format
```
lockstream <stream_name>
   [--user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<stream_name> ...`

  specifies the name of the stream you want to lock.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm lock MAINSTREAM
```

log – Display the repository history for a file

This command displays the version history for the specified file(s) in a stream.

Format

```
dm log <file1> <file2> ...
   [--stream <stream_name>
   --date <dateRange>
   -l
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file1> <file2>...`

  specifies the name(s) of the file(s) whose history you want to display.

- `--stream <stream_name>`

  Specifies the name of the stream whose file history you want to list.

  If it is not specified, then the current stream will be used.

- `--date <dateRange>`

  Specifies a date filter to be applied to the history that is displayed.

  You can use >, < and = operators. You cannot use multiple combinations of these operators, but you must specify one of them in the expression.

  The format used for the date must be one of the following:

  DD/MON/YYYY, DD-MON-YYYY, DD MON YYYY, MM/DD/YYYY, MM-DD-YYYY,

  MM DD YYYY, DD-Month-YYYY, DD/Month/YYYY, DD Month YYYY and HH24:MI:SS.

  Example date expressions are:

  ```
  > 20-jun-2009
  ```

  ```
  > 4 feb 2009 01:00
  ```

- `-l`

  This prints a long listing that includes additional file information.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm log --stream STREAM_A --date > 01-04-2009
```

logout – Clear the Dimensions login credentials

This command will clear your Dimensions login credentials. The next time you run a command you will be required to login.

Format `dm logout`

[--user <username>
`--quiet, q]`

where

- `--user <username>`

This is the username whose Dimensions credentials should be cleared.

- --quiet, -q

  Only print critical messages.

Example `dm logout --user USER1`

move – Move (rename) files

This command will rename a file or folder/directory or move it from one location to another.

The move command has a number of restrictions:

- You cannot move a controlled file onto another controlled file
- If you are moving a file from one directory to another, you must specify the name of the target directory AND file, not just the file
- The command does not support wildcards

Aliases mv, ren

Format
```
dm move <source> <dest>
   [--user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<source> ...`

  Specifies the name of the source file or folder/directory to move or rename.

- `<dest> ...`

  Specifies the name of the destination file or folder/directory of the move or rename.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm move qlarius\interfaces\ qlarius\common\
```

revert – Revert local changes made to a work area

This command will undo any local changes made in a work area. Optionally, you can also choose to have the changes made to the stream in the repository applied to the work area.

Locally changed files will be overwritten and local refactoring changes such as moves and deletions will be reverted as far as possible.

NOTE This command may not be able to restore all file/directory movements or deletions that might have been done in your area.

Aliases None

Format

```
dm revert <file1> <file2> ...
   [--stream <stream_name>
   --directory <directory>, --area <directory>
   --latest
   --recursive, -R
   --non-recursive, -N
   --verbose, -v
   --quiet, q
   --requestId <request1> <request2> ...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Optionally, specifies the names of files or directories/folders to retrieve from the stream.

- `--stream <stream_name>`

  Specifies the name of the stream to update the work area from.

  If it is not specified, then the current stream will be used.

- `--directory <directory> or--area <directory>`

  Specifies the name of the folder/directory of the work area to be updated from the stream.

  If it is not specified, then the current directory will be used.

- `--latest`

  This option will update the work area with any changes made to the stream in the repository as well as reversing changes made to the work area.

  If this option is not specified, the work area is not updated with changes made to the stream in the repository.

- `--recursive or -R`

This option will include the subdirectories and files of the specified directories. This is the default.

- `--non-recursive or -N`

  This option only processes the files and directories for the specified path. It does not include subdirectories.

- `requestId <request1> <request2> ...`

  Only revert the files that are related as *In-Response-To* the requests specified.

  Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used.

  This option cannot be used with the <file> parameter.

- `--verbose or -v`

  Print additional information about the update process..

- --quiet or -q

  Only print critical messages.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm revert calcs.h payroll.h --directory C:\mainline`

status – Report on changes to a local work area

This command reports on changes that have taken place in a local work area. It can be run in two different modes:

- Offline – This will only report on local changes that have been made in the work area. This is the default mode.
- Online – This will report the differences between the local work area and a repository stream.

If there have been no changes made, then the command will do nothing.

The status of a file is represented by the following keys:

- ? – A local file not under version control.
- ! – A local file has been renamed/moved.
- A – A file is in the stream that would be added to your local work area.
- C – A local file is in conflict with the file in the stream.

  The exact reason for the conflict will be provided as additional information.
- D – A file that has been deleted from your local work area, but is still in the stream.
- M – A file that has been locally modified.

Aliases stat, st

Format

```
dm status <file1> <file2> ...
   [-u, --show-updates
   --stream <stream_name>
   --directory <directory>, --area <directory>
   --recursive, -R
   --non-recursive, -N
   --verbose, -v
   --requestId <request1> <request2> ...
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- <file1> <file2> ...

  Optionally, specifies the names of files or directories/folders to compare.
- -u or--show-updates

  Switches the command to online mode. This will compare the repository against your local work area.
- --stream <stream_name>

  Specifies the name of the stream to use for the comparison.

  If it is not specified, then the current stream will be used.
- --directory <directory> or --area <directory>

Specifies the name of the folder/directory of the work area to be compared with the stream.

If it is not specified, then the default work area associated with the stream will be used.

- `--recursive or -R`

  This option will revert subdirectories and files of the specified directories.

  This is the default.

- `--non-recursive or -N`

  This option only processes the files and directories for the specified path. It does not include subdirectories.

- `--verbose or -v`

  This option prints additional information about the processing.

- `--requestId <request1> <request2> ...`

  Specify this option to only compare the files that are related as *In-Response-To* the requests specified.

  Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used. This option cannot be used with the <file> parameter or in offline mode.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example

```
dm status reports\src --stream STREAM_A --area c:\work\devarea\
```

switchstream – Switch the working stream

This command changes your current working stream and work area.

Aliases switch, sw

Format
```
switchstream <stream_name> [<directory>]
   [--requestId <request>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<stream_name> ...`

  specifies the name of the stream you want to set as your current working stream.

- `<directory>`

  Optionally, specifies the name of the folder/directory for the work area to be associated with the stream.

- `--requestId <request>`

  Optionally, specifies a default working request to set for the stream.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm switch STREAM_A c:\work\devarea\ --requestId QLARIUS_CR_112
```

unlockfile – Unlock a file in the repository

This command will unlock one or more files in the repository.

Alias unlock

Format
```
unlockfile <file>
   [--stream <stream_name>
   --user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<file> ...`

  Specifies the names of the files to unlock.

- `--stream <stream_name>`

  Specifies the name of the stream to unlock the files in. If it is not specified, then the current stream will be used.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm unlock include/calcs.h --stream STREAMA
```

unlockstream – Unlock a stream

This command will unlock a specified stream in the repository.

Alias unlocks

Format
```
unlockstream <stream_name>
   [--user <user_name>
   --password <password>
   --database <database>
   --server <server_name:port>
   --card]
```

where

- `<stream_name>`

  specifies the name of the stream you want to unlock.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example
```
dm unlock STREAM_A
```

update – Update a local work area

This command will update a local work area with a working copy of the latest contents of a stream. If conflicts are found between files that have been modified locally and in the stream, then this command will help to resolve those conflicts. If there are changes in the stream which do not conflict with the local work area, then these will also be retrieved from the stream.

Code resolution is done either automatically (via the use of a DIFF3 script) or interactively. For each file that is encountered with a conflict, the following codes are used to indicate how that conflict was resolved.

- A – A file from the repository was added to the work area.
- U – The local file was updated from the repository.
- S – The conflict was skipped either due to refactoring changes or by user decision.
- G – The conflict was resolved automatically and no further action is needed.
- C – Attempting to resolve the conflict automatically failed and manual intervention was required.
- D – The file has been deleted from your local work area.

When a conflict is encountered that needs manual interaction, a number of options will be presented as to the action that could be taken for that conflict.

These options are:

- i – ignore the conflict for the moment
- a – accept the repository version
- u – use your local version,
- s – show differences,
- m – merge the files and invoke an editor to resolve any conflicts
- g – go ahead with the merge and resolve conflicts later
- q – quit

Aliases up

Format dm update <file1> <file2> ...

```
[--stream <stream_name>
--baseline <baseline_name>
--directory <directory>, --area <directory>
--recursive, -R
--non-recursive, -N
--quiet, -q
--dry-run
--accept LOCAL or REPOSITORY or AUTOMERGE
--diff-cmd <diffcmd>
--editor-cmd <editorcmd>
--requestId <request1> <request2> ...
--user <user_name>
--password <password>
--database <database>
--server <server_name:port>
```

```
--force-add
--no-add
--force-delete
--no-delete
--card]
```

where

- `<file> ...`

  Optionally, specifies the names of files or directories/folders to retrieve from the stream.

- `--stream <stream_name>`

  Specifies the name of the stream to update the work area from.

  If it is not specified, then the current stream will be used.

- `--baseline <baseline_name>`

  Specifies the name of the baseline to update the work area from.

- `--directory <directory>, --area <directory>`

  Specifies the name of the folder/directory of the work area to be updated from the stream or baseline.

  If it is not specified, then the current will be used.

- `--recursive or -R`

  This option will update subdirectories and files of the specified directories. This is the default.

- `--non-recursive or -N`

  This option only processes the files and directories for the specified paths. It does not include subdirectories.

- `--quiet or -q`

  This option only prints critical messages.

- `--dry-run`

  This option dry-runs the process and prints out the actions that would take place without processing any files.

- `--accept LOCAL or --accept REPOSITORY or --accept AUTOMERGE`

  This option runs the update in non-interactive mode specifying the action that will be taken when an unresolvable conflict is encountered according to the option:

  - REPOSITORY: The locally modified file will be overwritten by the file from the stream.
  - LOCAL: The locally modified file will be skipped and the conflict postponed until it can be manually resolved.
  - AUTOMERGE: All text files will be merged automatically, whether there are conflicts or not. It is then the responsibility of the user to resolve those conflicts manually.

    The files will only be reported by the tool as being locally modified, so take care if you use this option.

- `--diff-cmd <diffcmd>`

  Override the default diff command used with the one specified in DIFFCMD.

- `--editor-cmd <editorcmd>`

  Override the default editor command used with the one specified in `EDITORCMD`.

- `--manual`

  Treat automatic merges as if they were manual and present the same options. This option is useful if you wish to review the results of the automatic merge before accepting them.

- `--requestId <request1> <request2> ...`

  Only process files that are related as *In-Response-To* the requests specified.

  Requests that are related as dependent to the specified requests will also be processed unless the --non-recursive option is used.

  This option cannot be used with the <file> parameter.

- `--force-add`

  When updating a work area from another stream (a stream that is not associated with the work area), place all new work files from the stream into the work area. This includes files that were previously removed from the stream that is associated with the work area. This option is ignored when updating from the stream that is associated with the work area.

- `--no-add`

  When updating a work area from another stream (a stream that is not associated with the work area), do not place any new files into the work area. This option is ignored when updating from the stream that is associated with the work area.

- `--force-delete`

  When updating a work area from another stream (a stream that is not associated with the work area), delete all managed files from the work area that do not exist in the stream you are updating from. This includes files that have not previously been removed from the stream that is associated with the work area. This option is ignored when updating from the stream that is associated with the area.

- `--no-delete`

  When updating a work area from another stream (a stream that is not associated with the work area), do not delete files from the work area. This option is ignored when updating from the stream that is associated with the area.

For details of the global options --user, --password, --database, --server, and --card, see "How do I Connect to the Database?" on page 577.

Example `dm update c:\work\devarea\ --stream STREAM_A`

Configuring the Developer Command-Line

This section details how to configure various behaviors of the Developer Command-Line (DM). There is a configuration file which controls the behavior of DM defined by the symbol DM_DAEMON_CONFIG_FILE in the DM.CFG file, located in <DM_ROOT>/dm.cfg

This file can be modified in the following ways to control how the DM utility works.

- Changing the tool that is used for comparing and merging files.

  The code differencing which is performed by the diff command and the update (show differences) option is controlled by the:

  ```
  DIFFCMD = <difftool>
  ```

  Flag. By default, this is set to use the Dimensions *diff* tool, but can be changed to another differencing tool if required. The utility that is used must take files as an option in the same way as diff.

- Changing the editor that is used.

  The editor that is used is controlled by the

  ```
  EDITOR = <editor>
  ```

  Flag. By default, this is not set. The tool itself defaults to "notepad" on Windows, and "vi" on UNIX. Enter the path of the executable file for the editor if you want to use a different one. If the editor is set, then it must be one which is run asynchronously, rather than synchronously, i.e. the editor must block if run from the operating system command prompt, not immediately return. If a synchronous utility is used, then the edit will fail.

Index

A

ABL command
    constraints 35
    description 34
    example 34
    syntax 34
AC command
    constraints 37
    examples 36
    syntax 36
access, to nodes (authorizing) 60
ACDI command
    description 38
    example 38
    syntax 38
ACDWS command
    description 40
    example 39
    syntax 39
ACF command
    constraints 41, 159
    description 41
    example 41
    syntax 41
ACL
    attributes 199
    permissions 199
    setting on directories 198
actioning
    baselines or items, to a new lifecycle 34
    items 44
    project (AWS) 61
    requests 36
ADF command
    constraints 42
    description 42
    example 42
    syntax 42
AGRPU command
    constraints 43
    description 43
    example 43
    syntax 43
AI command
    constraints 45
    example 44
    syntax 44
AIWS command
    constraints 48
    description 48
    example 46
    syntax 46
ANNOTATE command 49
annotate(DM command-line client) 589, 591
APNO command
    constraints 51
    example 51
    syntax 51
area commands
    create area (CA) 76
    create library cache area (CLCA) 110
    list areas (LA) 258
    list library cache areas (LLCA) 274
    populate areas (PA) 322
    relate area to project (RAWS) 347
    remove area (RA) 341
    remove library cache area (RLCA) 389
    unrelate area from project (XAWS) 524
    update area (UA) 454
    update library cache area (ULCA) 484
ART commands
    create archive (CAR) 79
    delete archive (DAR) 143
    read archive tape (RAT) 346
    remove archived item (RAI) 343
    remove archived material selected by archive (RAMA) 344
    remove archived material selected by product (RAMP) 345, 395
    retrieve offline archive (ROA) 395
    transfer baseline in (TBI) 452
    transfer baseline out (TBO) 453
attributes
    baselines, updating 459
    design parts, updating 494
    items, updating 479
    multivalue and multiline 17
    parts 494
    projects/streams, setting 448
    users 517
AUDIT command
    example 52
    syntax 52
AUGRP command
    constraints 54
    description 54
    example 54

syntax 54
AUPG command 55
AUR command
constraints 59
example 56, 57
syntax 56
AUTH command
description 60
examples 60
syntax 60
authorizing, access to nodes 60
automatic job triggering, see crontab
AWS command
constraints 61
description 61
example 61
syntax 61

B

baseline commands
action, to a new lifecycle (ABL) 34
compare (CMP) 117
create (CBL) 82
create merged (CMB) 113
create revised (CRB) 126
delete (DBL) 146
deploy baseline (DPB) 192
deploy item (DPI) 194
list baselines (LSBL) 292
relate baseline to baseline (RBBL) 350
relate baseline to project (RBWS) 353
relate project to project (RWWS) 417
report (RPT) 405
unrelate baseline from baseline (XBBL) 525
unrelate baseline from project (XBWS) 527
unrelate project from project (XWWS) 539
BC command
constraints 63
examples 62
syntax 62
BI command
constraints 65
example 64
syntax 64
BLD command
description 71
example 66
parameters 66
syntax 66
BLDB command
description 75
example 72
syntax 72
branch commands
define version branch (DVB) 214
remove (RMVB) 394
set, flags (SVBF) 444
browsing
items 64
print or requests 62
Build commands
audit a build (AUDIT) 52
build (BLD) 66
build baseline (BLDB) 72
create a Dimensions CM build project (DBPROJ) 147
define a Dimensions CM project (DPROJ) 202
delete a Dimensions CM build project (RBPROJ) 352
deliver build targets (DBT) 148
export build configuration (EXPORT) 232
extract (check out) build configuration (ECFG) 224
import build configuration (IMPORT) 257
list baselines (LSBL) 292
list contents of STAGE_CATALOGUE table (LSTG) 295
list Dimensions CM build projects (LBPROJ) 265
list Dimensions CM projects and build projects (LPROJ) 285
populate build area (PBA) 323
relate area to build configuration (RABC) 342
remove a Dimensions CM project (RPROJ) 400
remove area from build configuration (XABC) 523
return (check in) build configuration (RCFG) 358
update a Dimensions CM build project (UBPROJ) 460
update a Dimensions CM project (UPROJ) 512

C

CA command
constraints 78
description 78, 89
example 76
syntax 76
CAR command
example 79
syntax 79
case translation, in standalone utilities 543
cat (DM command-line client) 592
CBA command
constraints 81, 88
syntax 80
CBDB command

example 81
syntax 81, 88
CBL command
constraints 87
example 82
syntax 82
CBP command
examples 88
CC command
constraints 92
example 90
syntax 90
CCO command
example 93
syntax 93
CCS command 94
CCST command
example 95
syntax 95
CCU command
constraints 96
description 96
example 96
syntax 96
certificates 27
CFS command
example 97
syntax 97
CGRP command
constraints 98
description 98
example 98
syntax 98
checking in, items 378
checking out
baseline items 235
items 240
project items 248
CHMOD command
constraints 99
description 99
example 99
syntax 99
CI command
constraints 104
example 100
CINS command
example 105
syntax 105
CIP command 106
CIU command
constraints 109
example 108
syntax 108
CLCA command
constraints 111
description 110
example 110
syntax 110
CMB command
constraints 115
description 114
example 113
syntax 113
CMD
sequential commands 12
CMD command
example 116
syntax 116
CMDCLIENT, see DMCLI
CMP command
constraints 117
example 117
syntax 117
CNC command
example 118
syntax 118
CNDO command
syntax 119
CNN command
example 120
syntax 120
CNSJ command
constraints 121, 137
description 121
example 121
CNWO command
example 122
syntax 122
code metrics
update 465
command mode 12
introduction 12
command syntax
compound fields 17
continuation indicator 15
ellipses 14
exceptions
UNIX system 15
Windows system 15
function mnemonics 13, 14
multivalue and multiline attributes 17
optional qualifiers 14
parameters 13, 14
qualifiers 13
quoted strings, using 15
required qualifiers 14
syntax diagram 14
using comments in command files 17
command-line 12
command-line logging and usage analysis
all commands run by all users 29

audit trail of commands run 29
users who connect to Dimensions CM 30
comments
using in command files 17
commit (DM command-line client) 593
comparing
baselines 117
structures 117
compound fields, in command syntax 17
connection process for Window Dimensions CM client, see DMCLI
continuation indicator, in command syntax 15
conventions, typographical 13
converting metadata 545
COS command
example 123
syntax 123
CP command
constraints 124
example 124
syntax 124
CPV command
constraints 125
example 125
syntax 125
CRB command
constraints 133
description 130
error conditions 132
example 126
syntax 126
create stream (CS) 135
createstream (DM command-line client) 595
creating
baselines 82
build area 80
design part variants 125
design parts 124
items 100
merged baselines 113
project directories 142
requests 90
revised baselines 126
variant structures 140
credential set commands
CCS 94
DCS 151
LCS 268
UCS 468
crontab
dm_full_mail 553
dm_incremental_mail 553
CRSD command
syntax 134
CS command
syntax 135
CSJ command
description 138
example 138
syntax 138
CUSR command
description 139, 148
syntax 139
CVS command
constraints 141
example 140
syntax 140
CWSD command
constraints 142
example 142
syntax 142

D

daemon (Developer Command Line) 576
DAR command
example 143
syntax 143
data formats
defining 153
flags, setting (SDF) 423
DBC command
syntax 144
DBDB command
example 145
syntax 145
DBL command
constraints 146
example 146
syntax 146
DBPROJ command
syntax 147
DBTS command
example 148
syntax 148
DCH command
constraints 149
syntax 149
DCO command
example 150
syntax 150
DCS command 151
DCST command
example 152
syntax 152
DDF command
constraints 154
description 154
example 153
syntax 153
defining

data formats 153
item relations 165
product libraries 197
products 183
projects 216
user roles 212
version branches 214
delegating
items 170
requests 168
delete (Developer Command Line) 597
deletestream (DM command-line client) 598
deleting
baselines 146
design part variants 207
items 162
products 215
project directories 220
requests 149
deleting metadata 550
deliver (DM command-line client) 599
DELIVER command
description 155
example 156
syntax 155
design part commands
allocate part numbers (APNO) 51
attributes, updating (UPA) 494
create (CP) 124
create variant (CPV) 125
create variant structure (CVS) 127, 140
delete part variant (DPV) 207
move relationship (MDR) 304
relate (RP) 396
relate to requests 397
report current (RCP) 361
report design structure (RDS) 367
report, part numbers (RPNO) 399
suspend, variant (SPV) 440
unrelate (URP) 514
unrelate, from requests (XPCD) 534
update (UP) 493
update part numbers (UPNO) 510
design parts
relationship, moving 304
Developer Command Line
delete 597
DFS command
syntax 160
DGRP command
constraints 161
description 161
example 161
syntax 161
DI command
constraints 163
example 162
syntax 162
diff (DM command-line client) 602
Dimensions CM client commands, see DMCLI
Dimensions CM execution, ending 231
Dimensions CM standalone utilities, see utilities
DINS command
example 164
syntax 164
DIR command
constraints 165
description 165
example 165
syntax 165
directories, setting
see SET command
displaying metadata 547
DLCA command
example 166, 172
syntax 166
DLGC command
constraints 169
examples 168
syntax 168
DLGI command
constraints 171
example 170
syntax 170
DLGS command 172
DM command-line client
annotate 589, 591
cat 592
commit 593
createstream 595
deletestream 598
deliver 599
diff 602
export 604
get 606
getinfo 608
import 609
introduction 576
list 611
listbaselines 612
liststreams 614
lockfile 616, 626
lockstream 617
log 618
move 620
revert 621
status 623
switchstream 625
unlockstream 627
DM_AUDIT_CMD_USAGE 29
dm_auto_action utility
about 551

example (1st form) 552
example (2nd form) 552
syntax (1st form) 552
syntax (2nd form) 552
DM_ESCAPE_CHAR environment variable 16
dm_full_mail utility
see also, dm_incremental_mail utility
about 552
crontab 553
syntax 552
dm_incremental_mail utility
see also, dm_full_mail utility
about 552
crontab 553
syntax 552
DM_TEMPLATE_CATALOG 376, 391
DMCLI 12
connection process for UNIX Dimensions CM client 22
command-line login 23
-con login 23
GUI login 22
server login 23
connection process for Windows Dimensions CM client 19
command-line login 19
-con login 19, 21
server login 22
examples 24
running from a Dimensions CM client
syntax 19
DMDB 23
dmmeta 544
dmpasswd
about 554
constraints 554
examples 555
syntax 554
DNC command
example 180
syntax 180
DNDO command
syntax 181
DNN command
example 182
syntax 182
DNP command
constraints 184
example 183
syntax 183
DNWO command
example 185
syntax 185
DOS command
syntax 186
DOWNLOAD command
description 191
examples 187
syntax 187
DPB command
constraints 193
description 192
example 192
syntax 192
DPI command
constraints 196
description 194
example 194
syntax 194
DPL command
constraints 199
example 197
item library files, protecting 198
syntax 197
DPR command
constraints 201
description 200
example 200
syntax 200
DPROJ command
syntax 202
DPRP command
constraints 206
description 205
examples 203
syntax 203
DPV command
constraints 207
example 207
syntax 207
DREL command
constraints 208
syntax 208
DRSD command
syntax 209
DS command
constraints 210
example 210
syntax 210
DSJ command
constraints 211
description 211
example 211
DUR command
constraints 212
example 212
syntax 212
DUSR command
description 213
syntax 213
DVB command
constraints 214

description 214
example 214
syntax 214
DWP command
constraints 215
example 215
syntax 215
DWS command
constraints 219
description 137, 218
example 135, 216
syntax 216
DWSD command
constraints 221
example 220
syntax 220

E

ECDI command
description 223
example 222
syntax 222
ECFG commands 224
EI command
constraints 228
examples 225
syntax 225
ellipses, in command syntax 14
e-mail reminders, pending lists 552
encrypting base database names, connection strings, and passwords 554
environment variable
DM_AUDIT_CMD_USAGE 29
DM_ESCAPE_CHAR 16
DM_TEMPLATE_CATALOG 376, 391
DMDB 23
EOL command
CMD_TRACE 428
escape character, in quoted strings 15
ESJ command
constraints 230
description 230
example 230
executing
command files 116
execution authority, in standalone utilities 543
execution, ending 231
EXIT command
syntax 231
export (DM command-line client) 604
EXPORT command
description 233
syntax 232
extracting (checking out) items 225

F

FAT file system 198
FBI command
constraints 237
description 237
examples 235
syntax 240
FCDI command
description 239
example 238
syntax 238
fetching (getting)
baseline items 235
items 240
project items 248
FI command
constraints 244
examples 240
syntax 240
FIF command
constraints 246
description 245
example 245
syntax 245
file names, spaces in 28
FRC command
constraints 247
description 247
example 247
syntax 247
function mnemonics, in command syntax 13, 14
FWI command
constraints 250
description 250
examples 248
syntax 252

G

GENCERT command 251
get (DM command-line client) 606
get item, see FI command
getinfo (DM command-line client) 608
GREP command
constraints 254
examples 252
syntax 252

H

HELP command
description 255
example 255

syntax 255
HIDE command
syntax 256

I

import (DM command-line client) 609
IMPORT command
description 257
syntax 257
INFO command
CMD_TRACE 428
installation package, create 106
introduction, command mode 12
item commands
action (AI) 44
action, to a new lifecycle (ABL) 34
add revision to project (AIWS) 44, 46
assign data formats (ADF) 42
attributes, updating (UIA) 479
browse (BI) 64
cancel update (CIU) 100, 108
create (CI) 100
define data formats (DDF) 153
define relations (DIR) 165
delegate (DLGI) 170
delete (DI) 162
extract (check out) (EI) 225
fetch (get) (FI) 240
fetch (get) baseline items (FBI) 235
fetch (get) project items (FWI) 248
find item file (FIF) 245
merge item revisions (MI) 310
move (change) item type (MIT) 315
move, to another part (MIP) 313
relate, item to item (RI) 383
relate, to part (RIP) 385
relate, to requests (RICD) 381
remove, data formats (RMDF) 393
remove, relation definition (RIR) 386
remove, revision from project (RIWS) 387
report current (RCI) 359
return (check in) (RI) 378
suspend (SI) 437
unrelate, from item (XII) 531
unrelate, from part (XIP) 532
unrelate, from requests (XICD) 529
update (UI) 475
item files, finding 245
item library files
protecting, from unauthorized changes (on Windows) 198
items
checking in 378
checking out 235, 240, 248
data formats, assigning 42
defining data formats 153
relation definition, removing 386
relations, defining 165
revision, adding to project 46
revision, default 24
update, canceling 108
updating, without changing item revision 25

L

LA command
constraints 259
description 258
example 258
syntax 258
LAST command 260
LAVC command 261
LBA command
syntax 263
LBDB command
example 264
syntax 264
LBPROJ command
syntax 265
LCK command
constraints 266
description 266
example 266
syntax 266
LCO command
example 267
syntax 267
LCS command 268
LCST command
example 269
syntax 269
LFS command
example 270
syntax 270
LGRP command
constraints 271, 297
description 271
syntax 271
library cache area, purge (PLCA) 331
LINS command
example 273
syntax 273
list (DM command-line client) 611
listbaselines (DM command-line client) 612
listing metadata 550
liststreams (DM command-line client) 614
LLCA command
constraints 274
description 274

example 274
syntax 274
LMNR command
description 275
syntax 275
LNC command
example 276
syntax 276
LNDO command
syntax 277
LNN command
example 278
syntax 278
LNWO command
example 279
syntax 279
lockfile (DM command-line client) 616, 626
lockstream (DM command-line client) 617
log (DM command-line client) 618
LOG command 280
LOS command
example 283
syntax 283
LPRIV command
description 284
syntax 284
LPROJ command
syntax 285
LPRP command
constraints 286
description 286
examples 286
syntax 286
LPRT command
syntax 287
LPSP command
constraints 272, 288
description 272, 288
examples 272, 288
syntax 272, 288
LRSD command
example 290
syntax 290
LSAR command 291
LSBL command
description 292
example 292
syntax 292
LSJ command
description 294
example 293
LSTG command
description 295
example 295
syntax 295
LWS command
constraints 299
example 298
syntax 297, 298
LWSD command
constraints 301
example 300
syntax 300

M

mail reminders, pending lists 552
MCPC command
constraints 302
example 302
syntax 302
MCSC command
constraints 303
example 303
syntax 303
MDR command
constraints 304
example 304
syntax 304
MERGE command 305
merging
baselines 113
item revisions 310
projects 319
metadata
converting 545
deleting 550
displaying 547
listing 550
updating 549
metadata utility 544
MI command
constraints 312
description 312
example 310
syntax 310
MIP command
constraints 314
example 313
syntax 313
MIT command
constraints 315
example 315
syntax 315
move (DM command-line client) 620
moving
design part relationships 304
item types 315
items, to another part 313
project directories 321
requests 317

requests, to primary catalog 302
requests, to secondary catalog 303
multiline attributes, in command syntax 17
multivalue attributes, in command syntax 17
MVC command
constraints 318
example 317
syntax 317
MWS command
constraint 320
example 319
syntax 319
MWSD command
constraints 321
example 321
syntax 321

N

network commands
create a file system (CFS) 97
create a network node (CNN) 120
create a network node connection (CNC) 118
create a network node object (CNDO) 119
create a network object (CNWO) 122
create a new codeset (CCST) 95
create a new contact (CCO) 93
create an operating system (COS) 123
delete an existing codeset (DCST) 152
delete an existing contact (DCO) 150
delete an existing file system (DFS) 160
delete an existing network node (DNN) 182
delete an existing network node connection (DNC) 180
delete an existing network node object (DNDO) 181
delete an existing network object (DNWO) 185
delete an existing operating system (DOS) 186
edit an existing base database entry in network administration table (UBDB) 458
edit an existing database instance entry in network administration table (UINS) 483
edit an existing file system (UFS) 473
edit an existing network node connection (UNC) 487
edit an existing network object (UNWO) 491
edit an existing operating system (UOS) 492
edit an existing resident software definition (URSD) 515
list existing base database entries in network administration table (LBDB) 264
list existing codesets (LCST) 269
list existing contacts (LCO) 267
list existing database instance entries in network administration table (LINS) 273
list existing file systems (LFS) 270
list existing network node connections (LNC) 276
list existing network node connections (LNWO) 279
list existing network node objects (LNDO) 277
list existing network nodes (LNN) 278
list existing network protocols (LPRT) 287
list existing operating systems (LNOS) 283
list existing resident software definitions (LRSD) 290
register a base database entry in network administration table (CBDB) 81, 88
register a database instance entry in network administration table (CINS) 105
register a resident software definition (CRSD) 134
unregister an existing base database entry in network administration table (DRDB) 145
unregister an existing database instance entry in network administration table (DINS) 164
unregister an existing resident software definition (DRSD) 209
update a codeset (UCST) 470
update an existing contact (UCO) 467
nodes
access to, authorizing 60
notifications
list mail notification rules (LMNR) 275
subscribe to notification rule (SUB) 443
unsubscribe from notification rule (USUB) 516
NTFS file system 198

O

operating systems
differences 28
spaces, in file names 28
Windows UNC path 28
optional qualifiers, in command syntax 14

P

PA command
constraints 322

description 322
example 322
syntax 322
parameters, in command syntax 13, 14
parts, see design parts; design part commands
PBA command
syntax 323
pcms_auto_action, see dm_auto_action utility
PCMS_COMMAND_STATISTIC published view 29
PCMS_ESCAPE_CHAR, see DM_ESCAPE_CHAR environment variable
pcms_full_mail, see dm_full_mail utility
pcms_incremental_mail, see dm_incremental_mail utility
PEND (baseline) command
constraints 329
description 328
example 328
syntax 328
PEND (item) command
constraints 327
description 326
example 326, 330
syntax 326, 330
PEND (request) command
constraints 325
description 324
example 324
syntax (1st form) 324
syntax (2nd form) 324
pending lists, sending by e-mail 552
PLCA command
constraints 331
description 331
examples 331
syntax 331
prcs utility
constraints 564
description 562
environment 564
examples 556
files, Dimensions CM 564
named branches, Dimensions CM 564
notes 565
options, Dimensions CM 561
revisions 563
subcommands 557
print command, setting
see SET command
PRIV command
constraints 339
description 339
examples 338
syntax 338
privileges
assigning groups to a user (AGRPU) 43
assigning users to a group (AUGRP) 54
create group (CGRP) 98
delete group (DGRP) 161
list groups (LGRP) 271
manage privileges (PRIV) 338
update group (UGRP) 474
product commands
define libraries (DPL) 197
define new (DNP) 183
define whole (DWP) 215
product control plan, report (RCPC) 398
project commands
action (AWS) 61
create directory (CWSD) 142
define new (DWS) 216
delete directory (DWSD) 135, 216, 220
list (LWS) 297, 298
list directories (LWSD) 300
lock (LCK) 266
remove (RWS) 416
project directories
listing 300
moving 321
project permissions, setting 451
projects
listing 297, 298
locking 266
z/OS, specifying file names 28
psccs utility
constraints 572
description 571
example 565
files, Dimensions CM 572
named branches, Dimensions CM 572
options, Dimensions CM 570
revisions 572
subcommands 566

Q

qualifiers 13
optional 14
required 14
QUIT command 340
quoted strings
escape character 15
UNIX 15
using in commands 15
Windows 15

R

RA command
constraints 341

description 341
example 341
syntax 341
RABC command
description 342
syntax 342
RAI command
example 343
syntax 343
RAMA command
example 344
syntax 344
RAMP command
example 345
syntax 345, 395
RAT command
example 346
syntax 346
RAWS command
constraints 348
description 348
examples 347
syntax 347
RBA command
syntax 349
RBBL command
constraints 350
description 350
example 350
syntax 350
RBCD command
constraints 351
example 351
syntax 351
RBPROJ command
syntax 352
RBWS command
constraints 353
description 353
example 353
syntax 353
RCCD command
constraints 354
example 354
syntax 354
RCDI command
description 356
example 355
syntax 355
RCDWS command
description 357
example 357
syntax 357
RCFG command
description 358
syntax 358
RCI command
constraints 360
example 359
if /NEW is omitted 360
if /NEW is specified 359
syntax 359
RCP command
constraints 362
example 361
if /NEW is omitted 362
if /NEW is specified 361
syntax 361
RCS front end for Dimensions CM, see prcs utility
RCSJ command
constraints 363
description 363
example 363
RCU command
constraints 364
description 364
example 364
syntax 364
RDEL command
description 365
example 365
syntax 365
RDS command
constraints 369
example 367
if /NEW is omitted 368
if /NEW is specified 367
syntax 367
REL command
constraints 372, 374
example 370
syntax 370
relating
baselines to requests 351
design part 396
item to item 383
item to part 385
item to requests 381
part to requests 397
requests to request 354
release commands
delete (DREL) 208
release (REL) 370
removing
build area 349
item or request data formats 393
item relation definition 386
item revision from project 387
project 416
version branch 394
RENAME command
constraints 374, 378

description 374
example 373
syntax 373
replace command (GREP) 252
reporting
baselines 405
current items 359
current parts 361
design structures 367
part numbers 399
product control plan 398
requests 401
run, user-defined (RUR) 412
REQC command
constraints 375
description 375
examples 375
syntax 375
request baseline templates 86
request commands
action (AC) 36
assign data formats to request types (ACF) 41
browse or print (BC) 62
create (CC) 90
define data formats (DDF) 153
delegate (DLGC) 168
delete (DCH) 149
item, relating to (RICD) 381
move (MVC) 317
move, to primary catalog (MCPC) 302
move, to secondary catalog (MCSC) 303
relating (RCCD) 354
remove, data formats (RMDF) 393
report (RPT) 401
unrelate, from request (XCCD) 528
update (UC) 461
request types
data formats, assigning 41
requests
actioning (by date or attribute value), see dm_auto_action utility
data formats, removing 393
defining data format for request type 153
required qualifiers 14
return (check in) items 378
revert (DM command-line client) 621
REXEC command
description 376
syntax 376
RI command
constraints 380
example 378
syntax 378
RICD command
constraints 382
example 381
syntax 381
RII command
description 384
example 383
syntax 383
RIP command
example 385
syntax 385
RIR command
constraints 386
description 386
example 386
syntax 386
RIWS command
constraints 388
description 388
example 387
syntax 387
RLCA command
constraints 389
description 389
example 389
syntax 389
RLIST command
description 390
syntax 390
RMDF command
constraints 393
description 393
example 393
syntax 393
RMVB command
constraints 394
description 394
example 394
syntax 394
ROA command
example 395
syntax 395
RP command
constraints 396
example 396
syntax 396
RPCD command
constraints 397
example 397
syntax 397
RPCP command
constraints 398
example 398
syntax 398
RPNO command
constraints 399
example 399
syntax 399

RPROJ command
  syntax 400
RPT (Baseline) command
  constraints 406
  example 405
  syntax 405
RPT (requests) command
  additional parameters 402
  constraints 404
  example 401
  syntax 401
RRCD command
  constraints 408
  description 407
  example 407
RREG command
  description 409
  syntax 409
RSJ command
  constraints 410
  description 410
  example 410
RSTAT command
  description 411
  syntax 411
running commands from a Dimensions CM client, see DMCL
RUR command
  constraints 413
  example 412
  syntax 412
RWCD command
  constraints 415
  description 415
  example 415
  syntax 415
RWS command
  constraints 416
  example 416
  syntax 416
RWWS command
  constraints 417
  description 417
  example 417
  syntax 417

S

SCCS front end to Dimensions CM, see psccs utility
schedule job commands
  Cancel Schedule Job (CNSJ) 121
  Create Schedule Job (CSJ) 138
  Delete Schedule Job (DSJ) 211
  Edit Schedule Job (ESJ) 230
  List Scheduled Jobs (LSJ) 293
  Relate Command to Schedule Job (RCSJ) 363
  Run Schedule Job (RSJ) 410
  Unrelate Command from Schedule Job (UCSJ) 469
  View Log of Schedule Job Execution (VLSJ) 521
SCWS command
  example 419
  syntax 419
SDF command
  constraints 424
  description 423
  example 423
search command (GREP) 252
set CMD_TRACE command, see SET command
SET command
  CMD_TRACE 428
  constraints 430
  DIRECTORY 428
  example 428
  OVERWRITE 428
  PRINTER 428
  syntax 428
set DIRECTORY, see SET command
set EOL command, see SET command
set INFO command, see SET command
set OVERWRITE command, see SET command
set PRINTER, see SET command
setting
  current project 419
  data format flags 423
  project file names 446
  project permissions 451
  project/stream attributes 448
  version branch flags 444
SF command 431
SHELVE command 432
SHOW command
  syntax 436
SI command
  constraints 438
  example 437
  syntax 437
Smart Card
  Developer Command Line 578
Smart Card authentication 21
SPSP command
  description 439, 441
  example 439
  syntax 439
SPV command
  constraints 440
  description 440
  example 440
  syntax 440

standalone utilities, see utilities
status (DM command-line client) 623
stream
  create (CS) 135
  delete 210
  deliver 155
structured information return commands
  SAVE 418
  SSPM 442
SUB command
  description 443
  example 443
  syntax 443
suspending
  design part variants 440
  items 437
SVBF command
  constraints 445
  description 444
  example 444
  syntax 444
SWF command
  constraints 447
  example 446
  syntax 446
switchstream (DM command-line client) 625
SWS command
  constraints 450
  description 450
  examples 297, 448
  syntax 448
SWSP command
  syntax 451
syntax conventions, see command syntax
syntax diagram, for commands 14

T

TBI command
  example 452
  syntax 452
TBO command
  example 453
  syntax 453
triggering, automatic jobs, see crontab
typographical conventions 13

U

UA command
  constraints 456
  description 456
  example 454
  syntax 454
UBA command
  syntax 457
UBDB command
  syntax 458
UBLA command
  constraints 459
  description 459
  example 459
  syntax 459
UBPROJ command
  syntax 460
UC command
  constraints 464
  example 461
  syntax 461
UCM command
  description 466
  example 465
  syntax 465
UCO command
  syntax 467
UCS command 468
UCSJ command
  constraints 469
  description 469
  example 469
UCU command
  constraints 472
  description 471
  example 471
  syntax 471
UFS command
  syntax 473
UGRP command
  constraints 474
  description 474
  syntax 474
UI command
  constraints 478
  description 478
  example 475
  syntax 475
UIA command
  constraints 482
  description 481
  example 479
  syntax 479
UINS command
  syntax 483
ULCA command
  constraints 485
  description 485
  example 484
  syntax 484
ULCK command
  constraints 486

description 486
example 486
syntax 486
UNC command
syntax 487
UNC, see Universal Naming Convention
UNDO command 488
Universal Naming Convention 28
unlocking, projects 486
unlockstream (DM command-line client) 627
UNN command
syntax 490
unrelating
baselines from requests 526
design parts 514
item from item 531
item from part 532
item from requests 529
part from requests 534
requests from change document 528
unsupported commands, on z/OS 12
UNWO command
syntax 491
UOS command
syntax 492
UP command
constraints 493
example 493
syntax 493
UPA command
constraints 495
description 494
example 494
syntax 494
update
work area 496
Update Code Metrics 465
UPDATE command
examples 497
UPDATE command description 496
UPDATE command syntax 496
updating
baseline attributes 459
build area 457
design part PCS 493
item,revise item 475
part attributes 494
part numbers 510
request 461
user attributes 517
user pending lists 552
updating metadata 549
upgrade process, start 55
UPLOAD command
description 509
examples 505
parameters 156
syntax 505
UPNO command
constraints 510
example 510
syntax 510
UPROD 511
UPROJ command
syntax 512
UREG command
description 513
syntax 513
URP command
constraints 514
example 514
syntax 514
URSD command
syntax 515
user commands
assign roles (AUR) 56
authorize access to node (AUTH) 60
define roles (DUR) 212
update attributes (UUA) 517
user roles, assigning 56
user-defined reports 412
USUB command
description 516
example 516
syntax 516
utilities
case translation 543
dm_auto_action 551
dm_full_mail 552
dm_incremental_mail 552
dmpasswd 554
execution authority 543
introduction 542
prcs 556
psccs 565
wild card characters 543
UUA command
constraints 517
syntax 517
UWA command
constraints 520
description 520
example 518
syntax 518

V

variant
structures, creating 140
version commands
define branch (DVB) 214

remove branch (RMVB) 394
setting, branch flags (SVBF) 444
VLSJ command
description 521
example 521

W

wild card characters, in standalone utilities 543
Windows
UNC 28
work area
update 496
WRC command
constraints 522
description 522, 523
example 522
syntax 522

X

XABC command
example 523
XAWS command
constraints 524
description 524
examples 524
syntax 523, 524
XBBL command
constraints 525
description 525
example 525
syntax 525
XBCD command
constraints 526
example 526
syntax 526
XBWS command
constraints 527
description 527
example 527
syntax 527
XCCD command
constraints 528
example 528
syntax 528
XCMD, see CMD
XICD command
constraints 530
example 529
syntax 529
XII command
description 531
example 531
syntax 531
XIP command
constraints 533
example 532
syntax 532
XPCD command
constraints 534
example 534
syntax 534
XRCD command
constraints 536
description 535
example 535
XREG command
description 537
syntax 537
XWCD command
constraints 538
description 538
example 538
syntax 538
XWWS command
constraints 539
description 539
example 539
syntax 539

Z

z/OS
project file names, specifying 28
unsupported commands 12