MERANT™

Edition 5.1

PVCS® Dimensions™

Command-Line Reference Guide

Table of Contents

Welcome to Dimensions 9
Typographical Conventions. 10
Dimensions Documentation Set 11
Using the Online Manuals. 11
Dimensions I-Net GUI. 13
PC Client GUI 13
Project/File Merge GUI. 13
Motif Client GUI. 13

1 Use of Command Mode 15
Using Quoted Strings in Dimensions Commands. 21
Multi-Value and Multi-Line Attributes. 22
Spaces in Filenames 29
Windows UNC Paths 29
Specifying a Workset Filename in
OS/390 Item Operations 30

2 Command Reference 31
ABL - Action Baseline or Items 32
AC - Action Change Document. 34
ADF - Assign (Item) Data Formats. 36
AI - Action Item 38
AIWS - Add Item Revision to Workset 40
APNO - Allocate Part Numbers 42
AUR - Assign User Roles 44
AUTH - Authorize Access to Node 47

BC - Browse or Print Change Document 49
BI - Browse Item 51
CBL - Create Baseline 53
CC - Create Change Document 56
CI - Create Item. 59
CIU - Cancel Item Update. 66
CMB - Create Merged Baseline 68
CMP - Compare Structures or Baselines 72
CP - Create Design Part 74
CPCS - Check PCS 76
CPV - Create Design Part Variant. 78
CRB - Create Revised Baseline 80
CVS - Create Variant Structure. 86
CWSD - Create Workset Directory 89
DBL - Delete Baseline 90
DCH - Delete Change Document. 91
DDF - Define (Item) Data Formats. 92
DEI - Define Environment Items 95
DI - Delete Item 98
DIFF - Compare Items 100
DIR - Define Item Relations 104
DLGC - Delegate Change Document 106
DLGI - Delegate Item 108
DNP - Define New Product. 111

DPL - Define Product Libraries 114
DPV - Delete Design Part Variant 118
DREL - Delete Release 119
DUR - Define User Roles 120
DVB - Define Version Branch 122
DWP - Delete Whole Product 124
DWS - Define New Workset 125
DWSD - Delete Workset Directory 129
EDI - Edit Item 130
EI - Extract (Check Out) Item for Update 135
EXIT - End Dimensions Execution 140
FI - Fetch (Get) Item 141
FIF - Find Item File 146
GREP – Search and Replace 148
INFOR - Information (display README file) 151
IP - Item Processing 152
LCK - Lock Workset 159
LWS - List Worksets 160
LWSD - List Workset Directories 161
MCPC - Move Change Document To Primary (Main) Catalog . 163
MCSC - Move Change Document To Secondary Catalog 164
MDR - Move Design Part Relationship 165
MI - Merge Items 166
MIP - Move Item to another Part 169

MIT - Move (Change) Item Type 171
MVC - Move Change Document 173
MWS - Merge Worksets 177
MWSD - Move Workset Directory 179
QIIP - Query Items with IPDs 180
QIMO - Query Item Made-of List. 181
RAA - Generate Documents Index. 183
RAH - Generate PCS Requirements List. 184
RCCD - Relate Change Documents to Change Document . . . 185
RCI - Report Current Items. 187
RCP - Report Current Parts. 190
RDS - Report Design Structure. 193
REL - Release. 197
RHA - Generate Design Part Parts List. 200
RHAB - Generate Parts List for a Baseline. 201
RI - Return (Check In) Item. 202
RICD - Relate Item to Change Documents 207
RII - Relate Item to Item. 209
RIP - Relate Item to Part. 211
RIR - Remove Item Relation Definition 213
RIWS - Remove Item Revision from Workset 214
RMDF - Remove (Item) Data Formats 216
RMVB - Remove Version Branch 217
RP - Relate Design Part. 218

RPCD - Relate Part to Change Documents 219
RPCP - Report Product Control Plan (Process Model) 220
RPNO - Report Part Numbers 222
RPT - Report Change Documents 223
RPT - Baseline Detail Report 229
RUR - Run User-Defined Report 231
RWS - Remove Workset 234
SCWS - Set Current Workset 235
SDF - Set Data Format Flags 238
SET - Set DIR, PRINTER or OVERWRITE Environment. 240
SI - Suspend Item 242
SPV - Suspend Design Part Variant 244
SVBF - Set Version Branch Flags 245
SWF - Set Workset Filename 247
SWS - Set Workset Attributes 249
UBLA - Update Baseline Attributes 252
UC - Update Change Document 254
UI - Update Item 256
UIA - Update Item Attributes 261
ULCK - Unlock Workset 265
UP - Update Design Part PCS. 266
UPA - Update Part Attributes 268
UPNO - Update Part Numbers. 270
URP - Unrelate Design Part 272

XCCD - Unrelate Change Documents from Change Document273

XCMD - Execute Dimensions Command File 275

XICD - Unrelate Item from Change Documents 276

XII - Unrelate Item from Item 278

XIP - Unrelate Item from Part 280

XPCD - Unrelate Part from Change Documents 282

3 Standalone Dimensions Utilities 285

Introduction 287

General Information 289

Case Translation 289

Wild Card Characters 289

Execution Authority: Change-Manager or Tool-Manager 290

Action Change Documents by Date or Attribute Value: pcms_auto_action 291

Edit Detailed Description: pcms_cm_edit 293

Update Users' Pending Lists: pcms_cm_pend 294

Send Reminders of Pending Lists: pcms_full_mail & pcms_incremental_mail. 296

Automatic Job Triggering: Using crontab. 297

Update Users' Pending Items Lists: pcms_item_pend. 299

Rename a Design Part within the Same Product: pcms_rename_part. 300

Rename an Existing Product: pcms_rename_product 301

Set Dimensions Workset Permissions: pcms_workset_perm . 302

PRCS - RCS-like Front End for Dimensions 303

PSCCS - SCCS-like Front End to Dimensions 321

Welcome to Dimensions

Thank you for choosing MERANT™ PVCS® Dimensions™, a powerful process management and change control system that will revolutionize the way you develop software. Dimensions helps you organize, manage, and protect your software development projects on every level—from storing and tracking changes to individual files, to managing and monitoring an entire development cycle.

Purpose of this manual

This manual is a guide to Dimensions Command Mode Applications and Dimensions standalone utilities (the latter was formerly documented in the *Motif Client User's Guide*). The manual is intended for users with a knowledge both of Dimensions and a familiarity with its command line interface.

This manual covers all the Dimensions functions (except as detailed below) which are available in Command Mode, specifying and describing the parameters and qualifiers used by each one.

There are just two sets of functions which are *not* included here:

- The functions in the DBA – Database Administration subsystem available in command mode are detailed in the separate publication: *Database Administrator's Guide*.
- The command-mode functions for Dimensions Archive and Retrieval are detailed in the separate publication: *ART User's Guide*.

For more information

Refer to the *PVCS® Dimensions™ Getting Started Guide* for a description of the Dimensions Documentation Set, a summary of the ways to work with Dimensions, and instructions for accessing the Online Help.

Edition status

This is Edition *5.1* of the *PVCS® Dimensions™ Command-Line Reference Guide.* The information in this edition applies to *Release 7.1 of PVCS® Dimensions™* or later. This edition supersedes earlier editions of this manual.

Typographical Conventions

The following typographical conventions are used in the online manuals and online help. 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 Dimensions components or the host operating system.

Convention	**Explanation**
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.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
`monospace bold`	Indicates the results of an executed command.
vertical rule |	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.

Convention	Explanation
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.

Ordering Hard-Copy Manuals

As part of your Dimensions license agreement, you may print and distribute as many copies of the PVCS Dimensions manuals as needed.

If you do not want to print each of these online manuals, you can order hard-copy versions from MERANT. To order, please contact your sales representative for assistance.

Contacting Technical Support

MERANT provides technical support for all registered users of this product, including limited installation support for the first 30 days. If you need support after that time, contact MERANT using one of the methods listed in the installation guides, the *Getting Started Guide,* or the Online Help.

Technical support is available 24 hours a day, 7 days a week, with language-specific support available during local business hours. For all other hours, technical support is provided in English.

Support via the web, E-mail, and telephone

SupportNet Customers can report problems and ask questions on the SupportNet web page: `http://support.merant.com/`

To submit an issue, click on the **Report a Problem** link and follow the instructions. You can also submit issues via E-mail or phone. Refer to the installation guides, *Getting Started Guide*, or Online help for a list of contact numbers, including numbers to call for local language support.

The SupportNet Web site contains up-to-date technical support information. Our SupportNet Community shares information via the Web, automatic E-mail notification, newsgroups, and regional user groups.

SupportNet Online is our global service network that provides access to valuable tools and information for an online community for users. SupportNet Online also includes a KnowledgeBase, which contains how-to information and allows you to search on keywords for technical bulletins. You can also download fix releases for your PVCS products.

1 Use of Command Mode

In this Chapter

For this section…	See page…
Introduction	14
Command Mode Syntax	16
Note on the Selection of Item Revisions by Default	23
Updating the Content of an Item without Changing the Item Revision	24
Note on Workset Assignment	26
Change Document Attributes	26
Operating System Differences	27

Introduction

Command-mode (available for the majority, but not all, Dimensions functions) is an efficient alternative to Dimensions GUI-based clients, **provided that you are familiar with both Dimensions and the product to be processed**. Command mode is particularly suited for situations where you wish to perform unattended bulk batch operations (it should be noted that in batch mode Dimensions sends you e-mail confirming the submission and completion of each operation).

Command mode can be used in any of several ways. In all cases you must first set up your Dimensions environment either by explicitly setting your *PCMSDB* operating system environment variable or via the appropriate GUI login dialog. See also "Using Quoted Strings in Dimensions Commands" on page 19.

Several of the mechanisms detailed below are not supported by Dimensions Agent for OS/390[1]. Refer to the *Dimensions Agent for OS/390 User's Guide* for further details.

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

  NOTE Not supported by the Dimensions Agent for OS/390.

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

  NOTE Not supported by the Dimensions Agent for OS/390.

1. The term OS/390 in this manual covers both the OS/390 operating system and the z/OS operating system.

- 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 XCMD command. The file may contain any number of commands to be processed sequentially in a single batch job. However, the interactive functions (BI, EDI, and some uses of BC and UC) cannot be included.

  NOTE Not supported by the Dimensions Agent for OS/390.

- A single command may be entered at the Execute Command window in Dimensions PC Client's Run interface.
- A single command may be preceded by CMDCLIENT and entered at the operating system prompt.

  NOTE Not supported by the Dimensions Agent for OS/390.

- A single command may be entered at the *`CMDCLIENT`* prompt that results from typing CMDCLIENT at the Windows operating system prompt and filling in the login dialog. This login dialog is the same as that used for PC Client (see *PC Client User's Guide* for details). This mechanism enables you to execute commands on *remote* Dimensions servers if desired.

The following topics are covered in this chapter:

- Command mode syntax.
- Using quoted strings in Dimensions commands.
- Compound fields in Dimensions commands.
- Note on the selection of item revisions by default.
- Note on workset assignment.
- Change document attributes.
- Operating System Differences.

This is then followed by:

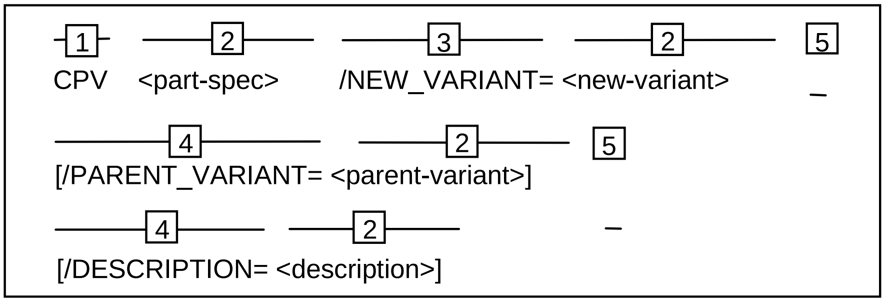
- Chapter 2, "Command Reference" which covers most of the Dimensions commands, specifying and describing the parameters and qualifiers used by each one. Refer to "Purpose of this manual" on page 9 for details on the two sets of functions which are **not** included in this guide.
- Chapter 3, "Standalone Dimensions Utilities" which identifies various miscellaneous Dimensions standalone utilities.

Command Mode Syntax

A command consists of a Dimensions function mnemonic, followed by parameters and qualifiers. A typical Dimensions command looks like this:

```
CPV :”:"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** (Create Design Part Variant):


1
CPV
2
<part-spec>
3
/NEW_VARIANT= <new-variant>
2
5
4
[/PARENT_VARIANT= <parent-variant>]
2
5
4
[/DESCRIPTION= <description>]
2

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

$\boxed{1}$ The *function mnemonic* identifies which Dimensions function is to be performed.

$\boxed{2}$ A parameter is indicated by lower-case letters enclosed in angle brackets. This shows where a variable value is to be substituted. Parameters which 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 21.

An *ellipsis (...)* indicates that a list of any number of parameters may be specified, separated by commas and enclosed in parentheses, for example:

```
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
```

If there is only one parameter in the list, the parentheses are not essential, 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.

$\boxed{3}$ A *required qualifier* is coded as shown. It always begins with an oblique (/) and usually ends with 'equals to' (=), 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 indicated by being enclosed in square brackets ([ ]), and **may be omitted in certain circumstances** (as detailed in this guide). 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 (e.g. /NEW_VAR= in the example above). Options are shown on consecutive lines **which have the same indentation**, with an underscored **<u>or</u>** 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 that and the end of the line.

Exceptions:

- *UNIX systems* (**only** 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* (**only** 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 characters outside the standard set, in the substitution for a parameter variable, then the variable value, or the part of it which contains the non-standard character(s), must be enclosed in double-quotation characters (" "), for example:

```
"RELEASE MANAGEMENT"
```

But there are additional conventions which you must follow if you want to enter a Dimensions command, which includes a quoted string, at the operating system prompt:

At the Windows System Prompt

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

At the 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 guide. **You must understand implicitly that it is required whenever the command is used in this way**.

The Escape Character

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 might otherwise be interpreted as part of the syntax of the command. Such characters include:

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

The default escape character is **@**, but the setting of the Dimensions symbol PCMS_ESCAPE_CHAR may be used to specify any alternative as the escape character. The Windows command line interface escape character is the 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 *cmdclient* from the Windows operating system prompt would be:

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

An example how to use the "@n" syntax for escaping newlines is given in the discussion of Multi-Line Attributes in the next section.

Multi-Value and Multi-Line Attributes

A multi-value 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 multi-line 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

Notes

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

- 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>*.

In the syntax diagrams there are five parameters, *<workset-spec>*, *<part-spec>*, *<item-spec>*, *<baseline-spec>* and *<release-spec>*, where the required substitution is a set of values or fields in a specific syntax format. This syntax is as follows:

<workset-spec>

This fully identifies a specific workset and has the following syntax:

```
<product-id>:<workset-id>
```

<part-spec>

This fully identifies a specific design part and has the following syntax:

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

<item-spec>

This fully 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>

This fully identifies a specific baseline and has the following syntax:

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

<release-spec>

This fully 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
```

Note on the Selection of Item Revisions by Default

When *<revision>* is not specified, it defaults to the *latest revision in the user's current workset*. Note that this **may not** necessarily be the latest revision of the item in the database, since only the revisions in the user's *workset* are considered. *Latest revision* means that version file content *has most recently been created or updated* – which **might not** be the one with the highest entry in the revision field. This algorithm does not take into account the branch names or whether the branches are locked or owned remotely (via replication). For items which had previously been checked out, the time of creation/update is to be regarded as the time of the check in (RI function), *not* the earlier time of the check out (EI function).

You should bear in mind that criteria other than the above are *not* used in selecting a revision by default. The best way to make this point clear is to consider an example. Suppose that Revision 2 of an item is the latest revision (by the above criteria), and also that Revision 1 of this item has been related to Change Document A_B_1 as *Affected*. Now an Extract (Check out) Item (EI) command, to create Revision 3 as *in-Response-to* Change Document A_B_1, *must specify Revision 1* in the `<item-spec>`, if Revision 3 is to start off as identical to Revision 1. Otherwise, by default, Revision 3 would start off as identical to Revision 2, despite the fact that it was not Revision 2 which was cited as *Affected*.

Updating the Content of an Item without Changing the Item Revision

If the PRODUCT-MANAGER has setup the process model (control plan) so as to allow you to update the *file content* of an item revision residing at the initial lifecycle state *without* having to change the item revision, you should bear the following in mind if you do such an edit.

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

Note on Workset Assignment

Each user must have a (mandatory) default workset, assigned as follows:

- When the Tool Manager initially registers the Dimensions user concerned, the user is automatically assigned to the default global workset called *`$GENERIC:$GLOBAL`*. This workset assignment enables the user to reference any item revision in any product in the base database to which his/her PCMSDB symbol points.
- Subsequently, the user can then use the command SCWS (Set Current Workset) or the `/WORKSET` qualifier found on certain commands to reference a specific workset e.g. 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 workset as described below:

  - `/DEFAULT` to specify that the current workset 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 workset assigned by SCWS is for the duration of the present process/session only, and that the current workset 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 workset to be used for the *duration of the command* concerned.

Each workset, when opened, must have a (mandatory) top level "workset root directory" assigned to it. This "workset root directory" defines a point in the directory hierarchy structure below which (or relative to which) the workset filename 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 workset filename as used in commands such as get or build consists of the relative directory path from the workset root directory `<directory-spec>` and the filename from `<ws_filename>` concatenated together.

Change Document Attributes

Change Document Attribute 1 must always be defined in the process model. It must be given the variable name TITLE and be declared as single-valued. Its length must be less than or equal to 80 characters.

Change Document 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 change documents being actioned to new states.

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.

The maximum length of a default value for an attribute is restricted to 240 characters.

There is no facility to specify a list of values as default value for a multiple-valued attribute. However, a single value attribute can be specified as the default value for such an attribute.

Operating System Differences

Spaces in Filenames

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

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 e.g. workset root directories or getting items. As with all directory specifications, if a workset root directory is set as a UNC path and that workset is opened by the same user on UNIX, the directory specification will not be recognized.

Specifying a Workset Filename in OS/390 Item Operations

When running Dimensions item operations from an OS/390 platform, you need to specify a 'backslash' character (\) in place of any parenthesis within a workset filename. For example, if the workset filename is *`TEST.COBOL(STAFF)`*, to *`get`* the item using the workset filename you would need a command such as:

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

2 Command Reference

This chapter contains an alphabetic listing of commands.

ABL - Action Baseline or Items

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

Example

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

- `<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 which is actioned. See Description on the page 31 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 31 for details.
- `/COMMENT= <text>`

  is an optional user-defined action-comment.

  Dimensions enters a default comment if this is omitted.

Description

This function 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

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

AC - Action Change Document

```
<chdoc-id>
[   /STATUS=<status>]
    [ /ACTION_CHECK
or  /CLOSURE_CHECK]
```

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
```

- `<chdoc-id>`

  is the identity of the change document to be actioned or checked.

- `/STATUS=<status>`

  specifies the new status to be given to the item revision (provided `ACTION_CHECK` is omitted). Unless the user has the role of CHANGE-MANAGER (see note below), the new status must be one reachable by a single lifecycle transition from the current status.

  If omitted, Dimensions will action the change document to its next state in the normal lifecycle, or if the change document is held, save it (which places it at its initial lifecycle state).

NOTE Saving results in a changed value for *<chdoc-id>*, but *$LAST* can be used to refer to it in subsequent commands in an XCMD file. (See note on CC command-mode function on page 54.)

CAUTION! *<status>* **cannot be omitted** if the current status is a state not in the normal lifecycle.

- `/ACTION_CHECK`

  indicates that Dimensions is to check whether the specified change document can be either actioned to the state specified by the `/STATUS` qualifier or the next normal state if `/STATUS` is not specified, while conforming to the current rules. This qualifier must not be used with the `/CLOSURE_CHECK` qualifier.

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

- `/CLOSURE_CHECK`

  indicates that Dimensions is to check whether the specified change document can be actioned to the final state in its normal lifecycle, while conforming to the current rules. This qualifier must not be used with the `/STATUS` qualifier or the `/ACTION_CHECK` qualifier.

NOTE A user with the role of CHANGE-MANAGER may action any change document to any valid state in its lifecycle. This includes the possibility of re-opening a change document which has been closed or rejected (i.e. has reached a final state).

Constraints

To action a change document, you must have a role required to action the change document to a new state.

To be able to select any lifecycle state from any stage of the lifecycle, you need CHANGE-MANAGER role or have the role on the lifecycle transition if that transition exists.

ADF - Assign (Item) Data Formats

```
<product-id>
/ITEM_TYPE=<item-type>]
/FORMAT_LIST=(fomat1,format2,format3,…)
```

Example

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

- `<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 function assigns item formats to particular item types. The item formats must have been previously defined using the Define (Item) Data Format (DDF) command. Once these formats are assigned to an item type, the choice of one these formats is compulsory when creating files 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.

This function is available *only* in Command Mode.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* for the product.

AI - Action Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /STATUS=<status>]
[ /COMMENT= <text>]
[ /WORKSET=<workset-spec>]
```

Example

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

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

<item-id>	may be omitted if <*filename*> is specified.
<variant>	may be omitted if only one exists.
<revision>	may be omitted if <*filename*> is specified.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *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>`

  is an optional user-defined action-comment.

  Dimensions enters a default comment if this is omitted.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

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

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.

Constraints

This command can be run if the current item revision is in your pending list or you have the role of PRODUCT MANAGER.

AIWS - Add Item Revision to Workset

```
<item-spec>
[ /FILENAME=<filename>]
/WORKSET=<workset-spec>
```

Example

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

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

<item-id>	may be omitted if *<filename>* is specified.
<variant>	may be omitted if only one exists.
<revision>	defaults to the latest revision in the workset specified by `/WORKSET`. If `/WORKSET` is unspecified, the user's default workset will be assumed.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *src/hello.c*, *hello.c* or *src/build/hello.c*.

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

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

Description

The item selected by the *`<item-spec> [/filename=<filename>]`* parameters from the current workset will be added to the workset *`/WORKSET=<workset-spec>`*.

Constraints

Normally this command can be run only by a user with the role of *`PRODUCT-MANAGER`* or *`WORKSET-MANAGER`* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

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"
```

- `<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

This command can be run only by a user who has the role of *`PARTS-CONTROLLER`* or by a user who has the role of *`PRODUCT-MANAGER`*.

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

AUR - Assign User Roles

```
<username>
/ROLE=<role>
[ /TYPE=<assignment-type>]
/PART=<product-id>:<part-id>.<variant>
[ /CAPABILITY=<capability>]
[ /ADD]
or  [ /DELETE ]
[ /WORKSET=<workset-spec>]
```

Example

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

- `<username>`

  is the login username of the Dimensions user to whom the role is (to be) assigned.

- `/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 The variant field may be omitted in order to apply the role to **all** variants of the design part.

- `/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 change document or item-spec e.g. so that they can add comments (called Action Descriptions in change documents). 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 as having the main responsibility for that role on the change document 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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  and is optional. If specified, the role assignment will apply to that particular workset. If unspecified, the role assignment will apply to all worksets, unless the role is *WORKSET-MANAGER* (in which case it is necessary to assign a specific workset for that role-assignment to be effective).

Constraints

This command can be run only by a user with the role of *PRODUCT-MANAGER*.

AUTH - Authorize Access to Node

```
[/NETWORK_NODE=<node-name>
[/USER=<userid>
[/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 the item libraries on node *MYNODE* for the user *MICKEY*, with password *MOUSE*.

- `/NETWORK_NODE=<node-name>`

  specifies the name of the node where your items are stored.

- `/USER=<userid>`

  is your User ID for the node specified.

- `/PASSWORD=<password>`

  is the password associated with the User Id specified

- `/NEW_PASSWORD=<new-password>`

  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.

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 item library nodes currently available, enter the command with *no* parameters
  - To obtain a list of authenticated users for one item library 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 to the item library on a node. Enter the command with the parameters *`/NETWORK_NODE`*, *`/USER`* and *`/PASSWOR`*`D`. You can also change the password at the same time, by specifying the parameter *`/NEW_PASSWORD`*.

BC - Browse or Print Change Document

```
<chdoc-id>
[ /FILENAME=<user-filename>]
[ /PRINT ]
[ /ACTION_NO=<number>]
```

Example

```
BC PROD_DR_25 /ACTION=2
```

- `<chdoc-id>`

  is the identity of the change document 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 change document will be gotten. If this qualifier is used, interactive browsing is not invoked.

  NOTE This parameter must be specified if you wish to run BC from the Dimensions Agent for OS/390 or the CMDCLIENT interface.

- `/PRINT`

  specifies that the change document is to be printed. If this qualifier is used, interactive browsing is not invoked.

- `/ACTION_NO=<number>`

  specifies that the change document 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 change document is shown.

Constraints

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

BI - Browse Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /BASELINE=<baseline-spec>]
[ /WORKSET=<workset-spec>]
```

NOTE BI cannot be run from the Dimensions Agent for OS/390 or the CMDCLIENT interface.

Example

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

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

`<item-id>`	may be omitted if *`<filename>`* 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 Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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 omitted, the specified or default `<revision>` (as described above) is browsed.

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

  ```
  <product-id>:<workset-id>
  ```

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

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.

CBL - Create Baseline

```
<baseline-spec>
/PART=<part-spec>
[ /TEMPLATE_ID=<template-id>]
[ /TYPE=<baseline-type>]
[ /WORKSET=<workset-spec>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /LEVEL=<integer> ]
```

Example

```
CBL PROD:"R M VERSION 2 FOR HP" -
    /TEMPLATE=SOURCE+DOC -
    /PART= PROD:"RELEASE MANAGEMENT".AAAB
```

■ `<baseline-spec>` comprises:
`<product-id>:<baseline-id>`

<baseline-id>	is the identity to be given to the baseline.

■ `/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.

■ `/TEMPLATE=<template-id>` is the identity of the baseline-template.

This must be:

- specified when creating a release-baseline
- omitted when creating a design baseline.

- `/TYPE=<baseline-type>`

  specifies the type of baseline being created.

  If omitted, the default type is RELEASE if a *`<template-id>`* has been specified. (If */TYPE* is not specified, then by default, a design baseline is created, which is simply a snapshot of the current stage of product development, and is not therefore expected to need an approval lifecycle.)

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  and is optional. If specified, only the items in the workset are considered for baselining; otherwise, only the items in the user's current workset are considered. See also, Dimensions LCK command (on page 157) concerning comment about locking the workset when baselining.

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

`<attrN>`	is the Variable Name defined for one of the 220 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.

- `/LEVEL=<integer>`

  optionally you may elect to restrict the baseline to a given number of levels in the design tree structure.

  The default of 0 (zero) signifies all levels below the design part selected by */PART*. For example, **1** signifies that items related to the selected design part only will be processed.

Constraints

Generally, to create a baseline you must have one of the roles for its specified top design part that are required to action it from its initial lifecycle state to a new state. However, if a user with the role *`PRODUCT-MANAGER`* has assigned the top design part role *`$ORIGINATOR`* to the first transition in the lifecycle for this baseline type, any Dimensions user can create a baseline of this type provided they have a role (any will suffice) on the top design part on which the baseline is being created.

CC - Create Change Document

```
<product-id>
<chdoc-type>
[ /BASED_ON=<chdoc-id>]
[ /DESCRIPTION=<desc-file>]
[ /AFFECTED_PARTS=(<part-spec>,<part-spec>, ... ) ]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /RELATIONSHIP=<rel_name>]
[ /[NO]HOLD ]
```

Example

```
CC PROD DR-
    /DESC=qrel_subdir.desc-
    /AFFECT= PROD:”RELEASE MANAGEMENT”.AAAA -
    /ATTRIB=(TITLE=”QREL Subdir problem”,SEVERITY=3)Sub
```

- `<product-id>`

  specifies the product which is to own the new change document.

- `<chdoc-type>`

  specifies the type of change document to be created.

- `/BASED_ON=<chdoc-id>`

  is used to base (i.e. prime) the creation of the new change document on the attributes of the change document given by the identity *<ch_doc_id>*.

  If omitted, the new change document’s data is derived solely from whatever other parameters are specified in this command.

- `/DESCRIPTION=<desc-file>`

  specifies a file containing the text body to be used as the detailed description of the change document.

- `/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.

  This is used to specify one or more design parts to be related to the new change document.

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

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

`<attrN>`	is the Variable Name defined for one of the 220 user-defined attributes for change documents, which has also been declared as usable for this *`<product-id>`* and *`<chdoc-type>`*.
`<valueN>`	is the substitution value to be given to this attribute.

- `/RELATIONSHIP=<rel_name>`

  is valid only if the qualifier *`/BASED_ON`* is used. It specifies the relationship type between the newly created change document and the base change document. The relationship is a bi-directional link with the new change document as child and the base change document as parent.

- `/HOLD`

  specifies that the new change document is to be placed on the user's Held List for possible modification before it is entered into the system.

The default is `/NOHOLD` meaning that the new change document is saved i.e. it is immediately entered into the system.

NOTE The *`<chdoc-id>`* generated by CC may be referenced as *`$LAST`* by a subsequent command within an XCMD command-file.

UNIX users only: This must be preceded by a backslash, thus: *`\$LAST`*.

Constraints

This command can be run (by default) by all users. However, a user with the role of *`PRODUCT-MANAGER`* can set a Process Model flag to insist that you must have a role (any role will suffice) on the product concerned before you can create change documents.

CI - Create Item

```
<item-spec>
/PART=<part-spec>
[/FILENAME=<filename>]
[ /WS_FILENAME=<ws_filename>]
[ /COMMENT=<comment text>]
[ /FORMAT=<format>]
[ /USER_FILENAME=<user-filename>]
[ /KEEP ]
[ /DESCRIPTION=<description>]
[ /EXTRA_VARIANTS=(<var1>,<var2>, ... ) ]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /CHANGE_DOC_IDS=(<cd1>,<cd2>, ... ) ]
[ /STATUS=<status>]
[ /WORKSET=<workset-spec>]
[ /CODEPAGE=<code-page>| DEFAULT]
```

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"
```

- `<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 Modeling User's Guide*), then the Item-id must **not** be specified.

- `/PART=<part-spec>`

  identifies the design part which will own the item.

  It comprises: `<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.

- `/FILENAME=<filename>`

  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>`*). *`<filename>`* **may** include subdirectory name(s) but must **not** specify an absolute path; thus:

• **UNIX** ⇒	*`<filename>`* **must not** begin with **/** (oblique).
• **Windows** ⇒	*`<filename>`* **must not** begin with **** (backslash) or <drive:>.

  If the filename is not specified, then a new name is derived from the workset filename qualifier.

- `/WS_FILENAME=<ws_filename>`

  identifies the relative pathname (directory plus filename) of the file to be used when the item created here is subsequently checked out or gotten as an item from the current workset.

`<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 file formats to particular item types. Uses for such a list include:

  - Validation on item creation, specifying file types for the Dimensions PC Client application and specifying MIME types for the Dimensions Internet Client (Dimensions I-Net).
  - Dimensions Build. 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 *`<filename>`* then *`<format>`* will default to that and does not need to be explicitly specified (but if *`<filename>`* 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>`

  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 220 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=(<cd1>,<cd2>, … )`

`<cdN>`	identifies a change document to which the new item is to be related *in-Response-to.*

- `/STATUS=<status>`

  specifies the status of the created item.

  NOTE 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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

- `/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 OS/390 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.

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.

CIU - Cancel Item Update

```
<item-spec>
[ /FILENAME=<filename>]
[ /WORKSET=<workset-spec>]
```

Example

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

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

`<item-id>`	may be omitted if <*`filename`*> 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.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory of the file to be used when the item is checked out from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  A workset 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 workset will be used.

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 role of *`PRODUCT-MANAGER`*.

CMB - Create Merged Baseline

```
<new-baseline-spec>
/PART=<part-spec>
[ /TYPE=<baseline-type>]
/BASELINE_LIST=(<baseline1>,<baseline2>, ... )
 [ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
```

Example

```
CMB PROD:"MERGED VER 2A FOR HP" -
    /PART= PROD:"RELEASE MANAGEMENT".AAAB -
    /BASE=("QUERY RELEASE MOD 2A", -
    "R M VERSION 2 FOR HP")
```

- `<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 220 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.

Description

1 This function 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 function 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 mechanically according to the specification above; and it is entirely the Dimensions user's responsibility to specify baseline-lists which 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.

CMP - Compare Structures or Baselines

```
<file-name>
[   /PART = <part-spec>
    or       /BASELINE = <baseline-spec>]
[<file-name>
             [     /PART1 = <part-spec>
             or    /BASELINE1 = <baseline-spec>]
    ]
```

NOTE CMP cannot be run from the Dimensions Agent for OS/390.

Example

```
CMP * /BASELINE= PROD:"R M VERSION 2 FOR HP" -
   hprm3_01.export -
   /PART1= PROD:"RELEASE MANAGEMENT".AAAB
```

- `<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.

- `/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.

CP - Create Design Part

```
<part-spec>
/CATEGORY= <category-name>
/PARENT_PART= <parent-part-spec>[1]
/DESCRIPTION= <description>
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
```

Example

```
CP PROD:"RELEASE MANAGEMENT" /PARENT=PROD:PROD -
/CAT=SUBSYSTEM /DESC="Release Support Sun Version"
```

- `<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.

- `/PARENT_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.

1. The parameter */FATHER_PART=<father-part-spec>* will also continue to be supported for backward compatibility.

- `/DESCRIPTION=<description>`

  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 220 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

This command can be run only by user with the role of *PRODUCT-MANAGER* or by a users appropriately assigned the role of *PCMS-PART-MANAGER*.

CPCS - Check PCS

```
<part-spec>
```

NOTE CPCS cannot be run from the Dimensions Agent for OS/390.

Example

```
CPCS PROD:"RELEASE MANAGEMENT".AAAA
```

- `<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 automatically writes a file *`check_pcs.out`* in your current directory. The file contains a report which examines all non-suspended parts in the current design structure under the selected part/variant and indicates whether the PCS of a part should be changed because the PCS of a part is different in an existing release baseline (i.e. baselines with a template) The report also indicates whether the PCS may need to be changed because the part's related items have changed.

Constraints

This command can be run only by users who have a role on the design part, for which only the current PCS may be selected. This design part must not be suspended and must be included in at least one baseline.

CPV - Create Design Part Variant

```
<part-spec>
/NEW_VARIANT=<new-variant>
[ /PARENT_VARIANT=<parent-variant>]1
[ /DESCRIPTION= <description>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
```

Example

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

- `<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**.**

- `/PARENT_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.

1. The parameter */FATHER_VARIANT=<father-variant>* will also continue to be supported for backward compatibility.

- `/DESCRIPTION=<description>`

  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 220 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

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* or by a user appropriately assigned the role of *`PCMS-PART-MANAGER`*.

CRB - Create Revised Baseline

```
<new-baseline-spec>
/BASELINE1=<existing-baseline-spec>
[ /TYPE=<baseline-type>]
[ /UPDATE_CHANGE_DOC_IDS=(<ucd1>,<ucd2>, ... ) ]
[ /REMOVE_CHANGE_DOC_IDS=(<rcd1>,<rcd2>, ... )]
[ /CANCEL_TRAVERSE ]
[ /WORKSET=<workset-spec>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
```

Example

```
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
```

- `<new-baseline-spec>` comprises:
  `<product-id>:<baseline-id>`

<baseline-id>	is the identity to be given to the revised baseline being created.

- `/BASELINE1=<existing-baseline-spec>` comprises:
  `<product-id>:<baseline-id>`

<product-id>	must be the same as that for `<new-baseline-spec>`.
<baseline-id>	is the identity of the existing release baseline to be used to create the revised baseline.

- `[ /TYPE=<baseline-type>]`

  specifies the type of the revised baseline being created.

  If omitted, the type will be the same as that of *<existing-baseline-spec>*.

- `/UPDATE_CHANGE_DOC_IDS=(<ucd1>, ... )`

  identifies one or more change document trees (see Description below) within which there are item revisions related *in-Response-to*. Each such revision replaces the revision of the same item which was previously in the baseline. If there was no such item already in the baseline, then the in-response-to revision is now added, *provided that* it is a relevant item (i.e. falls within the scope of the baseline).

- `/REMOVE_CHANGE_DOC_IDS=(<rcd1>, ... )`

  identifies one or more change document trees (see Description below) within which there are item revisions related as *Affected*. Each such revision, if it is found in the revised baseline, it is deleted from it. Although each of the above qualifiers */UPDATE* and */REMOVE* is optional, at least one of them must be specified.

- `/CANCEL_TRAVERSE`

  indicates that, in the above processing for */UPDATE* and */REMOVE*, traversal of tree structures is not to be performed, and that only the change documents cited in the lists are to be inspected for item revisions that have been related, respectively, in-Response-to and as *Affected*.

  If omitted, the processing is of tree-structures as described above (and explained below).

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken. When an item has been selected for addition to the baseline, the workset filename will be taken from this workset if another revision is not already in the workset.

- `/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 220 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.

Description

1 This function creates a new baseline as an exact copy of an existing baseline. So not only will the new baseline have the same top design part, but *it will also have the same list of included design parts* (i.e. it will have the same scope) as the existing baseline.

NOTE A new baseline created by the CBL function, with the same top design part, might well have a different scope, because the breakdown and/or usage relationships in the design structure could now be different.)

2 The `/UPDATE` list is processed.

- A candidate list of change documents is constructed from the specified list. Each specified change document is considered as the head of a family tree of change documents with relationships to it in the Dependent class.

- The specified change document and its Dependent-related children are added to the candidate list, then any Dependent-related children of these (i.e. grandchildren) are added (if not already present), and so on until all progeny have been included. (The reason some change documents could be already in the candidate list is that they could belong to more than one family tree: i.e. they can be Dependent-related to more than one parent.) Thus the candidate list contains the whole population of change documents in all the tree structures specified by the `/UPDATE` list; unless the `/CANCEL-TRAVERSE` option was used, in which case the candidate list is just the specified list itself.

3 All the item revisions related as `In-Response-To(R)` any of the change documents in the candidate list are added to the baseline, provided that the items are within the baseline's scope, and that those revisions are neither checked out nor suspended. During this process, any revisions of those same items, which were already in the baseline, are displaced and are removed from it. Thus these `In-Response-To` revisions can be a combination:

- both of new additions to the baseline (where the item was not already included);
- and of substitutions for item revisions which were already in the baseline.

4 The `/REMOVE` list is processed. A new candidate list of change documents is constructed from this specified list in exactly the same way as was described above for the `/UPDATE` list.

5 All item revisions related as `Affected(A)` in any of the change documents in this candidate list, and which are still included in the baseline, are deleted from it.

(If any, or even all, of these revisions are not found in the baseline, they are simply ignored — no message is issued; but see

"Error Conditions" below for the occasions when all the processing, both */UPDATE* and */REMOVE*, achieves nothing.)

The principal purpose of the */REMOVE* processing is to clear out from the baseline those items which are now redundant, i.e. those for which no *In-Response-To* replacement was required. Thus the */REMOVE* qualifier is likely to be needed less frequently, because the normal process of removing superseded revisions has already been achieved in the */UPDATE* processing. It follows that, when */REMOVE* is used, it is quite likely that the change documents cited will be the same as (some of) those in the */UPDATE* list, but this is not a constraint.

NOTE The revised baseline thus created is likely to be less self-consistent than one created normally using a baseline-template. For example, the original baseline may include some item revisions by using a **BUILT* template rule. Also, the sources, from which that revision was built, could have been displaced from the revised baseline, but the original built revisions will remain in the baseline, unless they too have been specifically displaced by the */UPDATE* or */REMOVE* processing. As a reminder of this potential hazard, it is wise to use some distinctive naming convention for the baseline-ids of revised baselines.

Error Conditions

If the *In-Response-To* item revisions found in the */UPDATE* change documents include more than one revision of any one item, this conflict is reported as an error and the revised baseline is not created.

If any of the revisions, due to be deleted as a result of the */REMOVE* processing, is the same as one which was added or substituted as a result of the */UPDATE* processing, this conflict is reported as an error and the revised baseline is not created.

If, on completion of both */UPDATE* and */REMOVE* processing, it is found that no change has been effected at any point (i.e. the revised baseline would in fact be identical in content to the existing baseline), this is reported as an error, and Dimensions automatically deletes the new baseline.

Constraints

The existing baseline must be either a release baseline which was created using a template with **no** **ALL* rules, or else an earlier revised or merged baseline.

The new baseline-id must be unique within the product.

Generally, this command can be run by users who have one of the roles for the top design part in the existing baseline (which will also be the top part for the new 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 revised baseline of this type provided they have a role (any will suffice) on the top design part on which the revised baseline is being created.

CVS - Create Variant Structure

```
<part-spec>
/NEW_VARIANT=<new-variant>
[ /PARENT_VARIANT=<parent-variant>][1]
[ /DESCRIPTION= <description>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
```

Example

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

- `<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**.**

- `/PARENT_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.

1. The parameter */FATHER_VARIANT=<father-variant>* will also continue to be supported for backward compatibility.

- `/DESCRIPTION=<description>`

  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 220 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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or by a user appropriately assigned the role of *PCMS-PART-MANAGER*.

This command fails if the structure already contains a variant.

CWSD - Create Workset Directory

```
<directory-path>
[ /WORKSET=<workset-spec>]
```

Example

```
CWSD src
```

- `<directory>`

  The CWSD command creates a workset-directory <*directory*> in the database workset structure relative to the workset root directory for subsequent use in workset operations_

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken. All item revisions to be affected by the command must be within the workset.

Constraints

Normally this command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

DBL - Delete Baseline

```
<baseline-spec>
```

Example

```
DBL PROD:"R M VERSION 2 FOR HP"
```

- `<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 role of *`PRODUCT-MANAGER`*.

A baseline that has been actioned can be deleted only by a user with the role of *`PRODUCT-MANAGER`*.

NOTE A baseline cannot be deleted if it has been used to make a release or archive.

DCH - Delete Change Document

```
<chdoc-id>
```

Example

```
DCH PROD_HELD_349
```

- `<chdoc-id>`

  is the identity of the held change document to be deleted.

Constraints

Change documents which have been saved in the other types of change document lists cannot be deleted.

This command can be run only on change documents that are pending (in the Held list). They can be deleted by the user who created them or by a user with the role of *`CHANGE-MANAGER`*.

DDF - Define (Item) Data Formats

```
<format>
/DESCRIPTION=<format-description>
/CLASS=<class-no>
/MIME=<mime-type>
```

Example

```
DDF TXT /DESCRIPTION="Plain text" -
    /CLASS= 1 /MIME="TEXT/PLAIN"
```

- `<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=<mime-type>`

  specifies the Multipurpose Internet Mail Extension (MIME) type. MIME types comprise seven broad categories, with each category also having sub-categories defined by using an oblique (/) separator. The broad categories are: Application, Audio, Image, Message, Multipart, Text and Video. An example of a sub-category is *APPLICATION/WOR*D.

- `No parameters or qualifiers`

  If DDF is executed without parameters or qualifiers, it prints a list of existing data formats together with their descriptions.

Description

This function enables a user with the role of *`TOOL-MANAGER`* to define a list of valid file formats for particular item types. 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 (Item) Data Formats (ADF) command.

This function is also available from the Process Modeler, File Formats and MIME Types option.

Uses for such a list of valid file formats include:

- Validation on item creation.
- Specifying file types: All items 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 type.
- Specifying MIME types for the Dimensions Internet Client (Dimensions I-Net). I-Net will use the MIME type associated with the item's format to display the item's content within the user's browser.
- Use in Dimensions Build. 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 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*.

Existing defined item data formats can be removed by the use of the Remove (Item) Data Format Definitions (RMDF) command.

Existing item data formats can also be updated using the Update Item Attributes (UIA) command.

See also "SDF - Set Data Format Flags" command on page 238.

Constraints

This command can be run only by a user with the role of *TOOL-MANAGER* of the product.

DEI - Define Environment Items

```
<item-spec>
    [ /FILENAME=<filename>]
    [ /DELETE ]

<environment-item-spec>
    [ /ENV_FILE=<filename>]
    [ /TAG=<tag>]
```

Example

```
DEI PROD:"QUERY RELEASE"-SRC PROD:"RM INCLUDE"-H -
    /TAG= RMINC
```

- `<item-spec>`

  specifies the item which uses environment items. It comprises:
  `<product-id>:<item-id>.<variant>–<item-type>;<revision>`

<item-id>	may be omitted if *<filename>* is specified.
<variant>	for item may be omitted, if only one exists.
<revision>	for item, if omitted, the latest is used (see note in Introduction on page 23)

- `/DELETE`

  indicates that the specified environment definition(s) are existing ones which are to be deleted, not added.

- `<environment-item-spec>`

  specifies the item which is to be related as an environment item. Up to three environment items may be specified in a command. For the second and third environment items, */ENV_FILE* and */TAG* cannot be specified, and if any of these options are to be used, then */ENV_FILE2* or */ENV_FILE3* and/or */TAG2* or */TAG3* respectively must be specified instead. It comprises:

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

<item-id>	may be omitted if *<filename>* is specified.
<variant>	for environment item may be omitted. In this case, the item processor will select whichever variant is included in the configuration.
<revision>	for environment item, is ignored.

- `/FILENAME=<filename>`
  `/ENV_FILE=<filename>`

  specify the names of the workset filenames.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *src/hello.c*, *hello.c* or *src/build/hello.c*.

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

- `/TAG=<tag>`

  is an optional logical symbol by which the item may refer to this environment item.

Description

Environment Item relationships are defined for use in Dimensions IP Build. See related document *Dimensions Build User's Guide*, for more details.

Constraints

None

DI - Delete Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /WORKSET=<workset-spec>]
```

Example

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

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

item-id>	may be omitted if *<filename>* 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 workset used for this command. If omitted, the latest revision is deleted (see note in Introduction on page 23)

- `/FILENAME=<filename>`

  specifies the name of the file containing the item in the item-library.

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken. All item revisions to be affected by the command must be within the workset.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* 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.

DIFF - Compare Items

```
<item-spec>
[ /FILENAME=<filename>]

[   <item-spec>
             [ /FILENAME1=<filename>]
or  /USER_FILENAME=<user-filename>]

[ /LIST=<list-filename>]
[ /WORKSET=<workset-spec>]
```

NOTE DIFF cannot be run from the Dimensions Agent for OS/390.

Examples

```
DIFF PROD:"QUERY RELEASE".AAAA-SRC;2
DIFF PROD: /FILENAME=qr.c /USER=new_qr.c
```

- `<item-spec>`

  specifies either or both of the items to be compared. Each comprises:

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

`<item-id>`	may be omitted if the *`<filename>`* associated with that *`<item-spec>`* is specified.
`<variant>`	may be omitted if only one exists for that *`<item-id>`*.

<item-type>	may be omitted if the <filename> associated with that *<item-spec>* is specified and is unambiguous (i.e. *<item-type>* must not be omitted if the *<filename>* appears in two or more different item libraries).
<revision>	defaults to the latest revision.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *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 a file in the user-area (or elsewhere, if directory names, either absolute or relative to the user-area, are included along with the filename), which is to be compared with the first <item-spec>.

  It must be omitted if both *<item-spec>*'s have been specified.

  If only one *<item-spec>* is specified, and *<user-filename>* also is omitted, then a default second *<item-spec>* is used. This default is the same item as the first *<item-spec>*, but the revision which is next-latest to it (see under "Description" below).

- `/LIST=<list-filename>`

  is the name of a file in the user area, into which the difference report will be placed.

If omitted, the report is either displayed at the terminal when the command is entered at the *`Dimensions>`* or operating system prompt, or included in the mailed log when the command is used in batch mode (in a command file processed by the Dimensions function XCMD).

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

  ```
  <product-id>:<workset-id>
  ```

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

  Item revisions to be affected by the command may be specified explicitly, and will be selected from the workset.

Description

This program compares two item revisions or one revision with an ordinary sequential disk file, and reports the differences between them graphically.

The operating system command used to perform the comparison is specified by *`PCMS_DIFF_SCRIPT`* – this is detailed in the *`Developer's Toolkit Reference Guide`*. You can customize this script to use a different "diff" tool to the one normally used by Dimensions.

When no revision is specified for an item, the latest revision is chosen. This is the one which has been created or updated most recently (by any of the Dimensions functions RI, EDI or UI, or as a built item revision by IP, or, if none of them has been used, then by CI); so it need not be the one whose revision field has the highest collating value.

Similarly the default second revision, chosen when both the second *`<item-spec>`* and the *`<user-filename>`* are omitted, is the one which has been created or updated next-to-most recently, using the same selection process.

NOTE The history records of the item are not consulted; and there is therefore no guarantee that the first of the revisions being compared is one which was originally created by copying from the second.

Constraints

This command can be run only by users who have, for each item, a role on the design part owning the item or a role on one of the ancestor nodes of that design part.

Directory-item revisions cannot be compared.

DIR - Define Item Relations

```
<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"
```

- `<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

This command can be run only by a user with the role of *`PRODUCT-MANAGER`*.

DLGC - Delegate Change Document

```
<chdoc-id>
/ROLE=<role>
/USER_LIST=(<user1>,<user2>, ... )
 [ /CAPABILITY=<capability>]
    [ /ADD ]
or  /REPLACE
or  /DELETE
```

Example

```
DLGC PROD_DR_25 /ROLE=REVIEWER -
    /CAPABILITY=P /USER= SMITH
```

- `<chdoc-id>`

  identifies the change document for which delegation is to be made.

- `/ROLE=<role>`

  identifies the role title to be delegated.

- `/USER_LIST=(<user1>, ... )`

  identifies by login username one or more Dimensions users to whom delegation of the role is to be made for this change document.

- `/CAPABILITY=<capability>`

  specifies that this role delegation is to be **P** for Primary, **S** for Secondary (default) or **L** for Leader. (See AUR function 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 change document 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 change document 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 change document.

Constraints

This command can be run only by a user with the role of *CHANGE-MANAGER* or by users for whom the change document to be delegated is in their pending list.

DLGI - Delegate Item

```
<item-spec>
[ /FILENAME=<filename>]
/ROLE=<role>
/USER_LIST=(<user1>,<user2>, ... )
[ /WORKSET=<workset-spec>]
/CAPABILITY=<capability>]
    [ /ADD
or   /REPLACE
or  /DELETE]
```

Example

```
DLGI  PROD:"QUERY RELEASE"-SRC -
    /ROLE=REVIEWER /CAPABILITY=P -
    /USER= SMITH /FILENAME= qr.c
```

- `<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=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory of the file to be used when the item is checked out from the current workset. The workset filename for the same item may **differ** between worksets e.g. *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 username one or more Dimensions users to whom delegation of the role is to be made for this item.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

- `/CAPABILITY=<capability>`

  specifies that this role delegation is to be **P** for Primary or **S** for Secondary (default) or **L** for Leader. (See AUR function 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 role of *`PRODUCT-MANAGER`* or by users for whom the item to be delegated is in their pending list.

DNP - Define New Product

```
<product-id>
[ /BASED_ID= <based-on-product>]
    [ /[NO]IPDS]]
    [ /STRUCTURE= ALL]
    [ /STRUCTURE= NO]
/DESCRIPTION= <description>
/PRODUCT_MANAGER= <product-manager>
[ /PARTS_CONTROLLER= <parts-controller>]
[ /CHANGE_MANAGER= <change-manager>]
/VARIANT= <variant>
/PCS= <pcs>
```

Example

```
DNP PRODTEST20 /VARIANT= AAAA /PCS= 1 -
    /DESC = "PROD Rel 2.0 Test Vehicle" -
    /PRODUCT_MANAGER = SMITH
```

- `<product-id>`

  specifies the identity of the new product.

- `/BASED_ID=<based-on-product>`

  specifies the *<product-id>* of an existing product on which the new one is to be based.

  If omitted, the new product is based on the *$GENERIC* product.

  `/IPDS`

  Copy IPDs from the *<based-on-product>* to the new product. The default is */NOIPDS*.

  `/NOIPDS`

  Do **not** copy IPDs from the *<based-on-product>* to the new product. This is the default.

`/STRUCTURE=ALL`

Copy structure from the *`<based-on-product>`* to the new product. The default is *`/STRUCTURE= NO`*

`/STRUCTURE=NO`

Do **not** copy structure from the *`<based-on-product>`* to the new product. This is the default.

- `/DESCRIPTION=<description>`

  specifies a description of the 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 also automatically assigned the role of *`PCMS-ROLE-MANAGER`* and *`PCMS-PART-MANAGER`* for the associated workset – other users can also be assigned the *`WORKSET-MANAGER`* role via the DUR function.)

- `/PARTS_CONTROLLER=<parts-controller>`

  assigns the role of *`PARTS-CONTROLLER`* to the specified user.

  It defaults to *`<product-manager>`*, if omitted.

- `/CHANGE_MANAGER=<change-manager>`

  assigns the role of *`CHANGE-MANAGER`* to the specified user.

  It defaults to *`<product-manager>`*, if omitted.

- `/VARIANT=<variant>`

  specifies the default *`<variant>`* which is to be used by Dimensions for this product, whenever new design parts and items are created.

- `/PCS=<pcs>`

  specifies the PCS which is to be used by Dimensions for this product, whenever new design part variants are created.

Constraints

This command can be run only by a user with the role of `TOOL-MANAGER`.

DPL - Define Product Libraries

```
<product-id>
    /ITEM_TYPE=<item-type>
              [ /DELTA ]
/LIBRARY=<directory-name>
/NETWORK_NODE=<dfs-node>
[ /PROTECTION=<protection>]
[ /ADD
    or       /UPDATE
    or       /DELETE]
```

NOTE For information about managing Dimensions item libraries on an OS/390 mainframe for access through the Dimensions Agent for OS/390, see the *Dimensions Agent for OS/390 Administrator's Guide*.

Example

```
DPL PROD /ITEM_TYPE=DATA /NET=eldarmar -
    /LIB= "/usr/PROD/library/item/data/"
```

- `<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 1: 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.

NOTE 2: The compress storage option is not available for delta library storage (see *Process Modeling User's Guide*).

- `/LIBRARY=<directory-name>`

  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._

- `/NETWORK_NODE=<dfs-node>`[1]

  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.

1. For the Dimensions Agent for OS/390, *<dfs-node>* must be a logical node name *not* a physical node name. For more details concerning logical and physical nodes see the *Network User's Guide*.

- `/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 NT/2000 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*.

Protecting the Item Library Files from Unauthorized Changes on Windows NT/2000

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 PVCS 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 change document libraries. Change documents are automatically stored by Dimensions in the RDBMS database.

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* 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.

DPV - Delete Design Part Variant

```
<part-spec>
```

Example

```
DPV PROD:"RELEASE MANAGEMENT".IBM
```

- `<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

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* or a user appropriately assigned the role of *`PCMS-PART-MANAGER`*.

The design part must meet the following criteria:

- it is not referenced in an open change document
- 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

```
<release-spec>
```

Example

```
DREL PROD:"R M 2.0 FOR HP"
```

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

Constraints

This command can be run only by a user with the role of *PRODUCT-MANAGER* on the product that owns the release or the owner of the baseline on which the release was based.

DUR - Define User Roles

```
<product-id>
/ROLE=<role>
[ /DESCRIPTION=<description>]
[ /ADD ]
or  [ /DELETE ]
```

Example

```
DUR PROD /ROLE=DEVELOPER -
    /DESC= "Creates and edits items"
```

- `<product-id>`

  identifies the product in which a role title is (to be) defined.

- `/ROLE=<role>`

  specifies a role title for use in the lifecycles used by this product.

- `/DESCRIPTION=<description>`

  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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *PCMS-ROLE-MANAGER*.

The role of *PCMS-ROLE-MANAGER* is sub-ordinate to the role of *PRODUCT-MANAGER* and cannot therefore be used to define that superior role.

DVB - Define Version Branch

```
<branch-id>
 /DESCRIPTION=<description>
[/[NO]LOCK]
```

Example

```
DVB MAINT -
    /DESCRIPTION="Principal branch for maintenance work"
```

- `<branch-id>`

  unique branch identifier.

- `/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** workset by use of the Set Workset Attributes (SWS) command. Alternatively, a valid list of branch-ids can be associated to a *new* workset at the time the workset is defined using the Define New Workset (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

This command can be run by a user with the role of *`TOOL-MANAGER`* or *`PRODUCT-MANAGER`*

DWP - Delete Whole Product

```
<product-id>
```

Example

```
DWP PRODTEST20
```

The DWP command deletes all the information about a product (items, change documents, design parts, 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 change documents are stored in the database they *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 workset to a product's items will be automatically removed from that workset when the product is deleted.

Constraints

This command can be run only by a user with the role of *`TOOL-MANAGER`*.

DWS - Define New Workset

```
<workset-spec>
/DESCRIPTION=<description>
[/WORKSET=<workset-spec>]
[/DIRECTORY=<directory-spec>]
[/BASELINE=<baseline-spec>]
[/STATUS=<status>]
[/BRANCH|/TRUNK]
[/[NO]AUTO_REV]
[/VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)]
[/DEFAULT_BRANCH=<branch-id>]
```

Example

```
DWS "PROD_X:TEST_WS" -
    /DESCRIPTION="Workset for portation work" –
    /BASELINE="PROD_X:prod 2.3 beta"
```

- `<workset-spec>` comprises:
  `<product-id>:<workset-id>`
- `/DESCRIPTION=<description>`

  specifies the description to be attached to the workset definition.
- `/WORKSET=<workset-spec>`

  is an optional qualifier and specifies the workset on which to base the new workset.
- `/DIRECTORY=<directory>`

  specifies the top level directory specification for the workset. This “workset root directory” defines the point in the directory hierarchy structure below which (or relative to which) the workset filename is placed.
- `/BASELINE=<baseline-spec>`

is an optional qualifier and specifies the Dimensions baseline on which to base the new workset.

- `/STATUS=<status>`

  allows the new workset to be created in either a *`LOCKED`* or *`UNLOCKED`* (default) state. The *`LOCKED`* state prevents new Dimensions items being added to the workset (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-spec is edited/updated.

- `/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.

- `/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 workset. If this parameter is omitted, an empty list is created – the Set Workset Attributes (SWS) command can subsequently be used, if desired, to associate a valid list of branch-ids to the workset created here.

This list defines the branches on which newly created item revisions can be placed.

If the workset is defined with **one or more** valid branches, every **new** item revision in the new workset must use one of these branch-ids.

If the workset 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.

Description

The DWS command is used to create a new workset within a Dimensions product. The workset can be empty, based on an existing workset or based on an existing baseline.

The Remove Workset (RWS) command is used to remove a workset.

NOTE The */BRANCH*, */TRUNK*, */AUTO_REV* and */NOAUTO_REV* qualifiers may further be used to alter the options associated with the workset. The permitted combinations of these qualifiers are:

```
DWS <workset-spec> /BRANCH
DWS <workset-spec> /TRUNK
DWS <workset-spec> /AUTO_REV
DWS <workset-spec> /NOAUTO_REV
DWS <workset-spec> /BRANCH /AUTO_REV
DWS <workset-spec> /BRANCH /NOAUTO_REV
DWS <workset-spec> /TRUNK /AUTO_REV
DWS <workset-spec> /TRUNK /NOAUTO_REV
```

Constraints

Normally this command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

DWSD - Delete Workset Directory

```
<directory-path>
[ /WORKSET=<workset-spec>]
```

Example

```
DWSD src
```

- `<directory>`

  The DWSD command deletes a workset-directory *`<directory>`* from the database workset structure relative to the workset root directory (this sub-directory previously having been used for workset operations, but no longer being required).

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

Constraints

Normally this command can be run only by a user with the role of *`PRODUCT-MANAGER`* or *`WORKSET-MANAGER`* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

EDI - Edit Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /BASELINE=<baseline-spec>]
[ /REVISION=<new-revision>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /CHANGE_DOC_IDS=(<cd1>,<cd2>, ... ) ]
[ /STATUS=<status>]
[ /COMMENT=<comment text>]
[ /WORKSET=<workset-spec>]
[ /FORCE_UPDATE ]
[ /CODEPAGE=<code-page>| DEFAULT]
```

NOTE EDI cannot be run from the Dimensions Agent for OS/390 or the CMDCLIENT interface. However, EDI on other platforms still supports the `/CODEPAGE` qualifier.

Example

```
EDI PROD:"QUERY RELEASE".AAAA-SRC;1 /REV=2 -
    /COMMENT="edited to reflect memo ERB95B1"
```

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

<item-id>	may be omitted if *<filename>* is specified.
<variant>	may be omitted if only one exists.
<revision	defaults to the latest revision in the workset specified by */WORKSET*. If */WORKSET* is unspecified, the user's current workset will be assumed.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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 edited from it.) *`<baseline-spec>`* comprises:
  `<product-id>:<baseline-id>`

  If omitted, the specified or default *`<revision>`* (as described above) is selected for editing.

- `/REVISION=<new-revision>`

  specifies a new revision for the item. This new revision will be placed in the workset specified by *`/WORKSET`*. If *`/WORKSET`* is unspecified, then the new revision will be placed in the user's current workset.

  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 220 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=(<cd1>,<cd2>, … )`

`<cdN>`	identifies a change document to which the item revision is to be related **in-Response-to**.

- `/STATUS=<status>`

  specifies a valid changed status (lifecycle state) for the edited revision.

  NOTE The status, if specified, must be one which would be valid if AI had been used separately.

  If omitted, the revision is assigned, or remains at, the initial state (in the lifecycle defined for *<item-type>*).

- `/COMMENT=<comment text>`

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  this optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken.

Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

- `/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 OS/390 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.

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.

EI - Extract (Check Out) Item for Update

```
<item-spec>
[ /FILENAME=<filename>]
[ /BASELINE=<baseline-spec>]
[ /USER_FILENAME=<user-filename>]
[ /REVISION=<new-revision>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /CHANGE_DOC_IDS=(<cd1>,<cd2>, ... ) ]
[ /WORKSET=<workset-spec>]
[ /[NO]OVERWRITE ]
[ /CODEPAGE=<code-page>|DEFAULT|SOURCE ]
[ /TOUCH ]
```

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
```

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

<item-id>	may be omitted if `<filename>` is specified.
<variant>	may be omitted if only one exists.
<revision>	defaults to the latest revision in the workset specified by */WORKSET*. If */WORKSET* is unspecified, the user's current workset will be assumed.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

The workset filename identifies the relative pathname (directory plus filename) from the workset root directory of the file to be used when the item is checked out from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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 workset filename – i.e. the file created in the user area (the current directory) will have the same name as that of the item's workset filename.

- `/REVISION=<new-revision>`

  specifies a new revision for the item. This new revision will be placed in the workset specified by *`/WORKSET`*. If *`/WORKSET`* is omitted, then the new revision will be placed in the user's default workset.

  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 220 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=(<cd1>,<cd2>, … )`

`<cdN>`	identifies a change document to which the item revision is to be related **in-Response-to**.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  if specified, the new revision of the item will be placed in the workset, otherwise it will be placed in the user's current workset.

- `/[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 240 – 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.

- `/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 OS/390 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.

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.

EXIT - End Dimensions Execution

```
[NO PARAMETERS]
```

NOTE EXIT cannot be run from the Dimensions Agent for OS/390.

EXIT may be used (but is not essential) to mark the end of a command file which is specified to XCMD.

Constraints

None

FI - Fetch (Get) Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /BASELINE=<baseline-spec>]
[ /USER_FILENAME=<user-filename>]
[ /[NO]EXPAND ]
[ /WORKSET=<workset-spec>]
[ /[NO]OVERWRITE ]
[ /CODEPAGE=<code-page>\DEFAULT\SOURCE ]
[ /TOUCH ]
```

Examples

```
FI PROD:"QUERY RELEASE".AAAA-SRC;1
FI "FS:CBEVENT C.A-SRC;b1#4" -
    /USER_FILENAME="e:\cpjtest\cbevent.c" /NOEXPAND -
    /NOOVERWRITE
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory of the file to be used when the item is gotten from the current workset. The workset filename for the same item may **differ** between worksets e.g. *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 filename is omitted, it will default (with the exception described below) to the workset filename – i.e. the file created in the user area (the current directory) will have the same name as that of the item's workset filename.

  NOTE If the FI command is submitted via PC Client's command-line interfaces, the `/USER_FILENAME` qualifier is **compulsory**.

- `/NOEXPAND`

  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 Modeling User's Guide* for a discussion of item header substitution.

  The default is */EXPAND* provided `item type` was defined in the process model to request **item header substitution**.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

- `/[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 240 – 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.

  To clarify the above, consider the following example:

  ```
  FI "FS:CBEVENT C.A-SRC;b1#4" -
     /USER_FILENAME="e:\cpjtest\cbevent.c" /NOEXPAND -
     /NOOVERWRITE
  ```

  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 OS/390 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.

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

```
<item-spec>
[ /FILENAME=<filename>]
[ /WORKSET=<workset-spec>]
```

Example

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

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	may be omitted if the file version is not required.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Description

This program displays an item's filename, 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).

NOTE *For UNIX systems only*. The revision number, which is shown preceded by a semicolon (;), is in fact preceded by a dot (.) in the filename.

Constraints

This command can be run only by users who have a role for the owner part.

The function is not available for items held in delta libraries.

GREP – Search and Replace

```
/SEARCH=<text>
[/REPLACE_TEXT=<text>]
[/FILENAME=<text>]
[/COMMENT=<text>]
[/USER_FILE=<text>]
[/IGNORE_CASE]
[/LATEST]
[/RECURSIVE]
[/PRODUCT=<text>]
[/TYPE_LIST=(type,type)]
[/FORMAT_LIST=(format,format)]
[ /WORKSET=<workset-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
    /IGNORE_CASE
```

will cause Dimensions to search for all occurrences of `printf` ignoring case, in all *c* files in the workset 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" /REPLACE_TEXT="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 change document *`PAYROLL_CR_1`* will be searched and the output of the command will be placed in the log file "*`C:\output.log`*".

- `/SEARCH=<text>`

  specifies the text to find.

- `/REPLACE_TEXT=<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`

  only look at 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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This specifies the workset to be used for this command: failing this, the user's current workset will be taken. The *`WS_DIR`* qualifier can be used to further scope the nature of the search.

- `/WS_DIR=<directory>`

  specifies the workset directory to search

  NOTE To search the top directory in a workset use only the oblique character (`/`).

- `/PART=<part specification>`

  specifies the design part to search

- `/CHANGE_DOC_ID=<doc id>`

  specifies the change document 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.

INFOR - Information (display *README* file)

```
[NO PARAMETERS]
```

NOTE INFOR cannot be run from the Dimensions Agent for OS/390 or the CMDCLIENT interface.

Example

```
INFOR
```

INFOR accesses and displays the *README* file. This file contains the latest information regarding the Dimensions release. This information will normally be of interest to Dimensions administrators.

The ASCII version of the *README* file is provided in the top-level directory of each CD1 distribution media (there is a separate CD1 for each Dimensions Server operating-system type supported). The Dimensions/System Administrator is responsible for installing this *README* such that it gets picked up by INFOR; if this has not been done, a "skeleton" *README* file provides the instructions to install the "full" file.

Constraints

None

IP - Item Processing

```
<item-spec>
[ /FILENAME=<filename>]
[ /OUTPUT
             [ /PART=<part-spec>]
             [ /FORMAT=<format>]
             [ /DESC=<description>]
    ]
[   /CONFIG_PART= <c-part-spec>
    or       /BASELINE= <baseline-spec>]
[ /CHANGE_DOC_IDS=(<cdi>, ... ) ]
[   /IPD_ID= <ipd-id>
    or       /TEST_IPD= <ipd-filename>]
[ /EXT_ITEMS=(<ei-filename> [=<user-filename>] , ... ) ]
[ /ADD_FILES=(<a-filename>, ... ) ]
[ /NODB ]
[ /KEEP ]
[ /REPORT ]
[ /WS_FILENAME=<ws_filename>]
[ /WORKSET=<workset-spec>]
```

NOTE IP cannot be run from the Dimensions Agent for OS/390.

Example

```
IP PROD:"QUERY RELEASE"-SRC;2 -
    /CONFIG= PROD:"RELEASE MANAGEMENT"
```

- `<item-spec>`

  specifies the item (INPUT or OUTPUT) which is to be built. It comprises:

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

<item-id>	must be specified if the item specified is OUTPUT and has not been created previously. Otherwise, it can be omitted if *<filename>* is specified.	
<variant>	INPUT, may be omitted if the item has only one variant; OUTPUT, is ignored.	
<revision>	INPUT	if omitted, the latest revision in the workset specified by */WORKSET* (if */WORKSET* is unspecified, then the latest revision in the user's current workset)
	OUTPUT	is ignored
	the next higher revision number is assigned, or 1 if the item has not been created previously.	

- `/FILENAME=<filename>`

  specifies the name of the file holding the item in the item library. It must be specified, if the item specified is OUTPUT and has not been created previously. Otherwise, it can be omitted if *<item-id>* is specified.

- `/OUTPUT`

  indicates the item specified by *<item-spec>* is the main OUTPUT from the process.

  If omitted *<item-spec>* is an INPUT, used to build a derived item (see *Process Modeling User's Guide*).

- `/PART=<part-spec>`

  specifies the design part which will OWN the OUTPUT item.

  It can be omitted if the item has been created previously.

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

`<product-id>`	must be the same as for the item that is to be created.
`<part-id>`	defaults to *`<item-id>`* if omitted and a new item is being created.
`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the item is OWNED by the current PCS of the design part.

- `/FORMAT=<format>`

  specifies the format of the item which is to be created.

  If omitted, then:

  - if the item has been created previously, it defaults to the earlier format.
  - else, if *`<ipd-id>`* is specified, it defaults to the INPUT format.
  - else, it defaults to the *`<type>`* of the created *`<filename>`* (see CI on page 57 for details).

- `/DESC=<description>`

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

  It is ignored, if the item has been created previously.

  If it is omitted for a new item, Dimensions supplies default text.

- `/CONFIG_PART=<c-part-spec>`

  identifies a configuration in the current product structure from which all input items and environment items are to be selected in the build process.

  If omitted, it defaults to the design part OWNing the build OUTPUT.

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

`product-id>`	must be the same as for the specified item.
`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored; the current PCS is always used.

- `/BASELINE=<baseline-spec>`

  identifies a release-baseline from which all input items and environment items are to be selected for building.

  If omitted, the build will be made from current revisions of items, using *`<c-part-spec>`* (or its default) as above.

  It comprises: *`<product-id>:<baseline-id>`*

  `<product-id>`

  must be same as for the specified item.

- `/CHANGE_DOC_IDS=(<cdi>, … )`

  `<cdi>`

  identifies a change document to which the modified revisions of affected items have been related. These revisions are selected in preference to those indicated by a configuration part or baseline.

- `/IPD_ID=<ipd-id>`

  is the identity of a declared IPD which is to be used to process the specified main item.

  If both *`<ipd-id>`* and *`<ipd-filename>`* are omitted, then the declared operational default IPD will be used.

- `/TEST_IPD=<ipd-filename>`

  identifies a user-area file which contains an IPD definition which is to be used to process the specified main item.

  No output files from this IPD are stored within Dimensions when this IPD is used, regardless of the *`/DB`* options which may be specified.

- `/EXT_ITEMS=(<ei-filename>[=<user-filename>], … )`

  `<ei-filename>`

  identifies the item library filename of an item when a file in the user area is to be used in place of the corresponding item in the item library.

  No output file is stored within Dimensions when this file is processed, regardless of the *`/DB`* options that may be specified in the IPD which is used to process the file.

  `<user-filename>`

  specifies the name of the *`<ei-filename>`* file in the user area.

  If omitted, then the user area file must have the same name as the Dimensions-file (specified by *`<ei-filename>`*).

- `/ADD_FILES=(<a-filename>, … )`

  `<a-filename>`

  specifies the name of an additional (non-Dimensions) file in the user area which is needed by the build tool.

The names of *<a-filename>* files are passed to the build tool as a list of filenames via the *#addfiles* symbol in an IPD.

- `/NODB`

  No files are stored within Dimensions when this parameter is specified, regardless of the */DB* options that may be specified in any IPD which may be used.

- `/KEEP`

  specifies that the output files are to be retained in the user area, regardless of the */NOKEEP* options that may be specified in any IPD which may be used.

- `/REPORT`

  prevents the execution of the build command-file generated by the item processor. The file can be examined to see the details and extent of the processing required for the specified build.

- `/WS_FILENAME=<ws_filename>`

  identifies the pathname (directory plus filename) of the file to be used in conjunction with the workset.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

NOTE The workset filename as used in build consists of the relative directory path from the workset root directory *<directory-spec>* and the filename from *<ws_filename>* concatenated together. Users wanting to put objects and sources in the same subdirectory must remember to include *#citpath* in their IPDs.

Constraints

This command can be run only by users who have a valid role for each of the items involved in the build process, and the relevant item-process definitions and data must have been previously declared to Dimensions.

This build process is performed by a generated batch job. Two or more such batch jobs from the **same user** must **not** be allowed to execute concurrently.

LCK - Lock Workset

```
workset <workset-spec>
```

Example

```
LCK WORKSET PROD_X:TEST_WS
```

- `<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  The specified workset must exist.

Description

This command locks the workset, with `workset` as a fixed parameter and *`<workset-spec>`* a user-defined parameter._

The locked state prevents the addition of new Dimensions items to the workset or the removal of existing item revisions that are in a locked workset (baselining is likely to occur in this state). In addition, items that exist in a locked workset will not be actionable or updateable from another workset unless the user has the role of *`PRODUCT-MANAGER`* for the item's product.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* or *`WORKSET-MANAGER`* for the workset concerned.

LWS - List Worksets

```
[ /FILENAME=<filename> ]
```

Example

```
LWS /FILENAME=worklist.txt
```

- `/FILENAME=<filename>`

  specifies that the list of worksets is output into the file specified in your ‘home’ directory.

  For each workset, the associated product, workset-id, workset-status, workset-owner and users with the role of *`WORKSET-MANAGER`* are detailed.

  LWS from PC Client **must** include this qualifier, which is created on the server.

  LWS without this qualifier outputs the list to the screen (*`stdout`* on UNIX systems).

Constraints

None

LWSD - List Workset Directories

```
<directory-path>
[ /RECURSIVE]
[ /FILES] or
    [ /ITEMS] or
    [ /FILES/ITEMS]
[ /USER_FILENAME=<filename>]
[ /WORKSET=<workset-spec>]
```

Example `LWSD src`

- `<directory>`

  The LWSD command lists:

  - The current workset-id or that specified by */WORKSET.*
  - The identities of the operating system directories of the worksets at subdirectory *<directory>* relative to the workset root directory.
  - The workset description.
  - The date at which the list was taken.
  - The items the workset 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 filename.

- `/RECURSIVE`

  recursively list all workset directories from the point defined above.

- `/FILES`

  list items using item library filenames (the default).

  - `/ITEMS`

    lists items using (traditional Dimensions) item specifications.

  - `/FILES/ITEMS`

    list items using both item-library filenames and (traditional Dimensions) item specifications.

- `/USER_FILENAME=<user-filename>`

  specifies that the list should be output to a file *`<user-filename>`* rather than to the screen.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

Constraints

None

MCPC - Move Change Document To Primary (Main) Catalog

```
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
```

Example

```
MCPC /CHANGE=(PROD_DC_22,PROD_DC_23)
```

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )`
  `<cdN>`

  identifies the change document(s) that are to be moved from the secondary catalog to the primary (main) catalog.

  NOTE The secondary catalog is intended mainly for change documents that are no longer active, and therefore users are not permitted to update a change document in this catalog.

Constraints

This command can be run only by a user who has *CHANGE-MANAGER* privileges.

Update functions are available only from the main catalog.

MCSC - Move Change Document To Secondary Catalog

```
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
```

Example

```
MCSC /CHANGE=(PROD_DC_30,PROD_DC_31)
```

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )`
  `<cdN>`

  identifies the change document(s) that are to be moved from the primary (main) catalog to the secondary catalog.

NOTE The secondary catalog is intended mainly for change documents that are no longer active, and therefore users are not permitted to update a change document in this catalog.

Constraints

This command can be run only by a user who has *CHANGE-MANAGER* privileges.

Update functions are available only from the main catalog.

MDR - Move Design Part Relationship

```
<part-spec>
/PARENT_PART= <part-spec>
```
[1]

Example

```
MDR PROD:"RELEASE MANAGEMENT".AAAA -
/PARENT_PART= PROD:"CONFIG DEF"
```

- `<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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *PCMS-PART-MANAGER*.

1. The parameter */FATHER_PART=<father-part-spec>* will also continue to be supported for backward compatibility.

MI - Merge Items

The MI command exists in two forms

- *Type A* whose syntax is used to merge two items to produce a new revision of the primary item.
- *Type B* whose syntax is used to merge two or more revisions of the same item.

Type A

The MI command will invoke a user configurable script, which in turn will invoke any manual or automatic merge tool. If after the merge, a merged file is available it will be used to create a new revision of the primary item.

NOTE This form of the MI command syntax is supported only on UNIX systems. It cannot be run from the Dimensions Agent for OS/390 or from a Windows platform.

```
<item-spec>
<item-spec1>
[/FILENAME=<filename>]
[/FILENAME1=<filename1]
[/REVISION=<new-rev-of-item-spec-1>
[/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... ) ]
[/COMMENT=<comment text>]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[/WORKSET=<workset-spec>]
```

Example

```
MI "PROD_X:HELLO WORLD.AAAA-C;2.1" -
    "PROD_X:HELLO WORLD.AAAA-C;2.2" -
    /REVISION="3.0" /CHANGE=PROD_DC_22 -
    /COMMENT="merged for CRB 73"
```

NOTE This command is used by Motif Client to merge multiple revisions in a series of merges.

- `<item-spec>`

  specifies the primary item that is to be merged

- `<item-spec1>`

  specifies the secondary item to be merged into the primary item

- `/FILENAME=<filename>`
  `/FILENAME1=<filename1>`

  specify the names of the workset filenames.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/REVISION=<new-revision>`

  specifies the new revision of the primary item that is to be created once the merge is completed and a merged file is available

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )`

  `<cdN>`

  identifies a change document to which the merged item revision is to be related *`in-Response-to`*.

- `/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.

- `/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>`*.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

Type B

The <*new-revision*> is created from the revision identified by <*primary-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*>.

The secondary item must not be specified in this form of the command.

Both /REVISION_LIST and /USER_FILENAME must be specified.

NOTE This form of the MI command syntax is supported on Windows and on UNIX.

```
<primary-item-spec>
/REVISION_LIST=(<merge-rev-1>,<merge-rev-2>,...)
/USER_FILENAME=<merged-file>
[/REVISION=<new-revision>]
[/COMMENT=<comment-text>]
[/FILENAME=<primary-item-filename>]
[/CHANGE_DOC_IDS=(<cd1>,<cd2>,...)]
[/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>,...)]
[/KEEP]
[/WORKSET]
```

- `/REVISION_LIST=(<merge-rev-1>,<merge-rev-2,...)`

  specifies that each revision specified has been merged

- `/USER_FILENAME=<merged-file>`

  the user file for *<new-revision>*

- `[/REVISION=new-revision>]`

  identified by *<primary-item-spec>* using the specified *<merged-file>* as the user file.

- `[/KEEP]`

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

-
  ```
  [/COMMENT=<merge-comment>]
  [/FILENAME=<primary-item-filename>]
  [/CHANGE_DOC_IDS=(<cd1>,<cd2>,...)]
  [/ATTRIBUTES=(<attr1>=<value1>,<attr2=value2>,...)]
  [/WORKSET]
  ```

  as Type A.

Constraints

These commands can be run by users who have:

- For each item to be merged, a role on the design part owning the item or a role on one of the ancestor nodes of that design part.
- For the resultant merged item, 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

```
<item-spec>
[ /FILENAME = <filename>]
/PART = <part-spec>
[ /WORKSET=<workset-spec>]
```

Example

```
MIP PROD:"QUERY RELEASE".AAAA-SRC -
/PART= PROD:"RELEASE CONTROL".AAAA
```

- `<item-spec>` comprises:
  `<product_id>:<item_id>.<variant>–<item-type>;<revision>`

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are moved to the specified design part.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory of the file to be used when the item is gotten from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`*.

MIT - Move (Change) Item Type

```
<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 PVCS:"MAIN".AAAA-SRC /TYPE=CODE
```

In this example the item's type is changed from *SRC* to *`CODE`*.

- `<item-spec>` comprises:
  `<product_id>:<item_id>.<variant>–<item-type>;<revision>`

`<item-id>`	may be omitted if *`<filename>`* 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

- This command can be run only by a user with the role of *`$PRODUCT-MANAGER`* or *`PRODUCT-MANAGER`*.
- You *cannot* move an item's type if the item:
  - Is on a named branch that remotely owned e.g. the item has been replicated (see *Replicator User's Guide*).
  - Is in the Dimensions OFFLINE state (see *ART User's Guide*).
  - Is a Dimensions IPD.
  - Has been included in a Dimensions build.
  - Is in a Dimensions baseline.
  - Is in a Dimensions release.
  - Is checked out.
  - Is referenced in an open change document.
  - 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 Change Document

```
<top-chdoc-id>
<target-product-id>
[ /CH_DOC_TYPE=<target-chdoc-type>]
[ /AFFECTED_PARTS=(<target-part-spec1>, ... ) ]
[ /CHANGE_DOC_IDS=(<doc-or-part1>,<doc-or-part2>, ... ) ]
[ /[NO]CHECK ]
```

Example

```
MVC OBLR_DR_52 PROD /CHANGE_DOC_IDS= (OBLR_DR_53, -
    PROD:"CONFIG DEF", OBLR_DR_55, OBLR_DR_56, -
    PROD:"CONFIG DEF", PROD:PROD.AAAA)
    /NOCHECK
```

- `<top-chdoc-id>`

  identifies the (principal) change document in the source product. Its dependent-related children can also be specified for moving at the same time (as detailed below).

  More exactly, the change document(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 change document(s) is (are) to be moved. It must be another product in the same Dimensions database. (To copy change documents to a different database, the baseline transfer facility of Dimensions ART may be used.)

- `/CH_DOC_TYPE=<target-chdoc-type>`

  specifies the change document type that the clone of *`<top-chdoc-id>`* is to have in the target product. The dependent children moved, if of the same type as

<top-chdoc-id>, are also translated to this type. (Any other dependent children do not receive a type translation.)

If omitted, all the cloned change documents 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-chdoc-id>*.

  If omitted, just the target product's top (i.e. product level) design part (its original variant) is related to this cloned change document.

- `/CHANGE_DOC_IDS=(<doc-or-part1>, ... )`

  where each *<doc-or-partN>* is of the form: either *<source-chdoc-id>* or *<target-part-spec>* although the second form cannot be used for the first entry in the list.

  The entries *<source-chdoc-id>* each identify a change document which has a relationship to *<top-chdoc-id>* in the Dependent class, and which is to be included in the group move.

  All the *<target-part-spec>* entries (if any) following one *<source-chdoc-id>* entry and up to the next one (or the end of the list) specify design part(s) which are to be related to the clone of the preceding *<source-chdoc-id>*. If there are no such entries (i.e. one entry of the form *<source-chdoc-id>* is immediately followed by the next one – or the end of the list), then just the target product's top (i.e. product level) design part (its original variant) is related to the clone of the preceding *<source-chdoc-id>*.

  If the */CHANGE_DOC_IDS* qualifier is omitted, then just *<top-chdoc-id>* is moved by itself.

- `/CHECK` or `/NOCHECK`

  specifies whether MVC is merely to check and report on the feasibility of moving the specified change document(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 change documents 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 chdoc-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 chdoc-id from which it was cloned.

For each change document moved, if there are user-defined attributes which have been declared for use by the product change document 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 role of *CHANGE-MANAGER* for the source product, or by users if all the specified change documents are in their Pending list.
- None of these change documents 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 change documents that are to be moved must all be related to the parent as Dependent and **not** Info.
- None of these change documents can be related in the *`Info`* relationship class to any other change documents, in either direction.
- The <*`top-chdoc-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 change document other than <*`top-chdoc-id`*>.
- The <*`top-chdoc-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 change documents must still be Open).
- The parameter *`$LAST`* (see note on the CC command - page 56) is not set by MVC, so the cloned change documents cannot be referenced subsequently in the same XCMD command file.
- This command does not copy attribute history information.

MWS - Merge Worksets

```
<workset-spec1>
<workset-spec2>
[/WORKSET=<workset-spec3>]
[/REPORT]
[/STATUS=<status>]
```

Example

```
MWS PROD:WS_001 -
    PROD:WS_002 /REPORT /WORKSET=PROD_X:TEST_WS
```

- `<workset-spec1>`

  comprises the specification for Workset 1:
  `<product-id>:<workset-id>`

  This workset is one of the inputs to the merge operation, and is also the target workset if */WORKSET* is not specified.

- `<workset-spec2>`

  comprises the specification for Workset 2.

  This workset is the second input to the merge operation.

- `/WORKSET=<workset-spec3>`

  This optionally specifies the target workset, which will be created if it does not already exist.

- `/REPORT`

  request that only the Merge Workset report be generated. This contains information concerning how the merge is processed. The report is mailed to the user.

- `/STATUS=<status>`

  allows the merged workset to be created in either a *LOCKED* or *UNLOCKED* state. The locked state prevents new

Dimensions items being added to the workset (baselining is likely to occur in this state).

The MWS command merges the two worksets given by <*workset-spec1*> and <*workset-spec2*>. The merged output is placed in a target workset, which is either:

- that given by <*workset-spec3*>, or
- that given by <*workset-spec1*> (if <*workset-spec3*> is not specified).

Constraints

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the target workset concerned.

MWSD - Move Workset Directory

```
<directory-path1>
<directory-path2>
[ /WORKSET=<workset-spec>]
```

Example `MWSD src dst`

- `<directory-pathN>`

  The MWSD command moves the workset structure (and items) from the source directory to the destination directory.

`<directory-path1>`	is the source directory.
`<directory-path2>`	is the destination directory.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

Constraints

Normally this command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

QIIP - Query Items with IPDs

`<part-spec>`

Example `QIIP PROD:"TEST CONFIGURATION"`

- `<part-spec>`

  identifies the configuration (design part) for which details of items and their IPDs are to be reported.

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

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored, the current PCS is always used.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`*.

QIMO - Query Item Made-of List

```
<item-spec>
[ /FILENAME=<filename>]
[ /DETAIL
              [ /NODERIVED ]]
[ /HISTORY ]
```

Example

```
QIMO PROD:"QUERY RELEASE"-EXE /DETAIL /HISTORY
```

- `<item-spec>`

  specifies the item, output from a build process, that is to be reported. It comprises:

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

<item-id>	may be omitted if <*filename*> is specified.
<variant>	may be omitted if only one exists.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *src/hello.c*, *hello.c* or *src/build/hello.c*.

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

- `/DETAIL`

  specifies that the source of items which were input to the build of the specified item are to be reported.

If omitted, only items that are direct input to the process of the specified output item are reported.

- `/NODERIVED`

  is relevant only when *`/DETAIL`* is specified. It specifies that intermediate derived items are to be excluded from the report. That is, only the basic source items will be reported.

  If omitted, input product items at every stage, both basic source and intermediate derived, are reported.

- `/HISTORY`

  specifies that the details for all revisions of the specified output item are to be reported.

  If omitted, only details of the specified revision of the item are reported.

Constraints

This list is a available only after an item has been subject to the build process.

RAA - Generate Documents Index

```
<part-spec>
```

NOTE RAA cannot be run from the Dimensions Agent for OS/390.

Example

```
RAA PROD:"RELEASE MANAGEMENT".AAAA
```

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

`<variant>`	may be omitted if only one exists.
`<pcs>`	is not applicable and may be omitted.

Constraints

- This command can be run only by a user who has a role on the selected design part which must have been included in at least one release-baseline.
- The number of items which may be included in the index is limited to 200 per design part.
- The index includes the latest PCSs, up to a maximum of 25.

RAH - Generate PCS Requirements List

```
<part-spec>
```

NOTE RAH cannot be run from the Dimensions Agent for OS/390.

Example

```
RAH PROD:"RELEASE MANAGEMENT".AAAA
```

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

`<variant>`	may be omitted if only one exists.
`<pcs>`	is not applicable and may be omitted.

Constraints

- This command can be run only by a user who has a role on the design part which must have been included in at least one release-baseline.
- The number of subordinate design parts which may be included in the list is limited to 1000 per design part.
- The list comprises nine columns to cope with a maximum of nine PCSs which are always the latest if their number exceeds nine.

RCCD - Relate Change Documents to Change Document

```
<chdoc-id>
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
[            /RELATIONSHIP=<relname>
    or       /DEPENDENT
    or       /INFO]
```

Example

```
RCCD PROD_CN_4 /CHANGE=PROD_DR_25
```

- `<chdoc-id>`

  is the identity of the change document to be the parent in the relationship to be created.

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )`

  specifies the identities of one or more change documents 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 change document concerned. Such users must have the parent Change Document in their Pending List.

However, the user with the role of *`PRODUCT-MANAGER`* can setup the Process Model to specify that no change document relationships can be created for certain change document types.

RCI - Report Current Items

```
<filename>
    /NEW
    or       [ /OLD ]
[   /PART=<part-spec>
    or       /BASELINE=<baseline-spec>]
[ /SORT=<sort>]
[ /WORKSET=<workset-spec>]
```

NOTE RCI cannot be run from the Dimensions Agent for OS/390.

Example

```
RCI sunrm3_01.export /NEW -
/PART= "PROD:RELEASE MANAGEMENT.AAAA"
```

If /NEW is specified

- `<filename>`

  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.

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 *`<filename>`*, *`<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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

If /NEW is omitted

- `<filename>`

  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>`

  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>`

  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=<workset-spec>` comprises:
  `<product-id>:<workset-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

```
<filename>
    /NEW
    or       [ /OLD ]
[   /PART=<part-spec>
    or       /BASELINE=<baseline-spec>]
[ /SORT=<sort>]
```

NOTE RCP cannot be run from the Dimensions Agent for OS/390.

Example

```
RCP sunrm3_01.export /OLD
```

If /NEW is specified

- `<filename>`

  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.

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 *`<filename>`*, *`<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

- `<filename>`

  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>`

  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>`

  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.

RDS - Report Design Structure

```
<filename>
    /NEW
    or       [ /OLD ]
[   /PART=<part-spec>
    or       /BASELINE=<baseline-spec>]
[ /SORT=<sort>]
[ /LEVEL=<level>]
[ /STRUCTURE= (<option>, ... ) ]
[ /WORKSET=<workset-spec>]
```

NOTE RDS cannot be run from the Dimensions Agent for OS/390.

Example

```
RDS sunrm3_01.export /STRUCTURE= ALL
```

If /NEW is specified

- `<filename>`

  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.

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 *<filename>*, *<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 change documents.
  - 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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

If /NEW is omitted

- `<filename>`

  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>`

  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>`

  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=<workset-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 <release-spec>

```
/BASELINE=<baseline-spec>
[ /TEMPLATE_ID=<template-id>]
[ /DESCRIPTION=<description>]
[ /DIRECTORY=<directory>]
[ /FILENAME=<filename>]
[ /DELTA
             /PREV_RELEASE=<release-id> ]
[ /[NO]EXPAND ]
[ /TOUCH ]
[ /OVERWRITE ]
```

Example

```
REL PROD:"R M 2.0 FOR HP" -
/BASE= PROD:"R M VERSION 2 FOR HP"
```

- `<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>`

  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.

- `/FILENAME=<filename>`

  generates command script to `<filename>` but does not run the command

- `/DELTA`

  specifies a delta release is to be created, i.e. items which 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"
  ```

  i.e. you do **not** specify *PROD* for */PREV_RELEASE*.

- `/[NO]EXPAND`
  `/TOUCH`
  `/OVERWRITE`

  adds the specified qualifier to each FI command in the script generated by REL.

Constraints

This command can be run by users who can 'get the items' that comprise 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 workset 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.

RHA - Generate Design Part Parts List

```
<part-spec>
[ /MULTI_LEVEL ]
```

NOTE RHA cannot be run from the Dimensions Agent for OS/390.

Example

```
RHA PROD:"RELEASE MANAGEMENT".AAAA /MULTI
```

- `<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.

- **/MULTI_LEVEL**

  specifies that a multi-level Parts List is required.

  If omitted, only the immediate child design parts are listed.

Constraints

This command can be run only by a user who has a role on the selected design part whose PCS must be OPEN.

RHAB - Generate Parts List for a Baseline

```
<part-spec>
/BASELINE=<baseline-spec>
[ /MULTI_LEVEL ]
```

NOTE RHAB cannot be run from the Dimensions Agent for OS/390.

Example

```
RHAB PROD:"RELEASE MANAGEMENT".AAAB -
/BASELINE= PROD:"R M VERSION 2 FOR HP"
```

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

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored (reports baselined PCSs).

- `/BASELINE=<baseline-spec>` comprises:
  `<product-id>:<baseline-id>`
- `/MULTI_LEVEL`

  specifies that a multi-level Parts List is required.

  If omitted, only the immediate child design parts are listed.

Constraints

This command can be run only by a user who has a role on the design part (which must **not** be suspended) for which the list is being generated. The number of subordinate design parts which may be included in the list is limited to 1000.

RI - Return (Check In) Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /USER_FILENAME=<user-filename>]
[ /KEEP ]
[ /STATUS=<status>]
[ /COMMENT=<comment text>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /WORKSET=<workset-spec>]
[ /FORCE_UPDATE ]
[ /CODEPAGE=<code-page>|DEFAULT ]
```

Example

```
RI PROD:"QUERY RELEASE".AAAA-SRC;1 -
/KEEP /STATUS= "UNDER TEST"” -
/COMMENT="checked in for CRB 91"
```

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

<item-id>	may be omitted if *<filename>* 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.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *src/hello.c*, *hello.c* or *src/build/hello.c*.

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

- `/USER_FILENME=<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 function).

- `/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.

- `/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 220 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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  and is optional (subject to the following). A checked out item can **only** be checked in to the specific workset from which it was originally checked out (an error will be generated if you try to check it in to another workset). Therefore, if your current workset is not that workset, either this qualifier must be used to specify the correct check in *`<workset-spec>`* or your current workset must be set to that *`<workset-spec>`* using the Set Current Workset (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.

- `/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 OS/390 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.

Constraints

This command can be run only by the user who previously checked out the item revision or by the PRODUCT-MANAGER.

RICD - Relate Item to Change Documents

```
<item-spec>
[ /FILENAME=<filename>]
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
[   /AFFECTED
    or       /IN_RESPONSE_TO
    or       /INFO  ]
[ /WORKSET=<workset-spec>]
```

Example

```
RICD PROD:"QUERY RELEASE".AAAA-SRC;1 -
/CHANGE_DOC=PROD_DR_25
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, … )`

  <cdN> identifies a change document 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 change documents. The qualifiers are mutually exclusive.

  The default is */AFFECTED*.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Constraints

This command can be run only by a user who has the change document in his/her pending list or who has the role of *CHANGE-MANAGER*.

RII - Relate Item to Item

```
<src-item-spec>
<dst-item-spec>
[ /FILENAME=<src-filename>]
[ /FILENAME1=<dst-filename>]/RELATIONSHIP=<relationship-id>
[ /COMMENT=<comment-text>]
[ /WORKSET=<workset-spec>]
```

Example

```
RII "PROD_X:INTERFACE_C.AAAA-C;1" -
"PROD_X:INTERFACE_H.AAAA-H;1" /RELATIONSHIP="INCLUDES" –
/COMMENT="INCLUDED HEADER"
```

- `<src-item-spec>`

  specifies the source item from which the relationship instance will start.

- `<dst-item-spec>`

  specifies the destination item to which the relationship will be made.

- `/FILENAME=<src-filename>`
  `/FILENAME1=<dst-filename>`

  specify the names of the workset filenames.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *src/hello.c*, *hello.c* or *src/build/hello.c*.

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

- `/RELATIONSHIP=<relationship-id>`

  specifies the relationship type as defined by the DIR command.

- `/COMMENT=<comment-text>`

  is an optional qualifier and specifies a comment to be attached to the relationship instance.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  this optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

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 278) is used to remove such relationships.)

RIP - Relate Item to Part

```
<item-spec>
[ /FILENAME=<filename>]
/PART=<part-spec>
[ /WORKSET=<workset-spec>]
```

Example

```
RIP PROD:"QUERY RELEASE".AAAA-SRC -
/PART= PROD:"RELEASE MANAGEMENT".IBM
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are related to the specified design part.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This optionally specifies the workset to be used for this command: failing this, the user’s current workset will be taken.

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

RIR - Remove Item Relation Definition

```
<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"
```

- `<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

This command can be run only by a user with the role of *`PRODUCT-MANAGER`*.

RIWS - Remove Item Revision from Workset

```
<item-spec>
/WORKSET=<workset-spec>
[ /FILENAME=<filename>]
```

Example

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

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

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This specifies the workset from which the item revision is to be removed.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets *e.g. src/hello.c*, *hello.c* or *src/build/hello.c*.

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

Description

This command will remove the item specified from the given workset. The specified item and workset must exist. If the specified item is not in the workset a warning will be given.

Constraints

Normally this command can be run only by a user with the role of *`PRODUCT-MANAGER`* or *`WORKSET-MANAGER`* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

RMDF - Remove (Item) Data Formats

```
<format>
```

Example

```
RMDF TXT
```

- `<format>`

  the format definition being removed.

Description

The RMDF command removes existing defined item data formats. For a discussion on item data formats refer to the DDF command.

Constraints

This command can be run only by a user with the role of *`TOOL-MANAGER`*.

RMVB - Remove Version Branch

```
<branch-id>
[/FORCE ]
```

Example

```
RMVB MAINT
```

- `<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 unless the *`/FORCE`* qualifier is used.

- `/FORCE`

  removes branch-id even if it is being used. Warning messages will **always** be issued to the standard output following the forced removal of a used branch-id. Items with removed branch-ids will continue to exist, but no further revisioning operations will be possible with respect to them e.g. EDI, EI or UI will not be possible. You will continue to be able to get these items, and it is always possible to re-define the branch-ids again (using DVB) and carry on revisioning the items.

Description

The RMVB command removes existing defined version branch-ids. The *`/FORCE`* qualifier allows removal of branch-ids which are being used.

Constraints

This command can be run only by a user with the role of *`TOOL-MANAGER`*.

RP - Relate Design Part

```
<participle>
/PARENT_PART=<parent-part-spec>[1]
```

Example

```
RP PROD:"LIBRARY CONTROL".AAAA -
/PARENT= PROD:"RELEASE MANAGEMENT".AABB
```

- `<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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *PCMS-PART-MANAGER*.

1. The parameter */FATHER_PART=<father-part-spec>* will also continue to be supported for backward compatibility.

RPCD - Relate Part to Change Documents

```
<part-spec>
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
```

Example

```
RPCD PROD:"RELEASE MANAGEMENT".AAAA -
/CHANGE_DOC= (PROD_DR_25,PROD_DR_26)
```

- `<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=(<cd1>,<cd2>, … )`

`<cdN>`	is the identity of a change document to which the specified design part is to be related.

Constraints

- This command can be run only by the user who has the change document in their Pending list or by a user with the role of *CHANGE-MANAGER* (a "change-manager")
- This command cannot be used with the secondary catalog list.
- This command cannot be run if the change document 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 change document back to its closed state.

RPCP - Report Product Control Plan (Process Model)

```
<product-id>
[ /DETAIL ]
[ /SECTION=(<section>, ... ) ]
```

NOTE RPCP cannot be run from the Dimensions Agent for OS/390.

Example

```
RPCP PROD /DETAIL /SECTION=(ITEM,CDOC)
```

- `<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 change document types.
• ROLE	reports on roles and users.
• RULE	reports on change management rules.

• ALL	(or if `/SECTION` is not specified) reports on all sections of the process model.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`*.

RPNO - Report Part Numbers

```
<product-id>
```

Example

```
RPNO PROD
```

- `<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.

Constraints

This command can be run only by a user with the role of *`PARTS-CONTROLLER`*.

RPT - Report Change Documents

```
<product-id>
<chdoc-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=<chdoc-status>]
[ /USER=<username>]
[ /FROM=<date-from>]
[ /TO=<date-to>]
[ /NOHEADING_WRAP ]
[ /NOWRAP ][ /INDENT ]
[ /DETAIL ]
[ /HOLD ]
[ /PRINT ]
```

Example

```
RPT PROD 4 /NAME=CH_DOC_LIST /CATALOGUE /DETAIL -
    /FILE=workpack_all.list /PRINT
```

- `<product-id>`

  is the product for which a report is required.

- `<chdoc-category>`

  is the change document category for all change documents to appear in the report. It may be specified as **%** (percent) to permit change documents 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 *Reports Guide*.

  For a *`BASELINE_DETAIL REPORT`*, please refer to the separate entry on page 229

- `/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 change documents.
/CATALOGUE /SECONDARY	report based on the secondary catalog of change documents.
/PENDING	report based on the change documents 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 change documents 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 change document attribute).

The **default** in each case is to specify **no** additional criteria: i.e. all change documents, 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 change documents to be reported on. Each *`<attrN>`* is the Variable Name defined for one of the 220 attributes. Each *`<stringN>`* is a string for that attribute's value to match.

  Wild card **%** may be used in each `<stringN>` (see above).

- `/CHANGE_DOC_ID=<ch-doc-id>`

  is a string to be matched by the identifier of each of the change documents to be reported on.

  Wild card **%** may be used (see above).

- `/CH_DOC_TYPE=<ch-doc-type>`

  is a string to be matched by the type of each of the change documents 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 change document, in order for that change document to be included in the report.

  Wild card **%** may be used (see above).

- `/PHASE=<phase>`

  is a string to be matched by the current phase of each of the change documents to be reported on.

  Wild card **%** may be used (see above).

- `/STATUS=<chdoc-status>`

  is a string to be matched by the current status of each of the change documents to be reported on.

  Wild card **%** may be used (see above).

- `/USER=<username>`

  is a string to be matched by a Dimensions user associated with each of the change documents to be reported on.

  Wild card **%** may be used (see above).

  A *`/CATALOGUE`* or *`/SECONDARY`* report includes all change documents which (meet all other specified criteria and) have been created or actioned by a Dimensions user matching *`<username>`* at any time between *`<date-from>`* and *`<date-to>`*.

  A *`/PENDING`* report includes all change documents which (meet all other specified criteria and) are awaiting actioning by a Dimensions user matching `<username>` at any time between *`<date-from>`* and *`<date-to>`*. (A change document can be awaiting actioning by any of several Dimensions users. For valid inclusion of the change document in the report, at least one of these users must match *`<username>`* as specified here, but they need not all do so.)

- `/FROM=<date-from>`

  specifies the initial date associated with each of the change documents 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 change documents 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 change document 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

This command can be run only by a user with the role of *`CHANGE-MANAGER`*.

RPT - Baseline Detail Report

```
<product-id>
<chdoc-category>
/NAME=BASELINE_DETAIL
/USER_DEFINED
[ /BASELINE=<baseline-spec>]
[ /FROM=<date-from>]
[ /TO=<date-to>]
/FILENAME=<outfile>
[ /NOHEADING_WRAP ][ /NOWRAP ][ /INDENT ]
[ /HOLD ]    [ /PRINT ]
```

Example

```
RPT PROD 1 /NAME=BASELINE_DETAIL /USER_DEFINED -
    /FILE=hp_rm_chdocs96.rep -
    /BASELINE=PROD:"R M VERSION 2 FOR HP" -
    /FROM=01-JAN-1989 /TO=31-DEC-1996
```

- `<product-id>`

  is the product for which the report is required.

- `<chdoc-category>`

  is any valid change document category. (Its value is not used in this report.)

- `/NAME=BASELINE_DETAIL`
  `/USER_DEFINED`

  are specified as shown.

  For other types of report, please refer to the **RPT** entry for **Report Change Documents**, which precedes this one on page 223.

- `/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 change document, in order for the change document 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 CHANGE DOCUMENTS command on page 223 used here (including *`/DETAIL`*) will be ignored.

Constraints

This command can be run only by a user with the role of *`CHANGE-MANAGER`*.

RUR - Run User-Defined Report

```
<report name>
/PRODUCT=<product-range>
[ /PARAMETER=(<param2>,<param3>, ... ,<param8>) ]
[ /FILENAME=<output file>]
[ /PRINT ]
[ /HOLD ]
```

Example

```
RUR "VARIANTS TYPES FORMATS" /PROD="CAR%" -
/PARAM=(AAA_,"%",C) /PRINT
```

- `<report name>`

  specifies the kind of report to be produced. It must be one of the report names which has been set up by the *SUR* function as applicable to the operating system – UNIX or Windows NT/2000 – under which this *RUR* command is going to be executed.

- `/PRODUCT=<product-range>`

  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 "Standard Command Line and ASCII Reports" chapter 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>)`

  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. Therefore, until the user is thoroughly familiar with the exact parameter requirements of that particular report name, it is strongly recommended that `RUR` is executed in interactive mode instead — where there are prompts to guide the user as to the number, order and contents of the parameter values.

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 purpose 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 filename 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.

NOTE If you are a secure database user (which will usually be the case), then a special procedure will be needed to run command scripts (set up in *SUR*) which access auxiliary files containing Oracle SQL*Plus statements–this is explained further in the "Standard Command Line and ASCII Reports" chapter in the *Reports Guide*.

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.

RWS - Remove Workset

```
<workset-spec>
```

Example

```
RWS "PROD_X:TEST_WS"
```

- `<workset-spec>` comprises:
  `<product-id>:<workset-id>`

Constraints

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the workset concerned.

SCWS - Set Current Workset

```
[<workset-spec>]
[ /DIRECTORY=<directory> ]
/[NO]DEFAULT
```

Example

```
SCWS PROD:"WS CUSTOM" /DIR="/product9" -
    /NODEFAULT
```

- No parameters or qualifiers

  SCWS without any parameters or qualifiers will display:

  - Login user name
  - The current workset specification.
  - The workset root directory.
  - Whether the workset is locked or not.
  - Whether trunk or branch revisioning is enabled.
  - Whether or not revisioning is enforced.
  - Branches assigned to the workset.
  - The workset default design part.

NOTE SCWS without parameters or qualifiers displays the current workset of the current Dimensions session. Additionally, it is only refreshed from the database at the start of the session.

- `<workset-spec>` comprises:
  `<product-id>:<workset-id>`

<workset-spec>	must be stated when */DIRECTORY* and/or */(NO)DEFAULT* are included in the command.

- `/DIRECTORY=<directory>`

  specifies the top level directory specification for the workset. This “workset root directory” defines the point in the directory hierarchy structure below which (or relative to which) the workset filename is placed e.g. in UNIX *<dir>/<ws_filename>*. This overrides what would have been the default workset root directory.

  Checked out or gotten items will then be directed to the workset file whose pathname is identified by this new pathname e.g. in UNIX *<dir>/<ws_filename>*.

- `/DEFAULT or /NODEFAULT`

/DEFAULT	specifies that the current workset specified by this command will remain the default for all future Dimensions sessions until respecified. */DEFAULT* is the default.
/NODEFAULT	specifies that the current workset 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 workset will revert to its former setting once the session is exited. This workset 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 workset before such commands as `CBL`, and then used again afterward to reset it to its earlier specification (if so desired).

Description

This command sets the user's current workset, and optionally the user's default workset.

To set the current workset to the "Global Workset", the `<workset-spec>` must be set to *`$GENERIC:$GLOBAL`* (see Introduction on page 25).

Constraints

This command can be run by users who have a role on the workset being set.

SDF - Set Data Format Flags

```
<format>
[/DESCRIPTION=<format -description>]
[/CLASS=<class-no>]
[/MIME=<mime-type>]
```

Example

```
SDF TXT /DESCRIPTION="Plain text" -
    /CLASS= 1 /MIME=”TEXT/PLAIN”
```

- `<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=<mime-type>`

  specifies the new Multipurpose Internet Mail Extension (MIME) type. MIME types comprise seven broad categories, with each category also having sub-categories defined by using an oblique (/) separator. The broad categories are: Application, Audio, Image, Message, Multipart, Text and Video. An example of a sub-category is APPLICATION/WORD.

Description

This function 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 (Item) Data Formats (ADF) command (page 34).

This function is also available from the Process Modeler, File Formats and MIME Types option.

Please see the DDF command (page 90) for a description of the uses for file formats.

Constraints

This command can be run only by a user with the role of *`TOOL-MANAGER`* of the product.

SET - Set DIR, PRINTER or OVERWRITE Environment

```
PRINTER <printer-cmd>
or
DIRECTORY <directory-spec>
or
OVERWRITE ON|OFF
```

Example

```
SET PRINTER "lpr -Pdev"
```

- `<printer-cmd>`

  a valid UNIX or Windows NT/2000 printer command. This is used in preference to the value assigned to the *PCMS_PRINT* symbol.

- `<directory-spec>`

  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.

- `[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 133 and page 139 respectively.

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.

Constraints

This command sets the above Dimensions parameters only for the duration of the current Dimensions session.

SI - Suspend Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /WORKSET=<workset-spec>]
```

Example

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

or, to suspend all revisions

```
SI PROD:"QUERY RELEASE".AAAA-SRC;*
```

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

`<item-id>`	may be omitted if *`<filename>`* 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 workset. If omitted, the latest revision is suspended (see note in Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the `workset`.

Constraints

This command can be run at any lifecycle state either by a user with the role of *`PRODUCT-MANAGER`* 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.

SPV - Suspend Design Part Variant

```
<part-spec>
```

Example

```
SPV PROD:"RELEASE MANAGEMENT".IBM
```

- `<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 role of *PCMS-PART-MANAGER* for the selected design part. This design part must be in an *OPEN* state but not be referenced in an open change document.

SVBF - Set Version Branch Flags

```
<branch-id>
[/DESCRIPTION=<description>]
[/[NO]LOCK]
[/OWNER]
```

Example

```
SVBF MAINT/LOCK /OWNER=LOCAL
```

- `<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 ORACLE 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

This command can be run only by a user with the role of *`TOOL MANAGER`*.

SWF - Set Workset Filename

```
<item-spec>
[ /FILENAME=<filename> ]
/WS_FILENAME=<ws-filename>
[ /WORKSET=<workset-spec>]
```

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 workset. The new filename may include a directory path relative to the workset e.g. *src/foo.c*.

- `<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 workset will be affected.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *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 workset filename for the item in the workset.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  This optionally specifies the workset to be used for this command. If unspecified, the user's current workset will be taken.

NOTE A workset filename (*<ws_filename>*) can also be assigned to an item when it is created (see Dimensions command CI on page 57).

A particular Dimensions item can have different workset filenames in different worksets.

Constraints

Normally this command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the workset concerned.

This constraint can, however, be relaxed via the stand-alone utility *pcms_workset_perm* as described on page 300.

SWS - Set Workset Attributes

```
<workset-spec>
[/BRANCH|/TRUNK]
[/[NO]AUTO_REV]
[ /DESCRIPTION=<description>]
[ /VALID_BRANCHES=(<branch-id1>,<branch-id2>,...)
[ /DEFAULT_BRANCH=<branch-id>]
```

Examples

```
SWS PROD:"WS MAIN DVL" /BRANCH
SWS PROD:"WS MAINT DVL"
/VALID_BRANCHES=(maint,upgrade)
```

- `<workset-spec>` comprises:
  `<product-id>:<workset-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.
- `/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-spec is edited/updated.

- `/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.

- `/DESCRIPTION=<description>`

  optionally specify a new description to be attached to the workset definition, thus replacing the one which was assigned when the workset 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 workset. The list of valid branch-ids replaces the list (if any) specified previously for this workset using DWS or SWS.

  This list specifies the branches on which newly created item revisions can be placed.

  If the workset attributes are set to have *one or more* valid branches, every *new* item revision in the workset must use one of these branch-ids.

  If the workset attributes are set to have *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.

Description

The SWS command is used to set (or reset) the attributes of an existing workset created using the DWS command.

NOTE The */BRANCH*, */TRUNK*, */AUTO_REV* and */NOAUTO_REV* qualifiers may further be used to alter the options associated with the workset. The permitted combinations of these qualifiers are:

```
SWS <workset-spec> /BRANCH
SWS <workset-spec> /TRUNK
SWS <workset-spec> /AUTO_REV
SWS <workset-spec> /NOAUTO_REV
SWS <workset-spec> /BRANCH /AUTO_REV
SWS <workset-spec> /BRANCH /NOAUTO_REV
SWS <workset-spec> /TRUNK /AUTO_REV
SWS <workset-spec> /TRUNK /NOAUTO_REV
```

Constraints

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *WORKSET-MANAGER* for the specified workset.

UBLA - Update Baseline Attributes

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

Example

```
UBLA PROD:"R M VERSION 2 FOR HP" -
   /ATTRIBUTES=(TESTED_BY=GROUP5, AUTH_CODE=542)
```

- `<baseline-spec>` comprises:
  `<product-id>:<baseline-id>`
- `/ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, … )`

<attrN>	is the Variable Name defined for one of the 220 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.

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 functions (CBL, CMB and CRB) which 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 role of *PRODUCT-MANAGER*.

UC - Update Change Document

```
<chdoc-id>
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
             [ /DESCRIPTION=<desc-file>]
and / or     [ /ADD_DESCRIPTION]

or           [ /EDIT_ACTION_DESCRIPTIONS]
or           [ /CANCEL_EDIT]
```

Example

```
UC PROD_DR_28 -
/ATTRIB=(TITLE="QREL Subdir format problem")
```

- `<chdoc-id>`

  is the identity of the change document to be modified.

NOTE All the following qualifiers and parameters are optional, but **only** one of them must be specified.

The qualifiers */ATTRIBUTES*, */DESCRIPTION*, */ADD_DESCRIPTION*, */EDIT_ACTION_DESCRIPTIONS* and */CANCEL_EDIT* are mutually exclusive i.e. you cannot submit a UC command with more than one.

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

<attrN>	is the Variable Name defined for one of the 220 user-defined attributes for change documents, which has also been declared as usable for the product and type specified in *<chdoc-id>*.
<valueN>	is the new **substitution** value to be given to this attribute.

- `/DESCRIPTION=<desc-file>`

  specifies a file containing the text body to be used as:

  - the detailed description of the change document, 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 change document (if it is held) or an action description (if it is saved).

  NOTE Must *not* be specified if you wish to run UC from the Dimensions Agent for OS/390 or the *CMDCLIENT* interface.

- `/EDIT_ACTION_DESCRIPTIONS`

  calls an editor to allow a user with a leader role to edit all the action descriptions entered since the change document was last actioned.

  NOTE Must *not* be specified if you wish to run UC from the Dimensions Agent for OS/390 or the *CMDCLIENT* interface.

- `/CANCEL_EDIT`

  undoes the effects of a failed edit of a change document.

Constraints

This command can be run only by a user with the role of *CHANGE-MANAGER* or by users who have the minimum role to action the change document for the current state to the next state. Your edit is, however, also subject to any update rules set by a user with the role of *PRODUCT-MANAGER*.

UI - Update Item

```
<item-spec>
[ /FILENAME=<filename>]
[ /USER_FILENAME=<user-filename>]
[ /KEEP ]
[ /REVISION=<new-revision>]
[ /ATTRIBUTES=(<attr1>=<value1>,<attr2>=<value2>, ... ) ]
[ /CHANGE_DOC_IDS=(<cd1>,<cd2>,...) ]
[ /STATUS=<status>]
[ /COMMENT=<comment text>]
[ /WORKSET=<workset-spec>]
[ /FORCE_UPDATE ]
[ /CODEPAGE=<code-page>|DEFAULT ]
```

Example

```
UI PROD:"QUERY RELEASE".AAAA-SRC;1 /KEEP -
    /CHANGE=(PROD_DC_16, PROD_DR_8) /STAT="UNDER TEST" -
    /COMMENT="updated for ERB 58"
```

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

<item-id>	may be omitted if *<filename>* is specified.
<variant>	may be omitted if only one exists.
<revision>	defaults to the latest revision in the workset specified by */WORKSET*. If /WORKSET is unspecified, the user's default workset will be assumed.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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 *`<filename>`* – 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 workset; otherwise, the new revision will be placed in the user's current default workset.

  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 220 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=(<cd1>,<cd2>, … )`

<cdN>	identifies a change document 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 update of this item revision. The comment text can be up to 1978 characters long, and can be made available within the item header.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  and is optional.

  If specified, the new item revision will be placed in that workset.

  If unspecified, Dimensions will place the new item revision in the user's current workset.

- `/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 OS/390 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.

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

```
<item-spec>
[ /FILENAME=<filename>]
[ /ATTRIBUTES=(<attr1>=<value1>, <attr2>=<value2>, ... )]
[ /COMMENT=<comment text>]
[ /WORKSET=<workset-spec>]
[ /FORMAT=<format>]
[ /DESCRIPTION=<item-description>]
[ /ORIGINATOR=<Dimensions-user>]
```

Example

```
UIA PROD:"QUERY RELEASE".AAAA-SRC;1 -
/ATTRIB=(DELIVERY_DATE=10-JUN-1998, AUTH_CODE=542) -
/COMMENT="updated for ERB 58a"
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the file holding the item in the item-library.

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

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

<attrN>	is the Variable Name defined for one of the 220 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.

- `/COMMENT=<comment text>`

  comment text to explain the reason for the update of this item attributes. The comment text can be up to 1978 characters long.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

- `/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.

NOTE Only the PRODUCT-MANAGER can modify a FORMAT.

- `/DESCRIPTION=<item-description>`

  descriptive name for item

  NOTE Only 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.

  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 functions (CI, EDI, EI and UI) which assign values to user-defined item attributes, the */ATTRIBUTES* qualifier is shown as optional. However, it cannot be omitted if the function 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 – only be run in accordance with the attribute update rules set by a user with the role of *PRODUCT-MANAGER.* 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 be a Product Manager.
- 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 function. Values can be assigned to such attributes for a revision, only by the function which creates it (and only by the CI function 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.

ULCK - Unlock Workset

```
workset <workset-spec>
```

Example

```
ULCK WORKSET PROD_X:TEST_WS
```

- `<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  The specified workset must exist.

Description

This command unlocks the workset with *`workset`* as a fixed parameter and *`<workset-spec>`* a user-defined parameter. The unlocked state allows new Dimensions items to be added to the workset.

Constraints

This command can be run only by a user with the role of *`PRODUCT-MANAGER`* or *`WORKSET-MANAGER`* for the workset concerned.

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"
```

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

`<variant>`	may be omitted if only one exists.
`<pcs>`	is ignored. On completion of this function, 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>`

  is optional text describing the variant of the design part.

  If omitted, the description is copied from the current PCS.

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

`<attr1>`	is the Variable Name defined for one of the 220 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 been assigned the role of *`PCMS-PART-MANAGER`* for the design part to which the variant being updated belongs.

UPA - Update Part Attributes

```
<part-spec>
[/ATTRIBUTES=(<attr1>=<value1>, <attr2>=<value2>, ... )]
[ /DESCRIPTION=<part-description>]
```

Example

```
UPA PROD:"ITEM OPS".AAAA;5 -
    /ATTRIBUTES=(TESTED_BY=GROUP5, AUTH_CODE=542) -
    /DESCRIPTION= ”LIMITED TO GROUP 5 WITH CODE 542”
```

- `<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 220 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.

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 functions (CP, CPV and UP) which 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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *PCMS-PART-MANAGER*.

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"
```

- `<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

This command can be run only by a user who has the role of *`PARTS-CONTROLLER`* or *`PRODUCT-MANAGER`*.

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

URP - Unrelate Design Part

```
<part-spec>
/PARENT_PART=<parent-part-spec>
```
[1]

Example

```
URP PROD:"LIBRARY CONTROL".AAAA -
    /PARENT=PROD:"RELEASE MANAGEMENT".AABB
```

- `/PARENT_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

This command can be run only by a user with the role of *PRODUCT-MANAGER* or *PCMS-PART-MANAGER*.

1. The parameter */FATHER_PART=<father-part-spec>* will also continue to be supported for backward compatibility.

XCCD - Unrelate Change Documents from Change Document

```
<chdoc-id>
/CHANGE_DOC_IDS=(<cd1>,<cd2>,...)
[   /DEPENDENT
or  /INFO    ]
```

Example

```
XCCD PROD_CN_4 /CHANGE=PROD_DR_25
```

- `<chdoc-id>`

  is the identity of the change document which is the parent in the relationship to be removed.

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, … )`

<cdN>	is the identity of a change document 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 change documents. The two qualifiers are mutually exclusive.

  The default is */DEPENDENT*.

Constraints

This command can be run by users who have a role (any role will suffice) on the product or products owning the specific change document concerned. Such users must have the parent Change Document in their Pending List.

However, a user with the role of `PRODUCT-MANAGER` can set up the Process Model to specify that no change document relationships can be modified for certain change document types.

XCMD - Execute Dimensions Command File

```
<command-filename>
```

NOTE XCMD cannot be run from the Dimensions Agent for OS/390.

Example

```
XCMD items.setup
```

- `<command-filename>`

  specifies the name of a file containing commands which are to be executed (see Introduction on page 14 for more information).

XICD - Unrelate Item from Change Documents

```
<item-spec>
[ /FILENAME=<filename>]
/CHANGE_DOC_IDS=(<cd1>,<cd2>, ... )
[   /AFFECTED
or           /IN_RESPONSE_TO
or           /INFO]
[ /WORKSET=<workset-spec>]
```

Example

```
XICD PROD:"QUERY RELEASE".AAAA-SRC;1 -
    /CHANGE_DOC=(PROD_DR_25, PROD_DC_16)
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	defaults to the latest revision (see Introduction on page 23).

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`src/hello.c`*, *`hello.c`* or *`src/build/hello.c`*.

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

- `/CHANGE_DOC_IDS=(<cd1>,<cd2>, … )`

`<cdN>`	identifies a change document 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 change documents. The qualifiers are mutually exclusive.

  The default is *`/AFFECTED`*.

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Constraints

This command can be run only by users who have the change document in their pending list or by a user with the role of *`CHANGE-MANAGER`*.

XII - Unrelate Item from Item

```
<src-item-spec>
<dst-item-spec>
[/FILENAME=<src-filename>]
[/FILENAME1=<dst-filename>]
/RELATIONSHIP=<relationship-id>
[/WORKSET=<workset-spec>]
```

Example

```
XII "PROD_X:INTERFACE_C.AAAA-C;1" -
    "PROD_X:INTERFACE_H.AAAA-H;1" -
    /RELATIONSHIP="INCLUDES"
```

- `<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.

- `/FILENAME=<src-filename>`

  is an optional qualifier and further specifies the source item by giving its filename.

- `/FILENAME1=<dst-filename>`

  is an optional qualifier and further specifies the destination item by giving its filename.

- `/RELATIONSHIP=<relationship-id>`

  specifies the relationship type as defined by the DIR command

- `/WORKSET=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

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

  Item revisions to be affected by the command may be specified explicitly, or they will be selected from the workset.

Description

The XII command is used to remove instances of relationships created by the RII command (on page 209). The source and destination item types must be consistent with the relationship definition.

XIP - Unrelate Item from Part

```
<item-spec>
[ /FILENAME = <filename>]
/PART = <part-spec>
[ /WORKSET=<workset-spec>]
```

Example

```
XIP PROD:"QUERY RELEASE".AAAA-SRC -
    /PART= PROD:"RELEASE MANAGEMENT".IBM
```

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

`<item-id>`	may be omitted if *`<filename>`* is specified.
`<variant>`	may be omitted if only one exists.
`<revision>`	is ignored; all revisions are unrelated from the specified design part.

- `/FILENAME=<filename>`

  specifies the name of the workset filename.

  The workset filename identifies the relative pathname (directory plus filename) from the workset root directory to the item to be used from the current workset. The workset filename for the same item may **differ** between worksets e.g. *`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=<workset-spec>` comprises:
  `<product-id>:<workset-id>`

  this optionally specifies the workset to be used for this command: failing this, the user's current workset will be taken.

  All Item revisions, regardless of whether they are in the workset, will be affected.

Constraints

This command can be run only by users who have the item revision in their pending list or have the role of *`PRODUCT-MANAGER`*.

XPCD - Unrelate Part from Change Documents

```
<part-spec>
/CHANGE_DOC_IDS=(<cd1>,<cd2>,...)
```

Example

```
XPCD PROD:"RELEASE MANAGEMENT".AAAA -
    /CHANGE_DOC= PROD_DR_26
```

- `<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=(<cd1>,<cd2>, …`

`<cdN>`	is the identity of a change document from which the specified design part is to be unrelated.

Constraints

- This command can be run only by the user who has the change document in their Pending list or by a user with the role of *CHANGE-MANAGER* (a "change-manager")
- This command cannot be used with the secondary catalog list.
- This command cannot be run if the change document 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 change document back to its closed state.

3 Standalone Dimensions Utilities

In this Chapter

For this section…	See page…
Introduction	285
General Information	287
Action Change Documents by Date or Attribute Value: pcms_auto_action	289
Edit Detailed Description: pcms_cm_edit	291
Update Users' Pending Items Lists: pcms_item_pend	297
Send Reminders of Pending Lists: pcms_full_mail & pcms_incremental_mail	294
Update Users' Pending Items Lists: pcms_item_pend	297
Rename a Design Part within the Same Product: pcms_rename_part	298
Rename an Existing Product: pcms_rename_product	299
Set Dimensions Workset Permissions: pcms_workset_perm	300

For this section…	**See page…**
PRCS - RCS-like Front End for Dimensions	301
PSCCS - SCCS-like Front End to Dimensions	319

Introduction

This chapter identifies various miscellaneous Dimensions 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:

- **pcms_auto_action** This utility is used to action change documents once a date (attribute value) has passed.
- **pcms_cm_edit** (only for change-managers or users as described on page 291). This utility enables users to re-edit a change document's detailed description.
- **pcms cm_pend** (only for use by change-managers). This utility enables users' pending change document lists to be updated. It will also recalculate the current phase of the change document.
- **pcms_full_mail** and **pcms_incremental_mail** These utilities are used to send users periodic reminders of change documents in their pending lists.
- **pcms_item_pend** (only for use by product-managers). This utility enables user's pending item lists to be updated.
- **pcms_rename_part** (only for use by a product-manager or a user with the role *PCMS-PART-MANAGER* for the design parts). This utility is used to rename a design part within the same product.

- **pcms_rename_product** (only for use by product-managers). This utility is used to rename an existing product.
- **pcms_workset_perm** (only for use by workset-manager). This utility allows for a relaxation in the workset rules as to who can create a workset, or update its contents.
- **prcs** [Not Windows NT/2000 or the Dimensions Agent for OS/390] This utility provides a RCS-like front end to the version control commands of Dimensions.
- **psccs** [Not Windows NT/2000 or the Dimensions Agent for OS/390] 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 related document *Data Migration 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 related document *Data Migration Guide*.
- **upload** This utility performs an upload of a list of specified files in a user directory into Dimensions. See the related document *Data Migration Guide*.

General Information

All these utilities are run as independent programs from the operating system prompt (or from a command script file). They 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 *PCMS_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 change documents 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.

Action Change Documents by Date or Attribute Value: pcms_auto_action

This utility is available to all users and supports two different syntaxes:

Syntax (1st form)

```
pcms_auto_action <product-id> <chdoc-type> <holding-state>
   <preferred-state> <fallback-state> <date-attribute-name>
```

Example (1st form)

```
pcms_auto_action PCMS CR HELD “CCB PENDING” \
   OUTSTANDING HOLD_UNTIL
```

This form of the utility actions each change document in <*`product-id`*> of type <*`chdoc-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 change document attribute whose Data-Type is Date for date format (for details, refer to the Process Modeler Object Type Definitions | Attributes | Edit dialog in *Process Modeling User’s Guide*).

The action will be to <*`holding-state`*> provided that this will allow the change document 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 change documents that meet the specified criteria are left at <*`holding-state`*>).

Syntax (2nd form)

```
pcms_auto_action <product-id> <chdoc-type> <chdoc-status>
   <preferred-state> <fallback-state>
   <attribute-name>=<value>
```

Example (2nd form)

```
pcms_auto_action PCMS PR IN_PROGRESS CLOSED \
   COMPLETED WORK_STATUS=COMPLETED
```

In this form of the utility, if any change documents of type *<change-type>* with status *<chdoc-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 change documents are actioned to the preferred state *<preferred-state>*, otherwise they are actioned to the fallback state *<fallback-state>*.

Edit Detailed Description: pcms_cm_edit

This utility is available to:

- A change-manager.
- A user who as a minimum has a role to action the change document for the current state to a next state.
- A user whose role at a specific point in the change document's lifecycle satisfies attribute update rules for the *PCMS_CHDOC_DETAIL_DESC* special system attribute, as setup by the product-manager using the Process Modeler. This allows such users to edit the detailed description at that designated point in the change document's lifecycle. If an update rule has not been explicitly set up, the default is *$ORIGINATOR* role at the *$TO_BE_DEFINED* state.

The utility allows the editing of the detailed description of a change document.

Syntax

```
pcms_cm_edit <chdoc-id>
```

<chdoc-id> is the identity of the change document whose detailed description is to be edited. It must still be open (i.e. not at a final lifecycle state).

An interactive editor is invoked. In UNIX this is as specified by the setting of the symbol *PCMS_CHD_EDT* or *PCMS_CHD_EDT_SCRIPT*.

Update Users' Pending Lists: pcms_cm_pend

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 which could mean a change to the people responsible for change documents, a utility (*pcms_cm_pend*) has been provided that allows the change-manager (only) to re-calculate the pending trays for users' change documents.

This utility also needs to be run whenever rules are enabled (Process Modeler Object Type Definitions | CM Rules dialog, see *Process Modeling User's Guide*) for one or more of the product's change document types, if at the time any change documents of these types already exist.

Syntax (1st form)

```
pcms_cm_pend -c <chdoc-id1> <chdoc-id2> ...
```

-c <chdoc-idN> is one of a list of change document identifiers separated by spaces. Pending lists are recalculated only for these specified change documents.

Syntax (2nd form)

```
pcms_cm_pend -p <product-id> [ -t <chdoc-type> ]
   [ -s <status> ] [ -u <user> ] [ -r <role> ]
```

If the syntax is of the second form, the utility processes every change document 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 *%*

-p <product-id> is a string to be matched by the product-id of each change document to be processed. This parameter is not optional, so just specify *-p %* if the processing is not to be limited by the value of product-id.

-t <chdoc-type>	is a string to be matched by the type of each change document to be processed.
-s <status>	is a string to be matched by the current status (lifecycle state) of each change document to be processed.
-u <user>	is a string to be matched by login usernames. Each change document is not processed unless it is currently in the pending list of at least one matching username.
-r <role>	is a string to be matched by role-titles of each change document to be processed. Each change document 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.

NOTE Whenever this utility is to process more than just a few change documents, it is highly recommended that it is executed only at times when database activity is otherwise light.

Send Reminders of Pending Lists: pcms_full_mail & pcms_incremental_mail

Two utilities, available to all users, are provided to mail users with details on their pending change documents:

- **full mail**, which lists all relevant change documents pending for each user; and
- **incremental mail**, which lists only relevant change documents last actioned (or, optionally, last modified) within the preceding few days.

Syntax

```
pcms_full_mail [-v] <product-id> <chdoc-type>

pcms_incremental_mail [-u] [-v] <product-id> <chdoc-type>
   <days>
```

-v	specifies a verbose format for each change document in the mail message (see below for details). The default is a brief format.
-u	is valid only for *pcms_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 change document falls within the designated period.
<product-id>	is the identity of the product whose change documents are to be reported. A null string may be used to specify all products in the database.
<chdoc-type>	is the change document type for the change documents to be reported. A null string may be used to specify all change document types.
<days>	is the number of days to be covered by the messages generated by *pcms_incremental_mail*. This period ends at midnight immediately before pcms_incremental_mail is started. For example, if three days are specified here, and pcms_incremental_mail is started at 2 a.m. on Monday, change

documents last actioned at 1 a.m. the previous Friday or at 11 p.m. on Sunday will be reported, while change documents 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 change documents 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 change document in both brief and verbose formats:

Change Document Identity

Current Status

Title (i.e. Attr-no 1): the first 70 characters

The following are given for each change document in verbose format only:

The relevant user roles

Related Design Parts

Related Change Documents and their Current Status

Originator

These utilities are intended to provide regular reports to users on change documents 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 *pcms_full_mail* will run at 3 a.m. on the first day of every month, while *pcms_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 change documents 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
pcms_incremental_mail $1 $2 $3
exit
```

Similar *crontab* entries can be made to run *pcms_auto_action*.

Update Users’ Pending Items Lists: pcms_item_pend

In the event that a user leaves, or design parts are moved around in the structure which could mean a change to the people responsible for items/files (hereinafter referred to as items for brevity), a utility (*pcms_item_pend*) has been provided that allows the product-manager (only) to re-calculate the pending trays for users’ items.

Syntax

```
pcms_item_pend -p <product-id> [ -t <item-type> ]
   [ -u <user_name> ] [ -s <status> ]
```

The utility will process every item which 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.

-p <product-id>	is a string to be matched by the product-id of each item-id of each item to be processed.
-t <item-type>	is a string to be matched by the type of each item to be processed.
-s <status>	is a string to be matched by the current status (lifecycle state) of each item to be processed.
-u <user_name>	is a string to be matched by login usernames. Each item is not processed unless it is currently in the pending list of at least one matching username.

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.

Rename a Design Part within the Same Product: pcms_rename_part

This utility is only available to a product-manager or a user with the role *PCMS_PART_MANAGER* for the design parts. It enables an existing part to be renamed within the same product.

Syntax

```
pcms_rename_part <ExistingPart> <NewPart> <Product>
```

Rename an Existing Product: pcms_rename_product

This utility is only available to product-managers. It enables an existing product to be renamed.

Syntax

```
pcms_rename_product <ExistingProduct> <NewProduct>
```

Set Dimensions Workset Permissions: pcms_workset_perm

This utility is only available to a Workset Manager. It allows for a relaxation in the workset rules as to who can create a workset, or update its contents.

Syntax

UNIX

-
  ```
  $PCMS_DBASE/pcms_workset_perm <base_database>/<password>
     <product_id> [0-3]
  ```

Windows

-
  ```
  X:\PVCS\Dimensions\x.y\dbase\pcms_workset_perm.bat
     <base_database>/<password> <product_id> [0-3]
  ```

  where `X:` is the drive upon which Dimensions is installed, and `x.y` is the version of Dimensions installed, for example, `7.1`.

<product-id> is the product to update or display and [0-3] determines the degree of relaxation:

0 =	Normal (default) Dimensions workset rules.
1 =	Any user with a role on the product may create a workset.
2 =	Any user with a role on the product may update workset contents (i.e. create workset directories etc).
3 =	Both 1 and 2.

PRCS - RCS-like Front End for Dimensions

```
prcs subcommand [ option ... ] [ filename ... ]
   [ Dimensions option ... ]
```

NOTE prcs cannot be run from the Dimensions Agent for OS/390.

Examples

1 To create a Dimensions item for file `program.c`, ensure that the desired workset is set and that you are in a sub-directory within the scope of the workset root directory, 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   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 workset directory held in Dimensions.

```
example% prcs grep printf PROD
```

6 To run *wc* on all files in the current workset directory held in Dimensions.

```
example% prcs eval wc -- PROD
```

(See page 312 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 316.

- `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 filename 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 workset filename will consist of a workset directory that matches the difference between the current working directory and workset root directory and the filename specified to the create command.

The *+F* Dimensions specific option must be used to specify the format if the filename 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 change documents 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 304) 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 filename, 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 302). 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 filename 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 workset. The *-l* and *-R* options cannot be combined.

l	Appends latest revision to filename.
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 workset. 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 workset. 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 302).

- `+V`

  Displays the Dimensions commands and messages generated by the `prcs` operation. The workset selected by `prcs` will also be displayed.

- `+W <Workset>`

  Defines the Dimensions workset to be used for the *prcs* command. This option is required if `prcs` is unable to determine the workset from the current directory. The workset root directory is set to the current working directory unless a *+D* option is also specified.

- `+D <Workset Root Directory>`

  Defines the workset root directory to be used for the workset given in the *+W* Dimensions Option. This cannot be specified without the *+W* option.

- `+C <Change Document List>`

  This option allows a list of change documents 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 filename(s) specified the `prcs` command and the associated Dimensions ‘item’ is derived from the filename by searching for a matching workset filename in the current workset.

The *prcs* command expects these ‘items’ to reside in a workset directory that when appended to the current workset root equates to the current working directory. Thus, when you invoke `prcs` from directory "*/usr/work*" which is marked as being the workset root directory for workset *WS1* with a filename argument, the subcommand will be applied to a Dimensions item in the root directory of workset *WS1* with workset filename equivalent to that specified to the command. If the command is issued in a subdirectory of "*/usr/work*", then the filename is appended to the offset from the workset root directory to the current directory and this name is used to find the ‘item'. If the `filename` 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 workset and which match the wildcard. The wildcard is in standard Dimensions format. Thus executing the *prcs* command in the workset root directory *prcs co program.c* would apply the *co* (check out) subcommand to an ‘item' with workset filename *program.c*. However, executing the same `prcs` command in the workset directory "*src*" would apply the *co* subcommand to an item with workset filename *src/program.c*, while the command *prcs co PCMS* executed in a directory "*/usr/work/src*" in workset *WS1* with root directory "*/usr/work/*" would apply the `co` subcommand to every item in Dimensions with the workset directory "*src*".

prcs uses the same mechanism as Dimensions Make to determine the correct workset for operation. If it is unable to determine a workset, then a diagnostic message is issued and the *+W* and *+D* Dimensions-specific options must be used (see Dimensions Options on page 310).

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 workset 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 workset 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

`$PCMS_PROG/prcs`	prcs program
`$PCMS_PROG/pid`	daemon
`$PCMS_PROG/pid_kill`	terminate daemon
`$PCMS_ROOT/man/PS/prcs.ps`	PostScript version of the UNIX manual page

Associated UNIX Documentation

- diff(1), grep(1), co(1), ci(1), rcs(1), rlog(1), rcsdiff(1),
- Programming Utilities and Libraries

Constraints

This command is available to all users who have the roles to perform the associated Dimensions operations but not from Windows NT/2000 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.

Notes

prcs is implemented using a daemon `pid` which is automatically started by *prcs* on its first invocation. This provides faster response for *prcs* commands. It will have the same Dimensions environment as the shell from which `prcs` was invoked. The daemon will disconnect from Dimensions after 30 minutes inactivity and will reconnect when the next *prcs* command is issued.

The daemon can be terminated using the *pid_kill* program.

Changes to the Dimensions process model while there is a currently executing daemon will not be applied to *prcs* operations. The daemon must be killed using *pid_kill* and restarted by issuing a `prcs` command. The *BRANCH/TRUNK* settings for Dimensions worksets are ignored by *prcs* which uses its own revision calculation.

There is a conflict between the Forced Revision Generation flag on Dimensions worksets and the new revisions generated by *prcs*. Do not set this flag if the workset 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. `prcs` can be set up to respond to the individual commands by executing the shell script *make_prcs_links.sh*. This should be run as root with a Dimensions environment set. Symbolic links bearing the names of the common RCS commands will be created in the *$PCMS_PROG* directory. Alternatively, 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 the Dimensions Agent for OS/390.

Examples

1 To create a Dimensions item for file *program.c,* ensure that the desired workset is set and that you are in a subdirectory within the scope of the workset root directory. Then use *psccs create*:

```
example% psccs create program.c
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% sccs 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 workset directory held in Dimensions.

```
example% psccs grep printf PROD
```

6 To run *wc* on all files in the current workset directory held in Dimensions.

```
example% psccs eval wc -- PROD
```

(See page 327 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 331 for non-supported SCCS options.)

- `check [-b] [-u[`*`username`*`]]`

  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[username]	Only checks files being edited by you. When *username* 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 filename 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 workset filename consists of a workset directory that matches the difference between the current working directory and workset root directory and the filename specified to the create command.
  - The *+F* Dimensions specific option must be used to specify the format if the filename 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] [-c`*`date-time`*`] [-r`*`revision`*`] 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`.
-c*date-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] [-c`*`date-time`*`] [-r`*`revision`*`] 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.
-c*date-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[`*`username`*`]]`

  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[username]	Only lists files checked out by you. When *`username`* 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] [-c`*`date-time`*`] [-r`*`revision`*`] 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`*.
-c*date-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]`*
-r*revision*	Specifies a given revision

- `sccsdiff -r`*`old-revision`*` -r`*`new-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[`*`username`*`]]`

  Displays the list of files that are currently checked out, one filename per line.

-b	Ignores branches.
-u[username]	Only lists files checked-out to you. When *`username`* 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 workset. The *-l* and *-R* options cannot be combined.

-l	Appends revision to filename.
-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 workset. 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 workset. 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 *filename* 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 320.)

- `+V`

  Displays the Dimensions commands and messages generated by the *`psccs`* operation. The workset selected by *`psccs`* will also be displayed.

- `+W <Workset>`

  Defines the Dimensions workset to be used for the *`psccs`* command. This option is required if `psccs` is unable to determine the workset from the current directory.

- `+D <Workset Root Directory>`

  Defines the workset root directory to be used for the workset given in the *`+W`* Dimensions Option.

- `+C <Change Document List>`

  This option allows a list of change documents 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 filename(s) specified to the *psccs* command and the associated Dimensions item is derived from the filename by searching for a matching workset filename in the current workset.

The *psccs* command expects these '`items`' to reside in a workset directory that when appended to the current workset root equates to the current working directory. Thus, when you invoke `psccs` from directory "*/usr/work*" which is marked as being the workset root directory for workset *WS1* with a *filename* argument, the subcommand will be applied to an item in the root directory of workset *WS1* with workset filename equivalent to that specified to the command. If the command is issued in a sub-directory of "*/usr/work*", then the *filename* is appended to the offset from the workset root directory to the current directory and this name is used to find the Dimensions item.

If the *filename* 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 workset and which match the wildcard. The wildcard is in standard Dimensions format. Thus, executing the *psccs* command in the workset root directory *psccs get program.c* would apply the `get` subcommand to an item with workset filename *program.c*

However, executing the same *psccs* command in the workset directory *"src"* would apply the `get` subcommand to an 'item' with workset filename *src/program.c* while the command `psccs get PCMS` executed in a directory "*/usr/work/src*" in workset *WS1* with root directory "*/usr/work/*" would apply the `get` command to every item in Dimensions with the workset directory *"src"*.

psccs uses the same mechanism as Dimensions Make to determine the correct workset for operation. If it is unable to determine a workset, then a diagnostic message is issued and the *+W* and *+D* Dimensions specific options (see Dimensions Options on page 325) 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

$PCMS_PROG/psccs	psccs program
$PCMS_PROG/pid	psccs daemon
$PCMS_PROG/pid_kill	terminate psccs daemon
$PCMS_ROOT/man/PS/psccs.ps	PostScript version of the PSCCS UNIX manual page

Associated UNIX Documentation

diff(1), grep(1), sccs-admin(1), sccs-delta(1), sccs-get(1), sccs-prs(1), sccs-sact(1), sccs-sccsdiff(1), sccs-unget(1),

Programming Utilities and Libraries

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.

NOTE *psccs* is implemented using a *daemon pid* which is automatically started by *psccs* on its first invocation. This provides faster response for *psccs* commands. It will have the same Dimensions environment as the shell from which *psccs* was invoked. The daemon will disconnect from Dimensions after 30 minutes inactivity and will reconnect when the next *psccs* command is issued.

The daemon can be terminated using the *pid_kill* program.