

HP Project and Portfolio Management Center

Software Version: 7.1

Commands, Tokens, and Validations Guide and Reference

Document Release Date: March 2007

Software Release Date: March 2007



Legal Notices

This document, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332; 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

U.S. GOVERNMENT RESTRICTED RIGHTS. This Software Documentation is a “commercial item” as defined at 48 C.F.R. 2.101 (October 1995). In accordance with 48 C.F.R. 12.212 (October 1995), 48 C.F.R. 27.401 through 27.404 and 52.227-14 (June 1987, as amended) and 48 C.F.R. 227.7201 through 227.7204 (June 1995), and any similar provisions in the supplements to Title 48 of the C.F.R. (the “Federal Acquisition Regulation”) of other entities of the U.S. Government, as applicable, all U.S. Government users acquire and may use this Documentation only in accordance with the restricted rights set forth in the license agreement applicable to the Computer Software to which this Documentation relates.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

All other company, brand and product names may be trademarks or registered trademarks of their respective holders. Mercury disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Intel®, Intel® Itanium®, Intel® Xeon™, and Pentium® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, and Windows® XP are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

Mercury provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. Mercury makes no representations or warranties whatsoever as to site content or availability.

© 1997- 2007 Mercury Interactive Corporation. All rights reserved.

Documentation Updates

This manual's title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to: http://ovweb.external.hp.com/lpe/doc_serv/.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Mercury Product Support

You can obtain support information for products formerly produced by Mercury as follows:

- If you work with an HP Software Services Integrator (SVI) partner (www.hp.com/managementsoftware/svi_partner_list), contact your SVI agent.
- If you have an active HP Software support contract, visit the HP Software Support site and use the Self-Solve Knowledge Search to find answers to technical questions.
- For the latest information about support processes and tools available for products formerly produced by Mercury, we encourage you to visit the HP-Mercury Software Support web site at: support.mercury.com.
- Contact your HP Sales Representative if you have additional questions.

HP Software Support

You can visit the HP Software Support web site at www.hp.com/managementsoftware/services.

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To find more information about access levels, go to: www.hp.com/managementsoftware/access_level.

To register for an HP Passport ID, go to: www.managementsoftware.hp.com/passport-registration.html.

Table of Contents

List of Figures	xi
List of Tables	xiii
Chapter 1: Getting Started with Commands, Tokens, and Validations	17
Introduction to Commands, Tokens, and Validations	18
Related Documents.....	19
Chapter 2: Using Commands	21
About Commands.....	22
Object Type Commands and Workflows	22
Request Type Commands and Workflows.....	23
Special Commands	24
Command Language	24
Command Conditions.....	25
About the Commands Tab	26
Configuring Commands	27
Examples of Command Uses.....	30
Chapter 3: Using Special Commands	33
About Special Commands	34
Special Command Parameters.....	34
Special Command Language.....	35
Special Command Conditions	35
Using the PPM Workbench to List Special Commands	36
About the Special Command Builder.....	37
Configuring Special Commands.....	37
Using Special Commands.....	42
Using the Special Command Builder	43
Nesting Special Commands	44
Using the Special Command Details Report to List Special Commands	44
Examples of Using Special Commands	45
Chapter 4: Using Tokens	47
About Tokens	48
Where to Use Tokens.....	48
Token Evaluation.....	49
About the Token Builder.....	50

Token Formats	51
Default Format	54
Explicit Entity Format	55
Nesting Explicit Entity Tokens within Other Tokens.....	56
User Data Format.....	57
Parameter Format.....	58
Request Field Tokens.....	59
Request Field Token Prefixes	59
Tokens in Request Table Components	59
Sub-Entity Format.....	62
Environment and Environment Application Tokens	63
Using the Token Builder	65
Chapter 5: Using Validations	67
About Validations.....	68
Validation Component Types.....	69
Accessing Validations Through Packages and Requests	71
Validations and Special Characters	72
Viewing System Validations	73
Configuring Validations	74
Configuring Static List Validations	76
Configuring Dynamic List Validations	78
Configuring SQL Validations.....	79
SQL Validation Tips.....	80
Command Validations	81
Configuring Short List Auto-Complete Field Validations.....	82
Configuring Long List Auto-Complete Field Validations.....	83
Configuring Automatic Value Matching and Interactive Select Pages	85
An Overview of Matching for “Starts with” or “Contains”	85
Configuration Tips	87
Adding Search Fields to Long List Auto-Complete Validations	88
Configuring the Filter Field Layout.....	91
Configuring an Auto-Complete List of Users (Special Case).....	92
Configuring the Auto-Complete Values.....	93
Configuring Validations by Commands With Delimited Output	94
Configuring Validations by Commands with Fixed-Width Output	96
Configuring User-Defined Multi-Select Auto-Complete Fields.....	98
Example of Token Evaluation and Validation by Command with Delimited Output	100
Configuring Text Field Validations.....	103
Text Data Masks for Validations	104
Configuring the Numeric Data Mask.....	105
Configuring the Currency Data Mask.....	107
Configuring the Percentage Data Mask	109
Configuring the Telephone Data Mask	110

Configuring a Custom Data Mask.....	112
Configuring Directory Chooser Validations	113
Configuring File Chooser Validations	114
Configuring Date Field Validations.....	116
Configuring 1800 Character Text Areas	117
Configuring the Table Component	118
Configuring Table Components	119
Configuring Table Rules.....	121
Example of Using a Table Component on an Order Form	122
Example of Setting Unit Prices.....	124
Example of Calculating Totals.....	125
Using Table Components.....	126
Using Tokens in Table Components	126
Calculating Column Totals	126
Appendix A: Tokens	129
Overview of Tokens.....	131
Application Server Tokens	131
Budget Tokens	131
Contact Tokens	133
Distribution Tokens.....	134
Document Management Tokens.....	135
Environment Tokens	136
Environment > Dest Env Tokens	136
Environment > Dest Env > App Tokens	139
Environment > Dest Env > Env Tokens	141
Environment > Env Tokens.....	145
Environment > Env > App Tokens	147
Environment > Env > Env Tokens	150
Environment > Source Env Tokens.....	153
Environment > Source Env > App Tokens	156
Environment > Source Env > Env Tokens	158
Command Tokens	162
Financial Benefit Tokens	163
Notification Tokens	164
Organization Unit Tokens.....	164
Package Tokens.....	166
Package > Package Line Tokens	168
Package > Pending Reference Tokens	169
Package Line Tokens	171

Program Tokens.....	171
Project Tokens.....	172
Project Detail Tokens	176
Release Tokens	176
Release > Distribution Tokens	177
Request Tokens	178
Request > Pending Reference Tokens.....	182
Request > Field Tokens.....	183
Request Detail Tokens.....	183
Request Detail > Field Tokens	184
Resource Pool Tokens	184
Security Group Tokens	185
Skill Tokens	185
Staffing Profile Tokens	186
Step TXN (Transaction) Tokens	187
System Tokens	188
Task Tokens.....	189
Tasks > Pending Tokens	191
Time Management Notification Tokens	193
User Tokens.....	193
Validation Tokens	195
Validation > Value Tokens	196
Workflow Tokens.....	197
Workflow > Workflow Step Tokens.....	198
Workflow Step Tokens.....	200
Request > Field Tokens	203
CMBD Application Tokens	203
Demand Management SLA Tokens	204
Demand Management Scheduling Tokens	204
MAM Impact Analysis Tokens	204
Portfolio Management Asset Tokens.....	204
Portfolio Management Project Tokens.....	205
Portfolio Management Proposal Tokens.....	206
Program Issue Tokens	207
Program Reference Tokens	207
Project Issue Tokens	207
Project Reference Tokens.....	207
Project Risk Tokens	208
Project Scope Change Tokens.....	208
Quality Center Defect Information Tokens.....	208

Quality Center Information Tokens.....	209
Resource Management Work Item Tokens	209
Index	211

List of Figures

Figure 2-1	Commands tab	26
Figure 3-1	Special Command Builder.....	37
Figure 3-2	RCS File Migration object type	42
Figure 4-1	Example of a token used in a SQL statement	49
Figure 4-2	Token Builder window	50
Figure 4-3	Table component formats.....	60
Figure 5-1	Auto-complete using command validation	81
Figure 5-2	Short list auto-complete.....	82
Figure 5-3	Long list auto-complete	84
Figure 5-4	Auto-complete field and matching values on the Select page	85
Figure 5-5	Filter fields in the auto-complete select window	88
Figure 5-6	Auto-complete list.....	94
Figure 5-7	Validation by command with delimited output	95
Figure 5-8	Validation by command with fixed width output	97
Figure 5-9	Validation window for the numeric data mask.....	106
Figure 5-10	Validation window for the currency data mask	108
Figure 5-11	Validation window for the percentage data mask	109
Figure 5-12	Validation window for the telephone data mask	111
Figure 5-13	Validation window for the custom data mask	112
Figure 5-14	Validation window for static environment override in file chooser.....	114
Figure 5-15	Validation window for token-based environment override in file chooser	115
Figure 5-16	Hardware information window.....	118
Figure 5-17	Rules window accessed from the Rules tab	122
Figure 5-18	Validations window.....	123
Figure 5-19	Rules window.....	125
Figure 5-20	Hardware information window.....	126
Figure 5-21	Sample validation for a Simple Order table component.....	127
Figure 5-22	Sample table component displaying a column total.....	128

List of Tables

Table 2-1	Example conditions	25
Table 3-1	Example conditions	36
Table 4-1	Entities	52
Table 4-2	Sample environment and application attributes.....	64
Table 4-3	Sample environment tokens.....	64
Table 5-1	Component types	69
Table 5-2	Column headers.....	79
Table 5-3	Automatic character matching field behavior.....	86
Table 5-4	Automatic character matching Select page behavior	86
Table 5-5	Fields in the Fields: New window.....	90
Table 5-6	Validation by command with delimited output	96
Table 5-7	Column headers.....	96
Table 5-8	Validation by command with fixed width output	97
Table 5-9	Column headers.....	98
Table 5-10	Data mask formats	104
Table 5-11	Fields for configuring the numeric data mask for text fields	106
Table 5-12	Fields configuring the currency data mask for text fields	108
Table 5-13	Fields configuring the percentage data mask for text fields	110
Table 5-14	Fields configuring the telephone data mask for text fields.....	111
Table 5-15	Sample telephone data mask formats.....	111
Table 5-16	Sample custom data mask formats.....	113
Table 5-17	File chooser field.....	114
Table 5-18	Static environment override.....	115
Table 5-19	Token-based environment override	116
Table 5-20	Date field formats.....	117
Table 5-21	Example, table component validation settings	123
Table 5-22	Example - Set Unit Price rule settings.....	124
Table 5-23	Example - Calculate Total rule settings	125
Table A-1	Application server tokens.....	131
Table A-2	Budget tokens	131
Table A-3	Contact tokens	133

Table A-4	Distribution tokens.....	134
Table A-5	Document Management tokens.....	135
Table A-6	Environment > Dest Env tokens	136
Table A-7	Environment > Dest Env > App tokens.....	139
Table A-8	Environment > Dest Env > Env tokens	141
Table A-9	Environment > Env tokens	145
Table A-10	Environment > Env > App tokens	147
Table A-11	Environment > Env > Env tokens	150
Table A-12	Environment > Source Env tokens	153
Table A-13	Environment > Source Env > App tokens.....	156
Table A-14	Environment Source Env > Env tokens	158
Table A-15	Command tokens	162
Table A-16	Financial Benefit tokens	163
Table A-17	Notification tokens	164
Table A-18	Organization Unit tokens.....	164
Table A-19	Package tokens	166
Table A-20	Package > Package Line tokens	168
Table A-21	Package > Pending Reference tokens	169
Table A-22	Package Line tokens	171
Table A-23	Program tokens	171
Table A-24	Project tokens.....	172
Table A-25	Project Detail tokens	176
Table A-26	Release tokens	176
Table A-27	Release > Distribution tokens.....	177
Table A-28	Request tokens	178
Table A-29	Request > Pending Reference tokens	182
Table A-30	Request Detail tokens	183
Table A-31	Resource Pool tokens	184
Table A-32	Security Group tokens	185
Table A-33	Skill tokens	185
Table A-34	Staffing Profile tokens	186
Table A-35	Step TXN (Transaction) tokens.....	187
Table A-36	System tokens	188
Table A-37	Tasks tokens.....	189

Table A-38	Tasks > Pending tokens	191
Table A-39	Time Management Notification tokens	193
Table A-40	User tokens	193
Table A-41	Validation tokens	195
Table A-42	Validation > Value tokens	196
Table A-43	Workflow tokens	197
Table A-44	Workflow > Workflow Step tokens	198
Table A-45	Workflow Step tokens	200
Table A-46	CMBD Application tokens	203
Table A-47	Demand Management SLA tokens	204
Table A-48	Demand Management Scheduling tokens	204
Table A-49	MAM Impact Analysis tokens	204
Table A-50	Portfolio Management Asset tokens	204
Table A-51	Portfolio Management Project tokens	205
Table A-52	Portfolio Management Proposal tokens	206
Table A-53	Program Reference tokens	207
Table A-54	Project Issue tokens	207
Table A-55	Project Issue tokens	207
Table A-56	Project Issue tokens	208
Table A-57	Project Scope Change tokens	208
Table A-58	Quality Center Defect Information tokens	208
Table A-59	Quality Center Information tokens	209
Table A-60	Resource Management Work Item tokens	209

1 Getting Started with Commands, Tokens, and Validations

In This Chapter:

- *Introduction to Commands, Tokens, and Validations*
 - *Related Documents*
-

Introduction to Commands, Tokens, and Validations

Commands, tokens, and validations are used throughout HP Project and Portfolio Management Center to enable advanced automation and defaulting.

Commands are at the heart of the execution layer within the deployment system. They determine which actions are executed at specific workflow steps. Actions performed at a workflow step can include file migration, script execution, data analysis, or code compilation. [Chapter 2, *Using Commands*, on page 21](#) provides an overview of commands, and examples of how to use them.

Special commands are commands with variable parameters and are used in object types, request types, report types, workflows, and validation command steps. (Workflows use special commands in their workflow step sources.) These command steps perform a variety of functions, such as copying files between environments and establishing connections to environments for remote command execution. [Chapter 3, *Using Special Commands*, on page 33](#) contains information about how to create, edit, and use special commands in PPM Center.

Tokens are variables that PPM Center entities use to reference information that is undefined until the entity is used in a specific context. For example, entities use tokens to set variables in commands or within notifications to specify recipients.

Field validations determine the field types (for example, a text field or drop-down list) and the values the field can accept. Workflow step validation controls the possible results of exiting steps. PPM Center uses two types of tokens: standard tokens and custom tokens. [Chapter 4, *Using Tokens*, on page 47](#) shows how to use tokens.

Validations determine the valid input values for user-defined fields, such as object type or request type fields. Validations also determine the possible results that a workflow step can return. Validations are used for the field component type and workflow step results. [Chapter 5, *Using Validations*, on page 67](#) provides detailed information on how to use tokens.



To access the user interface components described in this document, you must be granted the Configuration license.

Related Documents

The following documents also include information related to using commands, tokens, and validations:

- *HP Demand Management Configuration Guide*
- *HP Deployment Management Configuration Guide*

2 Using Commands

In This Chapter:

- *About Commands*
 - *Object Type Commands and Workflows*
 - *Request Type Commands and Workflows*
 - *Special Commands*
 - *Command Language*
 - *Command Conditions*
 - *About the Commands Tab*
 - *Configuring Commands*
 - *Examples of Command Uses*
-

About Commands

Commands are at the heart of the execution layer within PPM Center. They determine which actions are executed at specific workflow steps. Actions performed at workflow steps can include file migration, script execution, data analysis, field behavior, or code compilation. The following PPM Center entities use commands:

- Object types
- Request types
- Report types
- Validations
- Workflow step sources
- Special commands

Object Type Commands and Workflows

Object type commands are tightly integrated with the workflow engine. The commands in an object type are executed at execution workflow steps in HP Deployment Management package lines.

Keep in mind the following concepts regarding command and workflow interaction:

- To execute object type commands at a given workflow step, configure the workflow step as follows:
 - The workflow step must be an execution type step.
 - Specify the following parameter values:
 - Workflow Scope = Packages
 - Execution Type = Built-in Workflow Event
 - Workflow Command = `execute_object_commands`
- When the object reaches the workflow step (Workflow Command = `execute_object_commands`), all object type commands with conditions satisfied are run in the order in which they are listed on the command field for the object type.

- You can configure the object type to run only certain commands at a given step. To do this, specify command conditions. For information about how to specify command conditions, see [Command Conditions](#) on page 25.

Request Type Commands and Workflows

Like object type commands, request type commands define the execution layer within HP Demand Management. While most of the resolution process for a request is analytically based, cases may arise for specific request types for which system changes are required. In such cases, you can use request type commands to make these changes automatically.

Request type commands are tightly integrated with the workflow engine. The commands in a request type are executed at execution workflow steps. Keep in mind the following concepts regarding the interactions between command and workflow:

- To execute request type commands at a given workflow step, configure the workflow step as follows:
 - The workflow step must be an execution type step
 - Set the following parameter values:
 - Workflow Scope = Requests
 - Execution Type = Built-in Workflow Event
 - Workflow Command = `execute_request_commands`
- When the request reaches the workflow step (Workflow Command = `execute_request_commands`), all commands with all conditions satisfied are run in the listed order in which they are listed on the command field for the request type.
- To set up command conditions so that the request type runs only certain commands at a given step, specify command conditions. For information about how to specify command conditions, see [Command Conditions](#) on page 25.

Special Commands

Object types, request types, report types, workflows and validations all use commands to access the execution layer. To simplify the use of command executions, PPM Center provides a predefined set of special commands.

Special commands are commands with variable parameters, and are used in object type, request type, report type, workflow, and validation command steps. These command steps perform a variety of functions, such as copying files between environments and establishing connections to environments for remote command execution.

PPM Center features the following two types of special commands:

- System special commands are shipped with PPM Center. System special commands are read-only and have the naming convention `ksc_command_name`.
- User-defined special commands have the naming convention `sc_command_name`.

Special commands act as modules that you can reuse. It is often more efficient to create a special command for a program that you can reuse than to place an individual command into every object type or request type that requires it.



For more information about special commands, see [Chapter 3, Using Special Commands](#), on page 33.

Command Language

The command steps in a command define the system-level executions that must be performed to realize the command function. Command steps can be UNIX® commands, third-party application commands, or special commands. Special commands are reusable routines defined in PPM Center.

PPM Center also supplies several system special commands that you can use to perform common events (such as connecting to environments or copying files).



For more information about special commands, see [Chapter 3, Using Special Commands](#), on page 33.

Command Conditions

In many situations, it may be necessary to run a different set of commands, depending on the context of execution. To achieve this flexibility, you use conditional commands. To define the situation under which the associated command steps execute, you use the **Condition** field in the Edit Command or New Command window.

Conditions are evaluated as boolean expressions. If the expression evaluates to TRUE, the command is executed. If it evaluates to FALSE, the command is skipped and the next command is evaluated. If no condition is specified, the command is always executed. The syntax of a condition is identical to the WHERE clause in an SQL statement. It provide enormous flexibility in evaluating scenarios. [Table 2-1 on page 25](#) lists some example conditions. The condition can include tokens. For more information, see [Chapter 4, Using Tokens, on page 47](#).

Table 2-1. Example conditions

Condition ^a	Evaluates to
BLANK	Command is executed in all situations.
'[P.P_VERSION_LABEL]' IS NOT NULL	Command is executed if the parameter with the token P_VERSION_LABEL in the package line is not null.
'[DEST_ENV.ENVIRONMENT_NAME]' = 'Archive'	Command is executed when the destination environment is named Archive.
'[AS.SERVER_TYPE_CODE]'= 'UNIX'	Command is executed if the application server is installed on a UNIX machine.

a. You must place single quotes around string literals or tokens that are used to evaluate strings.

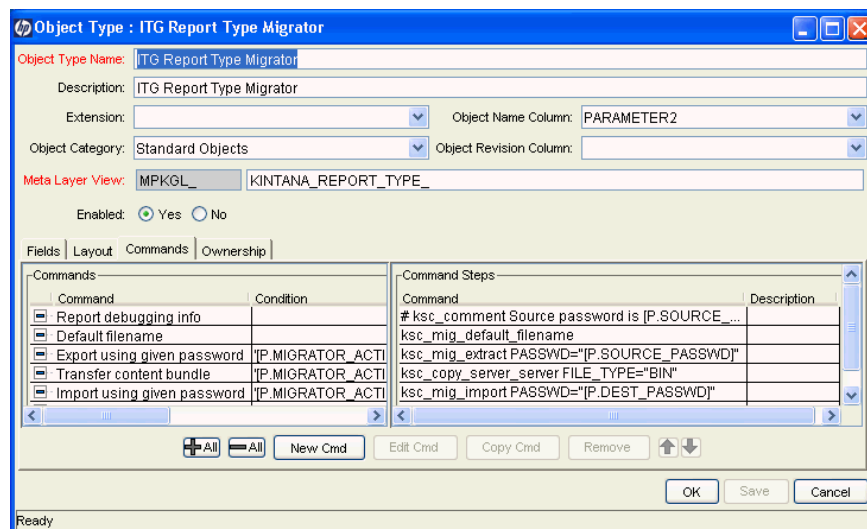
About the Commands Tab

Within PPM Center, commands are configured using the **Commands** tab for the following entities:

- Object types
- Request types
- Report types
- Validations
- Workflow step sources
- Special commands

You can access the tab by opening one of the listed entities, and then selecting the **Commands** tab. *Figure 2-1* shows the **Commands** tab in the Object Type window.

Figure 2-1. Commands tab



The **Commands** tab is divided into two sections. The **Commands** section defines the command-line directive or special command to be issued. The **Command Steps** section displays the steps specified to execute the commands. A command step can be an actual command-line directive that is sent to the PPM Server or target machine, or it can be one of the many special commands.

The execution engine executes the commands and command steps in the order in which they listed on the **Commands** tab. To change the order of the commands or the command steps:

- On the **Commands** tab, click the command or command step, and then use the up and down arrow buttons to change the placement of the selected item.

Configuring Commands

Each object type, request type, validation, workflow step source, or report type can have many commands, and each command can include many steps. You can think of a command as a particular function for an object. Copying a file can be one command, and checking that file into version control can be another. For these functions to operate, a series of events must take place. You define these events in the command steps. To define the events, you must configure commands using the Commands tab.

You configure commands using the Commands tab in the following PPM Center entity windows:

- Object Type
- Request Type
- Report Type
- Validation
- Workflow Step Source
- Special Command

Commands consist of command information and command steps. In the examples presented in this chapter, commands are accessed through the HP Deployment Management Object Type window. However, the controls are the same in the other entity windows that you can use to configure commands.

To configure commands associated with an object type:

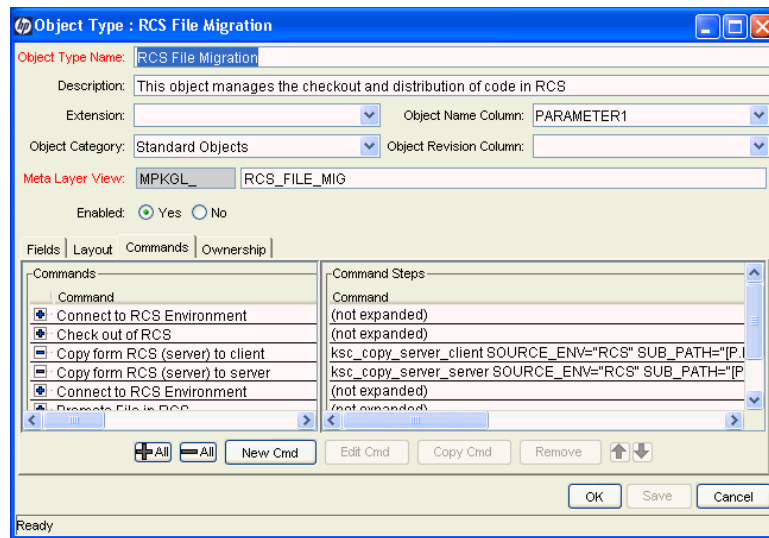
1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Deployment Mgmt > Object Types**.

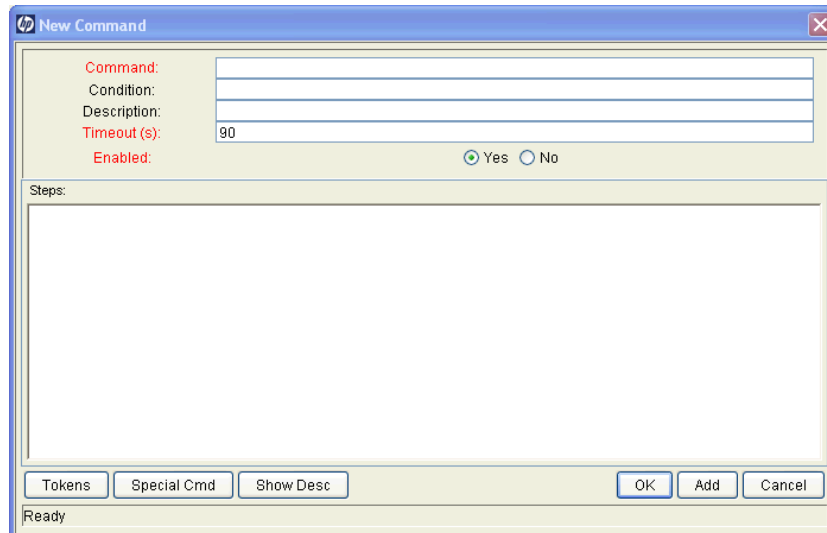
The Object Type Workbench window opens.

4. Open an existing object type.
5. In the Object Type window, click the **Commands** tab.



6. Click **New Cmd.**

The New Command window opens.



7. Complete the fields described in the following table.

Field Name	Description
Command	Command name.
Condition	Specific conditions under which the command steps are to be executed. This step is optional. For more information, see Command Conditions on page 25 .
Description	Command description. This step is optional. For more information, see Command Conditions on page 25 .
Timeout(s)	Length of time (in minutes) to run the command before stopping. The Timeout(s) setting is useful if a command hangs or takes too long to execute.
Steps	Command steps. (Enter at least one.)
Enable	Use the Yes and No option buttons to enable and disable the command, respectively.

- Click **Tokens** to open the Token Builder window and find a token to add to the command step. For information about tokens, see [Chapter 4, Using Tokens, on page 47](#).
- Click **Special Cmds** to open the Special Command Builder and find a special command to add to a command step. For information about special commands, see [Chapter 3, Using Special Commands, on page 33](#).

- To show or hide a **Descriptions** field in the **Steps** field, click **Show Desc** or **Hide Desc**.
8. Do one of the following:
- To add the command to the **Commands** tab and close the New Command window, click **OK**.
 - To add the command to the **Commands** tab and leave the New Command window open, click **Add**.

Examples of Command Uses

This section provides examples of commands.

To copy a file from one environment to another:

Command:

```
copy_client_client
```

Command Steps:

```
ksc_connect_dest_client
if [ ! d [P.P_SUB_PATH] ];
then mkdir -p [P.P_SUB_PATH]; fi
ksc_exit
ksc_copy_client_client SUB_PATH="[P.P_SUB_PATH]"
FILENAME="[P.P_FILENAME]" FILE_TYPE="[P.P_FILE_TYPE]"
```

To automatically update the staffing profile status to "In Planning:"

Command:

```
Update Staffing Profile Status
```

Command Steps:

```
ksc_set_staffing_profile_status USE_NAMES_FLAG="N"
STAFF_PROF_IDS="[REQ.P.KNTA_STAFFING_PROFILE]"
STATUS_NAME="In Planning"
```

To execute Oracle® SQL script against an Oracle Database using JDBC:

Command:

```
Execute SQL
```

Command Steps:

```
ksc_run_java com.kintana.core.server.execution.KSCSQLQuery
jdbc:oracle:thin:@[ENV="[ENV_NAME]".DB_NAME]:
[ENV="[ENV_NAME]".DB_PORT_NUMBER]:
[ENV="[ENV_NAME]".DB_ORACLE_SID]
[ENV="[ENV_NAME]".DB_USERNAME]
"[ENV="[ENV_NAME]".DB_PASSWORD]" "[QUERY_STRING]"
-token SQL_OUTPUT -delimiter "~" -file
[AS.PKG_TRANSFER_PATH][SYS.USER_ID].txt
[EXCEPTION_OPTION]
```

To log a program issue using a request type:

Command:

```
ksc_store
```

Command Steps:

```
ksc_store KNTA_ESCALATION_LEVEL="PROGRAM", "Program"
```

To run a report using UNIX:

Command:

```
Run report.
```

Command Steps:

```
ksc_local_exec [AS.ORACLE_HOME]/bin/[AS.SQLPLUS]
[AS.DB_USERNAME]/[AS.DB_PASSWORD]@[AS.DB_CONNECTION_STRING]
@./scripts/kntarpt_special_com
"[AS.REPORT_DIR]" "[RP.FILENAME]" "[P.P_FROM_COM]"
"[P.P_TO_COM]" "[P.P_SHOW_REF]"
```

To run a report using Windows®:

Command:

```
Run report.
```

Command Steps:

```
ksc_local_exec [AS.ORACLE_HOME]/bin/[AS.SQLPLUS]
[AS.DB_USERNAME]/[AS.DB_PASSWORD]@[AS.DB_CONNECTION_STRING]
@./scripts/kntarpt_special_com
'[AS.REPORT_DIR]' '[RP.FILENAME]' '[P.P_FROM_COM]'
'[P.P_TO_COM]' '[P.P_SHOW_REF]'
ksc_run_java
com.kintana.core.server.execution.CvtFileNameToLowerCaseCommand
"[AS.REPORT_DIR][RP.FILENAME].html"
```

3 Using Special Commands

In This Chapter:

- *About Special Commands*
 - *Special Command Parameters*
 - *Special Command Language*
 - *Special Command Conditions*
 - *Using the PPM Workbench to List Special Commands*
 - *About the Special Command Builder*
 - *Configuring Special Commands*
 - *Using Special Commands*
 - *Using the Special Command Builder*
 - *Nesting Special Commands*
 - *Using the Special Command Details Report to List Special Commands*
 - *Examples of Using Special Commands*
-

About Special Commands

Object types, request types, report types, workflows, and validations all use commands to access the execution layer. To simplify command execution, PPM Center provides a predefined set of special commands. Users can also create their own special commands.

Special commands are commands with variable parameters and are used in object types, request types, report types, workflows, and validation command steps. (Workflows use special commands in their workflow step sources.) These command steps perform various functions, such as copying files between environments and establishing connections to environments for remote command execution. PPM Center features two types of special commands:

- System special commands are shipped with the PPM Center. System special commands are read-only and have the naming convention `ksc_command_name`.
- User-defined special commands have the naming convention `sc_command_name`.

This chapter provides information about how to create, edit, and use special commands in PPM Center.

Special Command Parameters

Most special commands have parameters to override standard behavior. The **Parameters** tab displays these. Nearly all parameters are optional.

If a parameter is not passed to a special command and the default value for the parameter is a custom token, the entity using the command must contain a field with that token.

For example, the `ksc_copy_server_server` special command is used in an object type. The parameter `FILENAME` is not specified and defaults to `[P.P_FILENAME]` because it is not explicitly passed.

```
ksc_copy_server_server
```

This makes `ksc_copy_server_server` equivalent to:

```
ksc_copy_server_server FILENAME="[P.P_FILENAME]"
```

because `[P.P_FILENAME]` is the default token for the parameter `FILENAME`. The command execution engine evaluates the token `[P.P_FILENAME]` so it must be defined for the entity (the specific object type, report type or request type).

To override the default token, pass in another value for the parameter. A few examples are:

```
ksc_copy_server_server FILENAME="document.txt"  
ksc_copy_server_server FILENAME="[P.DOCUMENT_NAME]"
```

This method of passing parameters is explained in more detail in the section entitled *About the Special Command Builder on page 37*.



Custom tokens are defined for specific object types, request types, and report types, and are referenced using the `[P.TOKEN_NAME]` syntax.

Special Command Language

The command steps in a special command define the system-level executions that must be performed to realize the command function. Command steps can be UNIX commands, third-party application commands, or special commands. Special commands are reusable routines defined in PPM Center.

PPM Center also supplies several system special commands that you can use to perform common events such as connecting to environments or copying files.

Special Command Conditions

Depending on the context in which commands are executed, you may need to run a different set of commands. For example, one command may update a Web page, while another may set up an account on the Sales Automation application.

To achieve this flexibility, you use conditional commands. You can use the **Condition** field for an object command to specify the conditions under which the associated command steps are to be executed.

Conditions are evaluated as Boolean expressions. If the expression evaluates to TRUE, the command is executed. If it evaluates to FALSE, the command is skipped and the next command is evaluated to see if it should be run. If no condition is specified, the command is always executed.

The syntax of a condition is identical to the WHERE clause of a SQL statement, which allows flexibility when evaluating scenarios. *Table 3-1 on page 36* provides some example conditions.

Table 3-1. Example conditions

Condition	Evaluates to
BLANK	Command executes in all situations.
'[REQ.DEPARTMENT]' = 'SALES'	Command executes if the department for the request is named SALES.
'[REQ.PRIORITY]' = 'HIGH'	Command executes if the priority assigned to the request is HIGH.



In conditional commands, you must use single quotes to enclose strings.

A condition can include a token. For information on how to include tokens in conditions, see [Chapter 4, Using Tokens, on page 47](#) for more information.

Using the PPM Workbench to List Special Commands

To see a list of the special commands on your PPM Center instance:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. From the shortcut bar, select **Configuration > Special Commands**.

The PPM Workbench opens.

The Special Command Workbench window opens.

4. Click **List**.

The Special Command Workbench window lists descriptions of all the special commands and indicates the status (enabled or not) of each.

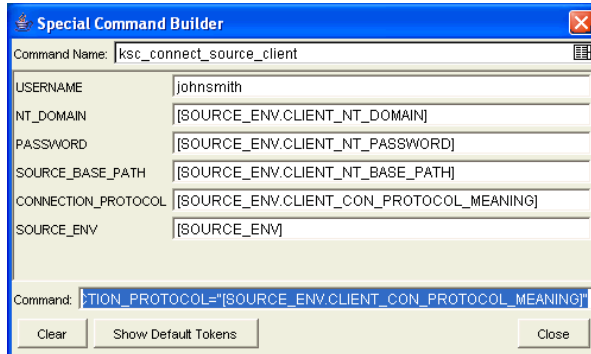


You can also use the Special Command Details report to view a list of special commands on your PPM Center instance. For information on how to access and run this report, see [Using the Special Command Details Report to List Special Commands on page 44](#).

About the Special Command Builder

The Special Command Builder (*Figure 3-1*) simplifies special command use by ensuring that you format command steps correctly. After you select a special command and specify its parameters, the Special Command Builder populates the **Command** field with a line of text that you can use as a command step.

Figure 3-1. Special Command Builder



For information about how to use the Special Command Builder, see *Using the Special Command Builder* on page 43.

Configuring Special Commands

To configure a new special command:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

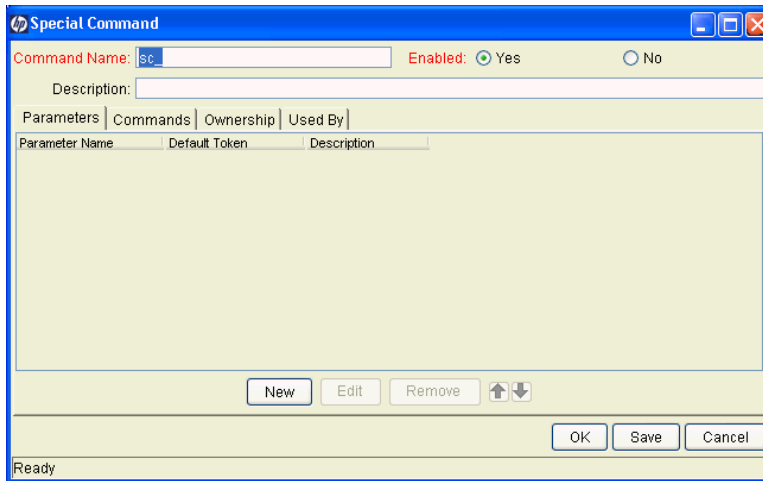
The PPM Workbench opens.

3. From the shortcut bar, select **Configuration > Special Commands**.

The Special Command Workbench opens.

4. Click **New**.

The Special Command window opens.

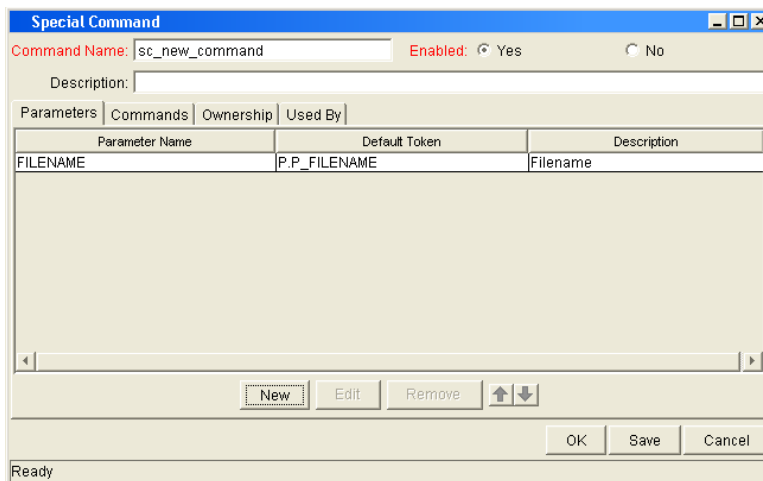


5. Complete the fields as specified in the following table.

Field Name	Description
Command Name	The name of the special command. This can only be updated when generating or editing a user-defined special command.
Enabled?	Determines whether or not the special command is enabled for use in workflows, object types, report types, request types, and validations.
Description	A description of the special command. This can only be updated when generating or editing a user-defined special command.

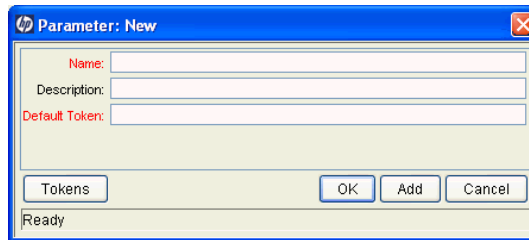
6. Configure a new parameter, as follows:

a. Click the **Parameters** tab.



- b. Click **New**.

The Parameter: New window opens.



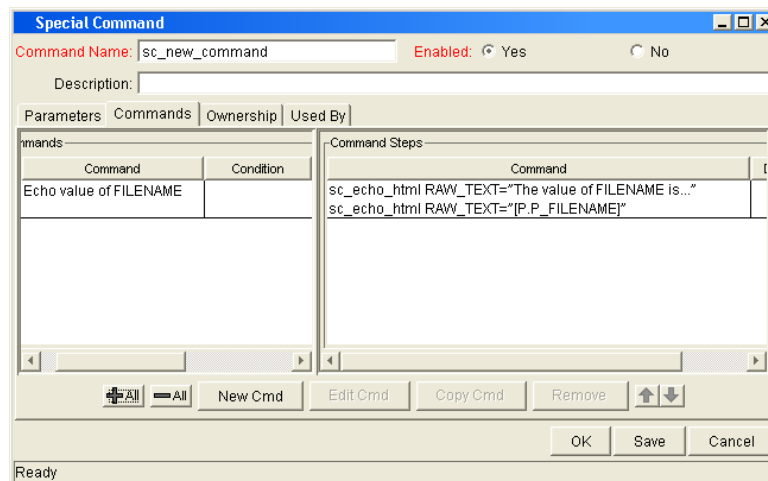
- c. Complete the fields as specified in the following table.

Field Name	Description
Name	The name of the new parameter.
Description	A description of the new parameter.
Token	The default token name. Type the token name, or click Token to open the Token Builder and select the name.

- d. In the **Name** field, type a name for the new parameter.

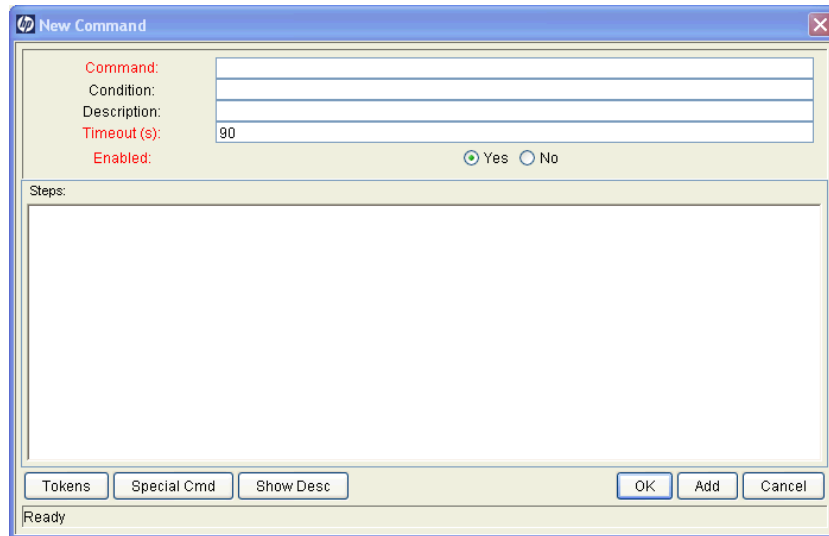
- e. To add the field to the **Parameters** tab, click **OK**.

- 7. Click the **Commands** tab.



- a. Click **New Cmd**.

The New Command window opens.

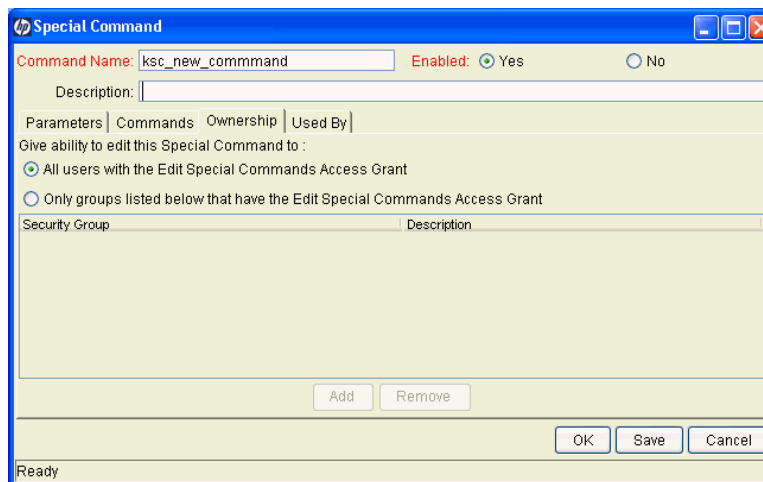


b. Complete the fields as specified in the following table.

Field Name	Description
Command	Command name.
Condition	Specific conditions under which the command steps are to be executed. This step is optional. For more information, see Special Command Conditions on page 35 .
Description	Command description. This step is optional. For more information, see Special Command Conditions on page 35 .
Timeout(s)	Amount of time (in minutes) to run the command. This setting is useful if a command hangs or takes too long to execute.
Steps	Enter at least one command step.
Enable	Yes and No option buttons enable or disable the command

- Click **Tokens** to open the Token Builder window. Use this window to find a token to add to the command step. For information about tokens, see [Chapter 4, Using Tokens, on page 47](#).
- Click **Special Cmds** to open the Special Command Builder. Use this tool to find a special command and add it to the command step. For information about special commands, see [Chapter 3, Using Special Commands, on page 33](#).

- Click **Show/Hide Desc** to show or hide a **Descriptions** field in the **Steps** field. If the **Descriptions** field is visible, you can add a description to the command step.
- c. To save the command, do one of the following:
- To add the command to the **Commands** tab and close the New Command window, click **OK**.
 - To add the command to the **Commands** tab and leave the window open, click **Add**.
8. Click the **Ownership** tab.



9. Under **Give ability to edit this Special Command to**, select **Only groups listed below that have the Edit Special Commands Access Grant**.
10. Click **Add**.

The Add Security Groups window opens.

11. Select the security groups.

12. Click **OK**.

The security groups are listed on the **Ownership** tab.

13. To add the security group to the special command:

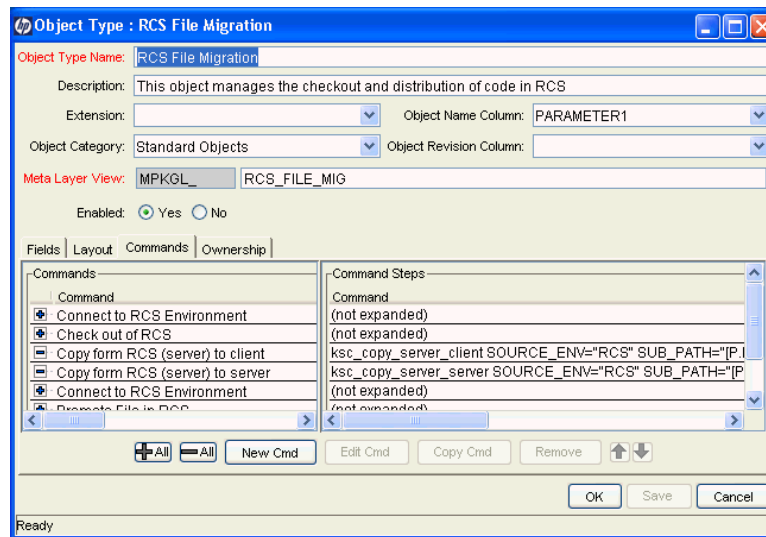
- To save the security group and close the Special Command window, click **OK**.
- To save the security group and leave the window open, click **Save**.

14. To see a list of entities that reference the selected special command, click the **Used By** tab.
15. To save the special command, click **OK**.

Using Special Commands

Special commands are added to command steps directly in the entity windows (for object types, request types, report types, validations and workflows). For example, *Figure 3-2* shows an example of an object type that was generated using a combination of special commands.

Figure 3-2. RCS File Migration object type



Using the Special Command Builder

You can add special commands to any set of command steps in the following entities:

- Object types
- Request types
- Report types
- Validations
- Workflow step sources
- Other special commands

You can access the Special Command Builder on the **Commands** tab for each of these entities.

To build a command step using the Special Command Builder:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Change Mgmt > Object Types**.

The Object Type Workbench opens.

4. Open an object type.
5. In the Object Type Window, click the **Commands** tab.
6. Click **New Cmd** or **Edit Cmd**.

The Command window opens.

7. Click **Special Cmd**.

The Special Command Builder window opens.

8. From the **Command Name** list, select the special command.

If you select a command name from the auto-complete, the Special Command Builder lists the command parameters.



You can use predefined (`ksc_command`) and user-defined (`sc_command`) special commands to build the command steps line.

9. Replace the associated default token value with parameter information.
 - a. To view the default tokens, click **Show Default Tokens**.
 - b. Copy the text in the Special Command Builder window **Command** field to the Command window **Steps** field.
10. Enter information in the remaining fields in the Command window.
11. For the **Enabled** option, click **Yes**.
12. To add the command step to the **Command** tab, click **OK**.

You can now use the new special command in an object type, request type, report type, validation, or workflow.



You can use special commands in an execution workflow step source. After you create the workflow step source (which contains the special commands), you can drag and drop it into a workflow.

Nesting Special Commands

You can use special commands within other special commands, but only within a command step. However, a special command cannot refer to itself.

Using the Special Command Details Report to List Special Commands

PPM Center comes with pre-configured special commands. To see a list of all special commands in your system, run the Special Commands Detail report. This report provides information on special commands, how to use them, the parameters of the special command, and where the special command is used.

To view the special commands on your instance:

1. Log on to PPM Center.
2. From the menu bar, select **Reports > Create a Report**.

The Reports window opens.

3. In the **Report Category** list, select **Administrative**.
4. From the displayed list of administrative reports, select **Special Command Details Report**.

The Special Command Details Report window opens.

5. To view all special commands, leave the **Special Command From** and **Special Command To** fields empty.

6. Under **Report Parameters**, select **Yes** for **Show References**.
7. Click **Submit**, and then wait to see the report displayed.



You can also use the Special Command Workbench to list the special commands on your PPM Center instance. For information on how to access the Special Command Workbench, see [Using the PPM Workbench to List Special Commands](#) on page 36.

Examples of Using Special Commands

This section provides examples of special commands.

To copy a file from one server to another server:

Special Command Name:

```
copy_server_server
```

Special Command Example:

```
ksc_connect_dest_server
if [ ! -d [P.P_SUB_PATH] ]; then mkdir -p [P.P_SUB_PATH]; fi
ksc_exit
ksc_copy_server_server SUB_PATH="[P.P_SUB_PATH]"
FILENAME="[P.P_FILENAME]" FILE_TYPE="[P.P_FILE_TYPE]"
```

To import using a given password:

Special Command Name:

```
ksc_mig_import
```

Special Command Example:

```
ksc_mig_import PASSWD="[P.DEST_PASSWD]"
```

To change the status of a project:

Special Command Name:

```
ksc_run_java
```

Special Command Example:

```
ksc_run_java com.kintana.core.server.execution.SetProjectStatus
-project [REQ.P.KNTA_PROJECT_PLAN] -status [P.P_STATUS]
-user [SYS.USER_ID]
```

To connect to a server and change permissions of a file:

Special Command Name:

```
ksc_connect_dest_server
```

Special Command Example:

```
ksc_connect_dest_server DEST_ENV="[DEST_ENV.ENVIRONMENT_NAME]"  
  
# 444 is read-only. if the locked flag  
# is no this is the permission set  
# the user requested  
chmod 0444 "[P.P_FILENAME]"  
  
ksc_exit
```

4 Using Tokens

In This Chapter:

- *About Tokens*
 - *Where to Use Tokens*
 - *Token Evaluation*
 - *About the Token Builder*
 - *Token Formats*
 - *Default Format*
 - *Explicit Entity Format*
 - *User Data Format*
 - *Parameter Format*
 - *Sub-Entity Format*
 - *Environment and Environment Application Tokens*
 - *Using the Token Builder*
-

About Tokens

PPM Center uses variables to facilitate the creation of general objects that can be used in a variety of contexts. These variables are called *tokens*.

PPM Center uses two types of tokens: standard tokens and custom tokens. Standard tokens come with the product. Custom tokens are generated to suit specific needs. You can reference the fields of the following entities as custom tokens:

- Object types
- Request types and request header types
- Report types
- User data
- Workflow parameters

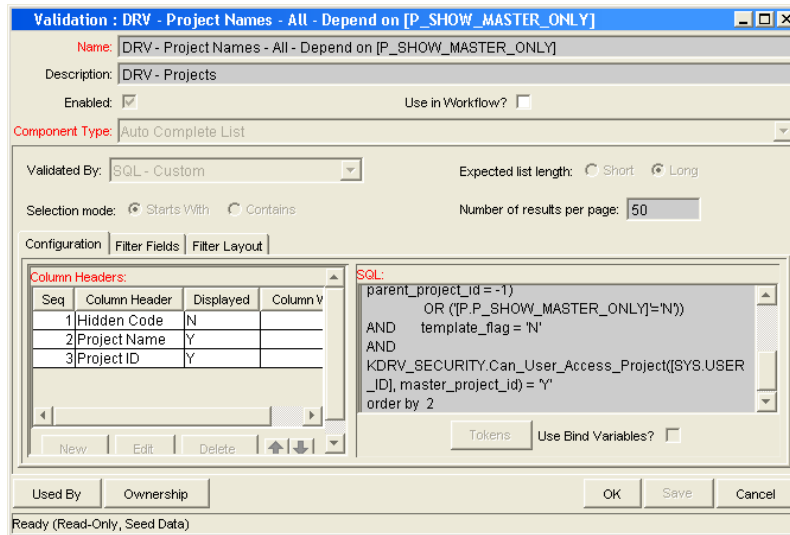
In addition, numerous standard tokens are available that provide other useful pieces of information related to the system. For example, PPM Center has a token that represents the users currently logged onto the system.

Where to Use Tokens

You can use tokens in the following entity windows:

- Object type commands
- Request type commands
- Validation commands and SQL statements
- Report type commands
- Executions and notifications for a workflow
- Workflow step commands
- Notifications in a report submissions
- Special command commands
- Notifications for tasks and time management
- Notes for request details

Figure 4-1. Example of a token used in a SQL statement



Token Evaluation

Tokens are evaluated at the point when PPM Center must know their context-specific values. At the time of evaluation, the token evaluation engine gathers information from the current context and tries to derive the value for the token. Values can only be derived for specific, known contexts (the current context is defined as the current package, package line, request, work plan, workflow step, or source and destination environments).

The token evaluation engine takes as many passes as necessary to evaluate all tokens, so one token can be nested within another token. During each pass, if the evaluation engine finds a valid token, it replaces that token with its derived value. Invalid tokens (for example, if the token name is misspelled or no context is available) are ignored.

For example, an object type command has the following Bourne-shell script segment as one of its command steps:

```
if [ ! -f [PKGL.P.P_SUB_PATH]/[PKGL.P.P_BASE_FILENAME].fmx ];
then exit 1; fi
```

When the command is executed, [PKGL.P.P_SUB_PATH] = Forms and [PKGL.P.P_BASE_FILENAME] = obj_maint. After token evaluation, this command step reduces to:

```
if [ ! -f Forms/obj_maint.fmx ]; then exit 1; fi
```

As another example, suppose a user data field is generated for all users called MANAGER. You could find the email address of the manager of the person who generated a request by using the following token:

```
[USR="[USR="[REQ.CREATED_BY_NAME]" .VUD.MANAGER]" .EMAIL_ADDRESS]
```

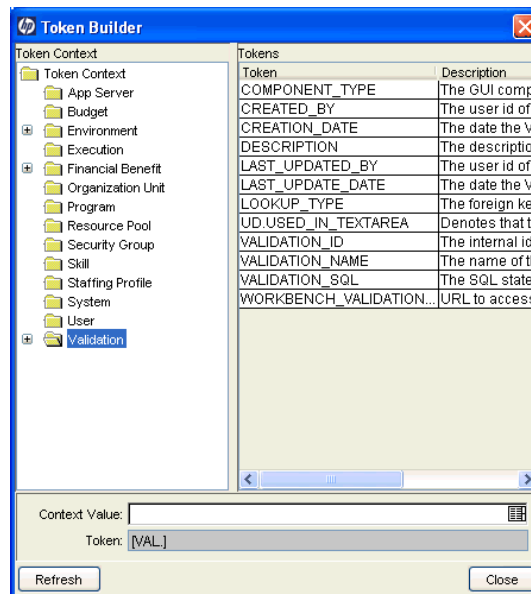
The token evaluation engine first evaluates the innermost token ([REQ.CREATED_BY_NAME]), and then the next token ([USR="<name>" .VUD.MANAGER]). Finally, token evaluation engine evaluates the outermost token, which provides the email address.

Tokens are evaluated at different points based on the token type. Tokens used in object type parameters and commands are evaluated during command execution. Tokens in a validation SQL statement are evaluated just before that statement is executed (such as generating a new package line). Tokens in an email notification are evaluated when a notification is generated.

About the Token Builder

From each of the entity windows listed in *Where to Use Tokens on page 48*, you can open the Token Builder window (*Figure 4-2*) to create a token. The tokens available in the token builder are limited to those that you can build for that entity. For example, if you open the token builder from the Request Type Workbench, package tokens are excluded.

Figure 4-2. Token Builder window



The folders displayed in the left pane of the Token Builder window contain groups of tokens that correspond to entities defined in PPM Center. For

instance, the Packages folder contains tokens that reference various package attributes. If you select the Packages folder, the available package tokens are listed in the right pane.

Some entities (folders) have sub-entities (sub-folders) that can be referenced by tokens. To view a list of sub-entities for an entity, click the plus character (+) next to the entity. Each sub-entity also has tokens, and you can reference any sub-entity tokens, as well as the parent entity tokens. For example, the package line entity is a sub-entity of the package entity.

As you select entity folders and corresponding tokens in the list, a character string is constructed in the **Token** field at the bottom of the Token Builder window. This is the formatted string used to reference the token. You can either copy and paste the character string, or type it where it is required.

Token Formats

Tokens can use one of several different formats, depending on how they are going to be evaluated. Tokens can be expressed in the following formats:

- *Default Format*
- *Explicit Entity Format*
- *User Data Format*
- *Parameter Format*
- *Sub-Entity Format*
- *Environment and Environment Application Tokens*

Table 4-1 on page 52 lists the entities and the formats that each entity supports.

Table 4-1. Entities (page 1 of 3)

Prefix (Entity)	Entity and Description	User Data Format	Parameter Format
AS	Application server	N	N
BGT	Budget	Y	N
CON	Contact	Y	N
DEST_ENV	Destination environment. If an app code is specified, it is used. Otherwise, use only values from ENV.	Y	N
DEST_ENV.APP	Destination environment (for the environment application). Only use app code values, even if they are null.	Y	N
DEST_ENV.ENV	Destination environment. Ignores app codes and only uses the ENV values.	Y	N
DIST	Distribution	Y	N
ENV	Environment	Y	N
ENV.APP	Environment (for the environment application). Only use app code values, even if they are null.	Y	N
ENV.ENV	Environment. Ignores app codes and only uses the ENV values.	Y	N
EXEC	Execution	N	N
FBEN	Financial benefit	Y	N
NOTIF	Notification	N	N
ORG	Organization Unit	Y	N
PKG	Package	Y	N
PKG.PKGL	Package (package line)	Y	N
PKG.PEND	Package (pending package)	Y	N
PKGL	Package line	Y	Y
PRG	Program	Y	N

Table 4-1. Entities (page 2 of 3)

Prefix (Entity)	Entity and Description	User Data Format	Parameter Format
PRJ	Work plan	Y	N
PRJD	Work plan details	N	Y
REL	Release	N	N
REL.DIST	Release (distribution)	Y	N
REQ	Request	Y	Y
REQ.FIELDSD	Request field groups	N	Y
REQ.PEND	Request (pending)	N	N
REQD	Request details	N	Y
REQD.P	Request details	N	Y
RP	Report submission	N	Y
RSCP	Resource pool	Y	N
SG	Security group	Y	N
SKL	Skill	Y	N
STFP	Staffing profile	Y	N
SOURCE_ ENV	Source environment	Y	N
SOURCE_ ENV.APP	Source environment (for environment application). Only use app code values, even if they are null.	Y	N
SOURCE_ ENV.ENV	Source environment. Ignores app codes and only uses the ENV values.	Y	N
SYS	System	N	N
TMG	Time Management	N	N
TSK	Task	Y	N
TSK.PEND	Task (pending)	N	N
USR (User)	User	Y	N
VAL	Validation	N	N

Table 4-1. Entities (page 3 of 3)

Prefix (Entity)	Entity and Description	User Data Format	Parameter Format
WF	Workflow	Y	N
WF.WFS	Workflow (step). Use this format to specify a specific workflow.	N	Y
WFS	Workflow step	Y	N

Default Format

Tokens are expressed as a prefix (a short name for the entity) followed by a token name. The prefix and token name are separated by a period and enclosed in square brackets with no spaces:

[PREFIX.TOKEN_NAME]

For example:

The token for the package number is expressed as:

[PKG.NUMBER]

The token for a request's workflow name is expressed as:

[REQ.WORKFLOW_NAME]

Certain tokens also support a sub-format. This sub-format is required for certain entities in order to evaluate to the correct context. For example, `WF` tokens resolve to information related to the workflow, whereas `WF.WFS` tokens resolve to workflow step information. Token sub-formats are included in the prefix, appended to the parent prefix, and separated by a period:

[PREFIX.SUB-PREFIX.TOKEN_NAME]

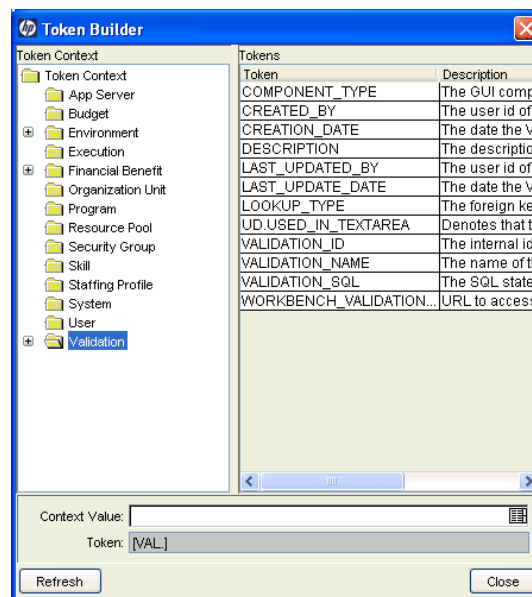
Tokens are evaluated according to the current context of PPM Center, which is derived based on information known at the time of evaluation. For more information, see [Token Evaluation on page 49](#).

Explicit Entity Format

You can provide a specific context value for an entity so that the default context can be overridden. Some tokens can never be evaluated in the default context. In these cases, you must use the following explicit entity format to set the context:

```
[PREFIX="<entity name>".<TOKEN_NAME>]
```

The token builder generates tokens in the explicit entity format by providing a list of possible values. When such a list is available, the **Context Value** field at the bottom of the Token Builder window is enabled. You can either type in the field to reduce the list, or click the auto-complete icon to open the Validate window. The value you select is placed in the token in the **Token** field to generate an explicit entity token.



For example, suppose you want to reference the email address for jsmith. The token to specify this reference is:

```
[USR="jsmith".EMAIL_ADDRESS]
```

To construct the token `[USR="jsmith".EMAIL_ADDRESS]` in the Token Builder window:

1. Open the Token Builder window.

See *About the Token Builder* on page 50.

2. In the Token Builder window, select the **User** folder.

Available tokens are listed in the Tokens column, and the **Context Value** field is enabled.

The **Token** field displays the string `[USR.]`.

3. In the **Context Value** field, select **jsmith**.

The **Token** field displays the string `[USR="jsmith"]`.

4. In the Tokens column, click **EMAIL_ADDRESS**.

The **Token** field displays the string `[USR="jsmith".EMAIL_ADDRESS]`.

This is the complete token. Since the token is now complete, the **Token** field becomes enabled.

5. Select and copy the text in the **Token** field.
6. Paste the text into another field.



For a list of all explicit entity format tokens, see [Appendix A, Tokens](#), on page 129.

Nesting Explicit Entity Tokens within Other Tokens

The explicit entity format can be used to nest tokens within other tokens to generate a value. For example, to print the description of the workflow that is associated with package #10203, the token would be:

```
[WF="[PKG="10203".WORKFLOW_NAME]".DESCRIPTION]
```

This token would have to be built in two steps. First, build the Description token for the workflow. Copy and paste that token into another field, then build the Workflow Name token for the package. Copy and paste that token within the Description token that was previously pasted.

Internally, this token is evaluated in two stages. The inner token is evaluated and the token has the following internal representation:

```
[WF="Workflow_Name".DESCRIPTION]
```

The remaining token is evaluated and the final result is printed:

```
description of my workflow
```

User Data Format

User data fields use tokens differently, as shown below:

```
[PREFIX.UD.USER_DATA_TOKEN]
```

The `PREFIX` is the name of the entity that has user data. The modifier `UD` indicates that user data for that entity is being referenced. `USER_DATA_TOKEN` is the name of the token for the specific user data field. For example, suppose that a field for package user data is generated, and its token is `GAP_NUMBER`. In the default format, the token would be:

```
[PKG.UD.GAP_NUMBER]
```

In this context, `PKG` indicates that the package entity is referenced, `UD` indicates that user data is referenced, and `GAP_NUMBER` is the token name.

When user data fields are generated, a validation that has both a hidden and visible value can be used. For example, if the validation `KNTA - Usernames - All` is used, the hidden value is the user ID and the displayed value is the username. The previous syntax references the hidden value only. To reference the visible value for a user data field, the syntax shown below must be used:

```
[PREFIX.VUD.USER_DATA_TOKEN]
```

If the modifier `VUD` is used instead of `UD`, the visible user data value is referenced.



Drop-down lists and auto-completes may have different hidden and displayed values. For all other validations, hidden and displayed values are identical.

If context can be determined, user data tokens are displayed with the system-defined tokens in the Token Builder window.

Parameter Format

Object type custom fields, request type custom fields, request header type fields, work plan fields, and workflow parameters use the parameter format for tokens as shown below:

```
[PREFIX.P.PARAMETER_TOKEN]
```

In this specific case, `PREFIX` is the name of the entity that uses a custom field. The modifier `P` indicates that parameters for that entity are referenced. `PARAMETER_TOKEN` is the name of the token for the specific parameter field.

➤ Package lines reference object type fields. Requests reference request type and request header type fields. Workflows reference workflow parameters.

For example, suppose a field for an object type named Gap Number (Token = `GAP_NUMBER`) is been generated for use on package lines. In the default format, the token would be:

```
[PKGL.P.GAP_NUMBER]
```

In this context, `PKGL` is the prefix, because the package lines entity is referenced, `P` indicates that parameters are referenced, and `GAP_NUMBER` is the token name.

Custom fields store both a hidden and visible value. For example, if the field uses the validation `KNTA - Usernames - All`, the hidden value is the user ID and the displayed value is the username. The previous syntax references the hidden value only. To reference the visible value for a parameter, use the syntax as shown:

```
[PREFIX.VP.PARAMETER_TOKEN]
```

If the modifier `VP` is used instead of `P`, the visible parameter value is referenced.

➤ Drop-down lists and auto-completes may have different hidden and displayed values. For all other validations, the hidden and displayed values are identical.

Request Field Tokens

Tokens can access information on custom fields included on a request. These fields can be defined in a:

- Custom request type field
- Request header field (standard)
- Request header field (custom fields)
- Request header field (field groups)
- Table component field

Request Field Token Prefixes

All fields defined in the request header type (field group fields, custom header fields, and standard header fields) use the `REQ` prefix. The following examples could use `P` or `VP`.

```
REQ.<standard header Token>
REQ.DEPARTMENT_CODE
REQ.P.<custom header field Token>
REQ.P.BUSINESS_UNIT
REQ.P.<field group Token starting with KNTA_>
REQ.P.KNTA_SKILL
```

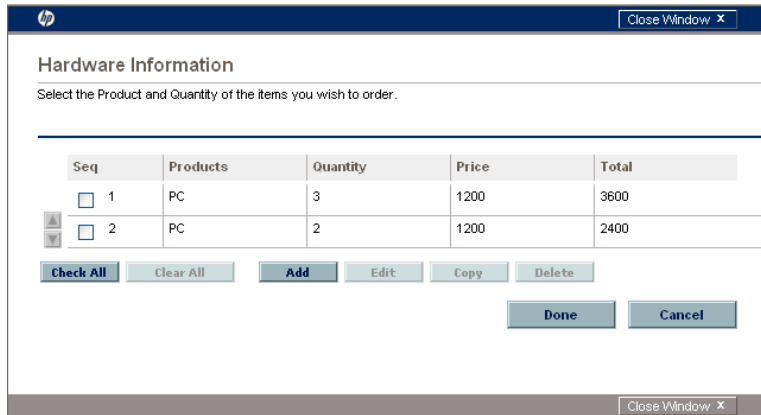
Fields defined in the request type use the `REQD` prefix. You can also access standard header fields using the `REQD` prefix. For example:

```
REQD.P.<custom detail field>
REQD.<standard header Token>
```

Tokens in Request Table Components

To refer to items in a table component, tokens must follow specific formats. The formats used depends on the table item referenced. [Figure 4-3 on page 60](#) shows the basic elements of a sample table. These elements are used as examples for referencing data within the table using tokens.

Figure 4-3. Table component formats



The format `[REQD.T.<TABLE_TOKEN>]` is used to represent the table. The format `[REQD.T.<TABLE_TOKEN>.<SPECIFIC_TOKENS>]` is used to represent specific tokens. The following sections provide examples of the formats used for tokens that reference items related to the table component:

- *To access the table row count from a Request context:*
- *To access the Salary Column Total value from a Request context:*
- *To access the Name of the first employee in the table from a Request:*
- *To access the Code of the first employee in the table from a Request:*
- *To access the Department Cell value of the current row (Table Row Context):*
- *To obtain a delimited list of a column's contents (Request Context)*

In these examples, a table component named Employee has the following four columns:

- Employee Name
- Years of Service
- Department
- Employee Salary

These columns are defined as follows:

Table Component "Employee" table with [EMPLOYEE] as the Token.
Column 1 - Employee Name; Token = [NAME]
Column 2 - Years of Service; Token = [YEARS_OF_SERVICE]
Column 3 - Department; Token = [DEPARTMENT]
Column 4 - Employee Salary; Token = [SALARY]

To access the table row count from a Request context:

[REQD.P.EMPLOYEE] - returns the raw row count without any descriptive information.

[REQD.VP.EMPLOYEE] - returns the row count with descriptive information. Example "13 Entry(s)".

WHERE: EMPLOYEE is the Token given to a table component type.

To access the Salary Column Total value from a Request context:

[REQD.T.EMPLOYEE.TC.VP.SALARY.TOTAL]

WHERE: EMPLOYEE is the Token given to a table component type and SALARY is the Token name given the table's first column.

To access the Name of the first employee in the table from a Request:

[REQD.T.EMPLOYEE.TE="1".VP.NAME]

To access the Code of the first employee in the table from a Request:

[REQD.T.EMPLOYEE.TE="1".P.NAME]

To access the Department Cell value of the current row (Table Row Context):

[TE.VP.DEPARTMENT]

You can use this table component token in a Table Column Header validation SQL or in a table component rule SQL.

To obtain a delimited list of a column's contents (Request Context)

[REQD.T.EMPLOYEE.TC.VP.NAME]

where EMPLOYEE is the token given to a table component type and SALARY is the token name given the first column of the table. This is very useful if a column lists user names. This list can be used to send the users notification.

Sub-Entity Format

Some entities have sub-entities that can be referenced. To see a list of sub-entities for an entity, in the Token Builder window, click the plus character (+) next to the entity. To reference a token from a sub-entity, in the context of a parent entity, use the following syntax:

```
[PREFIX.SUB_ENTITY_PREFIX.TOKEN]
```

In this case, the `PREFIX` is the name of the entity, the `SUB_ENTITY_PREFIX` is the prefix for a sub-entity, and `TOKEN` is a token of the sub-entity. Typically, it is not necessary to use this syntax. However, you can reference specific sub-entities using the explicit entity syntax. For example, to reference the step name of the workflow step in the current context, both of the following tokens have the same meaning:

```
[WFS.STEP_NAME]  
[WF.WFS.STEP_NAME]
```

However, to reference the step name of the first workflow step for the current workflow, use the following token:

```
[WF.WFS="1".STEP_NAME]
```

By not using the explicit entity format for the workflow entity, the token indicates that the workflow in the current context should be used. But by using the explicit entity format for the workflow step entity, the current context is overridden and a specific workflow step is referenced. In contrast, to reference the step name of the first workflow step in a workflow whose name is 'my workflow,' use the following token:

```
[WF="<workflow_name>".WFS="1".STEP_NAME]
```

With this token, the current context for both the workflow and the workflow step are overridden.

Environment and Environment Application Tokens

Tokens for the environments and environment application entities can have many different forms depending on the information to be referenced. During object type command execution, there is generally a source and a destination environment. The token prefixes `SOURCE_ENV` and `DEST_ENV` are used to reference the current source and destination, respectively, as shown in the following example:

```
[SOURCE_ENV.DB_USERNAME]
[DEST_ENV.SERVER_BASE_PATH]
```

In addition, you can use a general `ENV` prefix in the explicit entity format to reference specific environments, as shown in the following example:

```
[ENV="Prod".CLIENT_USERNAME]
```

During normal environment token evaluation, the evaluation engine first evaluates the app code on the package line (if one is specified). If the corresponding app code token has a value, then the value is used. Otherwise, if no app code was specified or the app code token has no value, the corresponding base environment information is used.

To override the normal environment token evaluation and only evaluate the environment information (without first checking for the app code), construct the `SOURCE_ENV` and `DEST_ENV` tokens as shown in the following examples:

```
[SOURCE_ENV.ENV.DB_USERNAME]
[DEST_ENV.ENV.SERVER_BASE_PATH]
[ENV="Prod".ENV.CLIENT_USERNAME]
```

The evaluation engine can be instructed to look only at the app code information (without checking the base environment information if the app code token has no value). Construct the `SOURCE_ENV` and `DEST_ENV` tokens as shown in the following example:

```
[SOURCE_ENV.APP.DB_USERNAME]
[DEST_ENV.APP.SERVER_BASE_PATH]
[ENV="Prod".APP.CLIENT_USERNAME]
```

You can only use the prefix `APP` in the sub-entity format. For example, the following token is invalid because a context environment that includes the app code has not been specified.

```
[APP.SERVER_BASE_PATH]
```

In addition, you can use the explicit entity format with the app code entity to reference a specific app code, as shown in the following examples:

```
[SOURCE_ENV.APP="AR".DB_USERNAME]
[DEST_ENV.APP="OE".SERVER_BASE_PATH]
[ENV="Prod".APP="HR".CLIENT_USERNAME]
```

For example, suppose objects are migrated on a package line at a given workflow step, and the line uses the app code `HR`. The workflow step has `QA` as the source environment, and `Prod` as the destination environment. *Table 4-2* shows other attributes of the environments and applications.


Table 4-2. Sample environment and application attributes

Environment	App Code	Server Base Paths
QA		/qa
QA	OE	/qa/oe
QA	HR	/qa/hr
Prod		/prod
Prod	OE	/prod/oe
Prod	HR	no value

Table 4-3 lists some sample tokens and the evaluation of each within the sample environment.

Table 4-3. Sample environment tokens

Token	Evaluation
[SOURCE_ENV.SERVER_BASE_PATH]	/qa/hr
[DEST_ENV.SERVER_BASE_PATH]	/prod
[SOURCE_ENV.ENV.SERVER_BASE_PATH]	/qa
[DEST_ENV.ENV.SERVER_BASE_PATH]	/prod
[SOURCE_ENV.APP.SERVER_BASE_PATH]	/qa/hr
[DEST_ENV.APP.SERVER_BASE_PATH]	no value
[ENV="QA".APP="OE".SERVER_BASE_PATH]	/qa/oe

 If PPM Center Extensions are installed, there are more environment tokens with the prefix 'AC.' For information about these tokens, see the documentation for the PPM Center Extension(s).

Using the Token Builder

Some tokens can never be evaluated in the default format. In these cases, you must use the explicit entity format to set the context, such as:

```
[PREFIX="<entity name>".<TOKEN_NAME>]
```

Token Builder generates tokens in the explicit entity format by providing a list of possible entity name values. When such a list is available, the **Context Value** field at the bottom of the Token Builder is enabled. You can either type in the field to reduce the list, or click the auto-complete icon to open the Validate window (see [About the Token Builder on page 50](#)). The selected value is inserted into the token in the **Token** field to generate an explicit entity token.

For example, you need to reference the email address for jsmith. The token to specify this reference is:

```
[USR="jsmith".EMAIL_ADDRESS]
```

To configure the token [USR="jsmith".EMAIL_ADDRESS] in the Token Builder window:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Demand Mgmt > Request Types**.

The Request Types Workbench opens.

4. Open a new or existing request type.

The Request Type window opens.

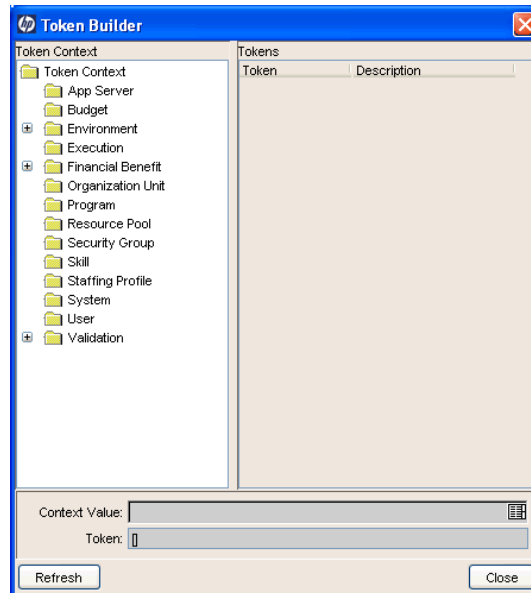
5. Click the **Commands** tab.

6. Click **New Cmd**.

The Commands window opens.

7. Click **Tokens**.

The Token Builder window opens.



8. Select the **User** folder.

Available tokens are listed in the right pane, and the **Context Value** field is available at the bottom of the window

The **Token** field displays the string: [USR].

9. Click the auto-complete icon in the **Context Value** field.

A Validate window opens.

10. In the list of users, scroll down to and select **jsmith**.

11. Click **OK**.

The **Token** field displays the string: [USR="jsmith"].

12. In the **Tokens** column, select **EMAIL_ADDRESS**.

The **Token** field displays the string: [USR="jsmith".EMAIL_ADDRESS].

Because this is the complete token, the **Token** field is enabled.

13. Copy the text in the **Token** field, and then paste it into another field.

5 Using Validations

In This Chapter:

- *About Validations*
 - *Validation Component Types*
 - *Accessing Validations Through Packages and Requests*
 - *Validations and Special Characters*
 - *Viewing System Validations*
 - *Configuring Validations*
 - *Configuring Static List Validations*
 - *Configuring Dynamic List Validations*
 - *Configuring SQL Validations*
 - *Configuring Short List Auto-Complete Field Validations*
 - *Configuring Long List Auto-Complete Field Validations*
 - *Configuring Automatic Value Matching and Interactive Select Pages*
 - *Adding Search Fields to Long List Auto-Complete Validations*
 - *Configuring the Filter Field Layout*
 - *Configuring an Auto-Complete List of Users (Special Case)*
 - *Configuring User-Defined Multi-Select Auto-Complete Fields*
 - *Configuring Text Field Validations*
 - *Text Data Masks for Validations*
 - *Configuring the Numeric Data Mask*
 - *Configuring the Currency Data Mask*
 - *Configuring the Percentage Data Mask*
 - *Configuring the Telephone Data Mask*
 - *Configuring a Custom Data Mask*
 - *Configuring Directory Chooser Validations*
 - *Configuring File Chooser Validations*
 - *Configuring Date Field Validations*
 - *Configuring 1800 Character Text Areas*
 - *Configuring the Table Component*
 - *Configuring Table Components*
 - *Configuring Table Rules*
-

About Validations

Validations determine the acceptable input values for user-defined fields (such as object type or request type fields). Validations also determine the possible results that a workflow step can return. Validations are used for the following two functions:

- **Field component type.** Users can create fields for several entities, including object types, request types, request header types, and user data. Validations determine the field component type (for example, text field or drop-down list) and define possible field values.
- **Workflow step results.** Validations determine the possible results of exiting a workflow step. For example, the validation WF - Standard Execution Results contains the possible execution step results of **Succeeded** or **Failed**.

Every PPM Center installation includes predefined system validations, which you can use as you configure your system. If no system validation meets your specific requirements, you can use the Validation Workbench to create your own validation. (For details, see [Configuring Validations on page 74](#).)

Validation Component Types

You can only use certain component types in a workflow step source validation. *Table 5-1* summarizes the field component types available.

Table 5-1. Component types (page 1 of 2)


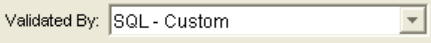
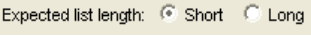
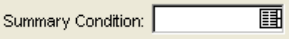

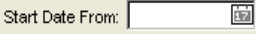



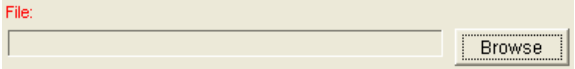



Component Type	Use In Workflow?	Description and Example
Text field	Yes	Text entry fields displayed on a single line. You can configure text fields to display the data in a specific format. For example, you can configure a text field to accept and format a hyphenated nine-digit social security number or a ten-digit phone number. 
Drop-down list	Yes	Field that displays a list of values. 
Option buttons (Yes/No)	No	Field that accepts yes/no input. 
Auto-complete list	Yes	Field that lets you open a dialog box that lists choices. 
Text area	No	Text entry field that can include multiple lines. 
Date field	No	Field that lets you specify date and time in one long, medium, short, or no format. 
Web address (URL)	No	Text entry field for entering a URL. Clicking U opens a browser window to the specified Web address. 

Table 5-1. Component types (page 2 of 2)

Component Type	Use In Workflow?	Description and Example
File chooser	No	<p>Used only in object types. Requires that two fields be defined with the tokens P_FILE_LOCATION and P_SUB_PATH. For configuration details, see Accessing Validations Through Packages and Requests on page 71.</p> 
Directory chooser	No	<p>Used only in object types. Requires a parameter field defined with the token P_FILE_LOCATION.</p> 
Attachment	No	<p>Field used to locate and attach files.</p> 
Password field	No	<p>Field used to capture passwords.</p> 
Table component	No	<p>Used to enter multiple records into a single component. The table component can be configured to include multiple columns of varied data types. This component supports rules for populating elements within the table and provides functionality for capturing column totals. For details, see Configuring the Table Component on page 118.</p> <p>You can only add fields of this component type to request types, request header types and request user data.</p> <p style="text-align: center;">Hardware Information 2 Entries </p>
Budget, staffing profile, financial benefit	No	<p>Field that you can add to the request type to enable access to view, edit or create budgets, staffing profiles, or financial benefits associated with a request, project, or work plan.</p> <p>You can only add fields of this component type to a request type.</p> <p style="text-align: center;">Budget: Team T Allocations </p>

Accessing Validations Through Packages and Requests

You can access the package and request group validations directly from the Package window. You do not have to use the Validation Workbench to specify that a package belongs to a new or unique package group that is not named in the auto-complete validation list.

To access the package and request groups validation window from the Package Workbench:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Package > New Package Group**.

The Validation window opens and lists the existing HP Deployment Management package groups.

Seq	Code	Meaning	Enabled	Description	Default
1	CUSTOMIZATION	Customization	Y	Customization	N
2	SETUP	Setup	Y	Setup	N
3	UPGRADE	Upgrade	Y	Upgrade	N



Although all users can view this window, only users with the required security privileges can change the package and request groups validation list.

To access the CRT - Request Type Category validation directly from the Request Types Workbench:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.
The PPM Workbench opens.
3. From the shortcut bar, select **Request Type > Request Type Category Setup**.
The Validation window opens and lists the existing request type categories.

Seq	Code	Meaning	Description	Enabled	Default
1	MISCELLANEOUS	Miscellaneous		Y	N



Although all users can view this window, only users with the required security privileges can change the CRT - Request Type Category validation list.

Validations and Special Characters

You cannot enter the question mark character (?) in the validation **Name** field. The PPM Workbench prevents users from typing this character in the field.

Viewing System Validations

PPM Center comes with several pre-configured validations. Note that some of these validations may have been altered to better match the specific business needs of your organization. To see a list of all validations in your system, run the Validations report. This report provides information on validation values and commands.

To view the existing validations on your instance:

1. Log on to PPM Center.
2. From the menu bar, select **Reports > Create a Report**.

The Reports window opens.

3. In the **Report Category** list, select **Administrative**.

The Reports window lists the Administrative reports.

4. Select **Validations Report**.

The Validations Report window opens.

5. Provide the following information:

- To view all of the special commands, leave the fields **Validations From** and **Validations To** empty.
- Under **Report Parameters**:
 - For **Show Validation Values**, select **Yes**.
 - For **Show Validation Commands**, select **Yes**.
 - For **Expand Special Commands**, select **Yes**.

6. Click **Submit**, and then wait to see the report displayed.

Configuring Validations

You can create, edit, and delete validations using the Validations Workbench. Be sure to exercise caution if you edit existing validations that are in use by fields or workflow step sources. Both field and workflow step validations can be tied to workflow logic. Changing the validation values can invalidate a process. To create, edit, or delete a validation requires the correct access grants. For more information about access grants, see the *Security Model Guide and Reference*.

Guide and Reference.

You cannot delete a validation if it is:

- A system validation (delivered with the product as seed data).
- Currently used by a workflow step source. You can only disable validations referenced by workflow step sources. Although a disabled validation continues to function in existing workflow steps, you cannot use it to define a new step source.
- Currently used by a field in a product entity (object type, request type, user data, report type, or project template field). You can only disable validations referenced by entity fields. Although a disabled validation continues to function in existing fields, you cannot use it to define a new field.



Although you may not be able to delete a custom validation, you can disable it. This allows any active workflows or product entities to use the validation, but keeps it from being used in new workflows or entity definitions.

To configure a new validation:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

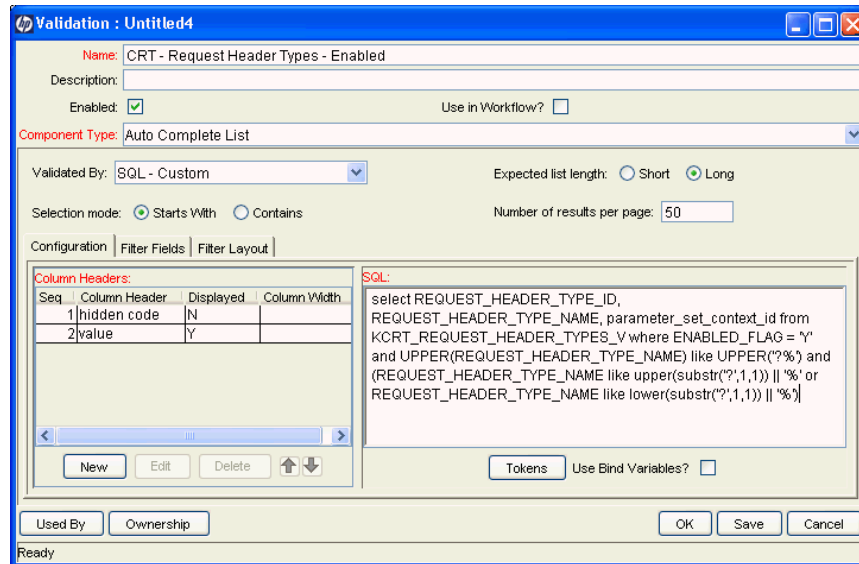
The PPM Workbench opens.

3. From the shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

4. Click **New Validation**.

The Validation window opens.



5. Provide the information described in the following table.

Field Name	Description
Name	Name of the new validation.
Description	Brief description of the validation.
Enabled	Select this checkbox to enable the validation.
Use in Workflow	Select this checkbox to use the validation in a workflow step source. You can only use text field, list, and auto-complete component types within workflow step sources.
Component Type	<p>Select a validation type. Selecting a listed value dynamically updates the Validation window to display fields used to configure the selected validation type.</p> <p>The component types are:</p> <ul style="list-style-type: none"> ■ Text Field ■ Drop Down List ■ Radio Buttons ■ Auto Complete List ■ Text Area ■ Date Field ■ Web Address ■ File Chooser

6. Enter any additional information required for the selected component type.

Additional information depends on the component type selected. Selecting a components type dynamically changes the remaining fields. The remainder of this chapter details how to configure the difference component types.

7. Specify which users through security groups can edit, copy, and delete this validation.

- a. Click **Ownership**.

The Ownership window opens.

- b. Select **Only groups listing below that have the Edit Validations Access Grant**.

- c. Click **Add**.

The Add Security Group window opens.

- d. Add security groups to the validation.

- e. Click **Apply** to add a security group. Click **OK** to add a security group and close the Add Security Group window.

8. Click **OK**.

Configuring Static List Validations

A static list validation can be a drop-down list or an auto-complete component. You can create static list validations that provide a static list of options to the user. For example, XYZ Corporation creates a validation called Engineering Teams for its engineering teams. The validation consists of the values New Product Introduction, Product One, and Product Two.

To create a static list validation for the engineering teams:

1. Log on to PPM Center.
2. From the menu bar, select **Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

4. Open a validation.

The Validations window opens.

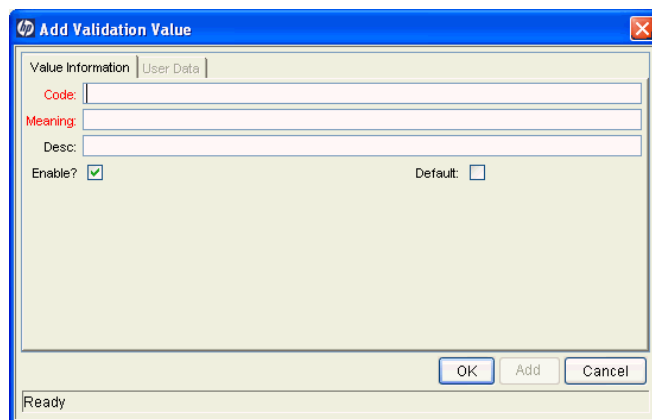
5. In the **Component Type** field, select **Drop Down List** or **Auto Complete List**.

The fields in the Validation window change dynamically, depending on your selection.

6. In the **Validated By** field, select **List**.

7. Click **New**, and then add a value.

The Add Validation Window opens.



8. Provide the information for the validation value as described in the following table.

Field Name	Description
Code	Underlying code for the validation value. The code is the value stored in the database or passed to any internal functions, and is rarely displayed.
Meaning	Displayed meaning for the validation value in the drop-down list or auto-complete.
Default	Default value for the list. This value is initially displayed in drop-down lists (it is not used for auto-completes). There can be only one default value per list.

9. To set the validation value as the default, select the **Default** checkbox.

The default option is only available for drop-down lists.

10. To add the value to the validation:
 - To add the value to the validation and close the Add Validation Value window, click **OK**.
 - To add the value and keep the Add Validation Value window open, click **Add**.
11. To save your changes and close the window, click **OK**.
12. To change the order in which validation values are listed, use the up and down arrow buttons. The sequence of the validation values determines the order that the values are displayed in the list.

You can copy existing values defined in other validations using the **Copy From** button. Click **Copy From** and query an existing list-validated validation and choose any of the validation values. Click **Add** or **OK** in the Copy From window and the selected value or values are added to the list.



Be careful when creating validations (drop-down lists and auto-complete fields) that are validated by lists. Each time the set of values changes, you must update the validation. Consider, instead, validating using an SQL query or PL/SQL function to obtain the values from a database table.

Configuring Dynamic List Validations

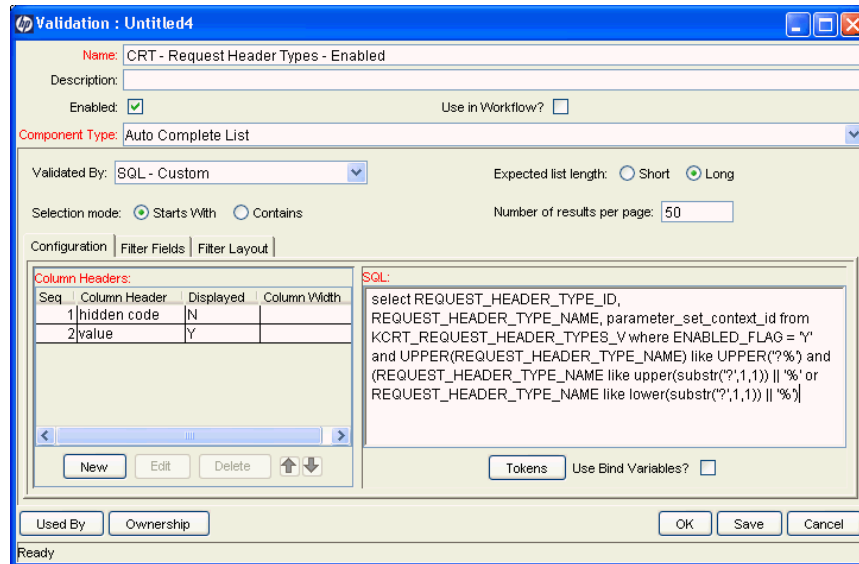
You can create validations that provide a dynamic list to the user. This is often a better approach than defining static list validations. Static list validations must be updated manually. Dynamic list validations can be constructed to automatically pick up and display changed values. A dynamic list validation can be created using a drop-down or an auto-complete component.

For example, XYZ Corporation needs a field validation that lists all users on their support team. They have a static validation that is validated by a list of users, but any time members join or leave the support team, the list must be manually updated.

XYZ decides to create a dynamic list validation. To do this, they create an auto-complete validation that is validated by an SQL statement. The SQL statement returns the names of all users who belong to the Support Team security group. If Security Team membership changes, the validation is automatically updated with the current values.

Configuring SQL Validations

You can use an SQL statement to generate the values in a validation. SQL can be used as a validation method for drop-down lists and auto-complete components. To define a dynamic list of choices, set a drop-down list or auto-completes to Validated By - SQL. Then in the **SQL** field, enter the Select statement that queries the necessary database.



If you are using an auto-complete component, you can define headers for the selected columns. These column headers are used in the window that opens if a value from an auto-complete is selected. Under the **Column Headers** table, click **New**. *Table 5-2* shows the fields that can be entered for a column header. If a column header is not defined for each column in an SQL query, a default name is used.

Table 5-2. Column headers

Field Name	Description
Column Header	Column name to display in the auto-complete window.
Display	Determines whether or not the column is visible. The first column is never visible and the second column is always visible.

For example, XYZ Corporation creates an auto-complete field that lists all users in the Engineering department. They choose to validate the list by SQL.

```
SELECT U.USER_ID, U.USERNAME, U.FIRST_NAME, U.LAST_NAME
FROM KNTA_USERS U, KNTA_SECURITY_GROUPS SG, KNTA_USER_SECURITY
US
WHERE SG.SECURITY_GROUP_ID = US.SECURITY_GROUP_ID AND US.USER_
ID = U.USER_ID
AND SG.SECURITY_GROUP_NAME = 'Engineering'
and UPPER(u.username) like UPPER('?%')
and (u.username like upper(substr('?',1,1)) || '%'
or u.username like lower(substr('?',1,1)) || '%')
order by 2
```

After a new user account is created and added to the Engineering security group, that user is automatically included in the auto-complete.



A validation may already exist that meets your process requirements. If it does, consider using that validation in your process. Also consider copying and modifying validations that are similar to the validation you want. For a complete list of validations that are delivered with the product, see [Viewing System Validations on page 73](#).

SQL Validation Tips

The following guidelines are helpful in writing an SQL statement for an SQL-validated validation:

- The SQL statement must query at least two columns. The first column is a hidden value that is never displayed, and is typically stored in the database or passed to internal functions. The second column is the value that is displayed in the field. All other columns are for information purposes and are only displayed in the auto-complete window. Extra columns are not displayed for drop-down lists.
- When something is typed into an auto-complete field, the values displayed in the auto-complete window are constrained by what was first typed in the field. Typically, the constraint is case-insensitive. To do this, you write the SQL statement to query only values that match text that was typed.

Before the auto-complete list is displayed, all question marks in the SQL statement are replaced by the text that the user typed. Typically, if the following conditions are added to the WHERE clause in an SQL statement, the values in the auto-complete window are constrained by what the user typed.

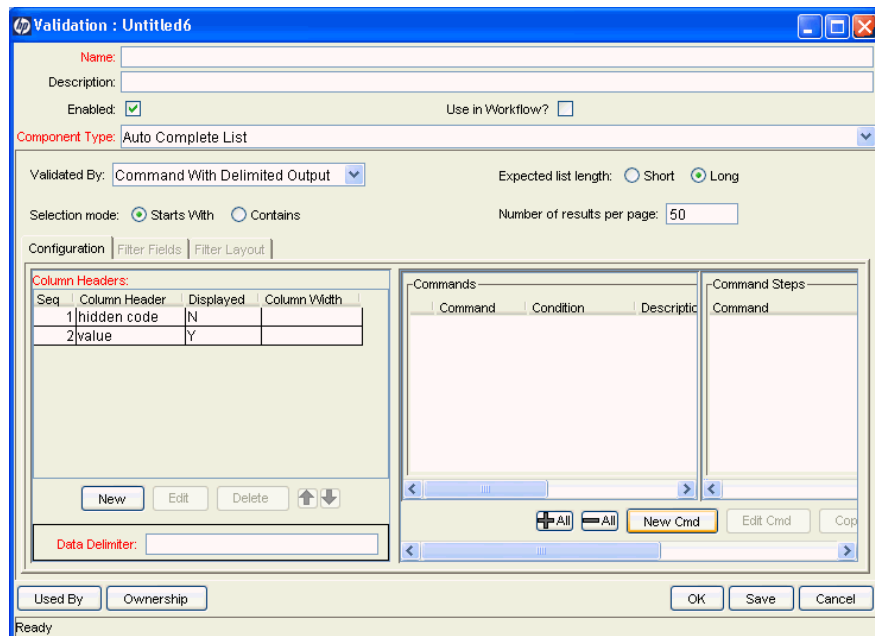
```
where UPPER(<displayed_column>) like UPPER('?%')
and (<displayed_column> like upper(substr('?',1,1)) || '%'
or <displayed_column> like lower(substr('?',1,1)) || '%')
```


Any column aliases included directly in the SQL statement are not used. The names of the columns, as displayed in auto-completes, are determined from the section **Column Headers**. Drop-down lists do not have column headers.

Command Validations

An auto-complete can contain command line executions that return and display a list of values. To define a dynamic list of choices, set an auto-complete to Validated By - Command with Delimited Output or Command with Fixed Width Output. Then enter commands the **Commands** section. See *Configuring the Auto-Complete Values* on page 93 for detailed instructions.

Figure 5-1. Auto-complete using command validation

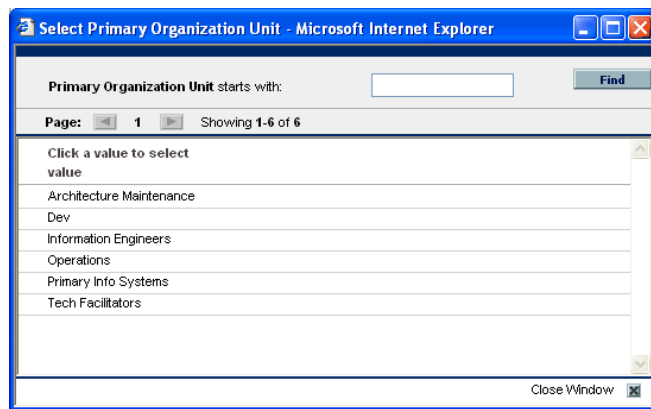


Configuring Short List Auto-Complete Field Validations

Auto-complete fields are used throughout the PPM Center to provide users with an efficient way to select field values from a set of valid choices. Auto-complete fields can be used for validations with a small or large number of choices. The auto-complete can be configured to behave differently depending on the expected number of values. For example, if you expect a large number of entries, the auto-complete window includes an interface that lets you page through your results. You can configure how the auto-complete feature for the field behaves. For example, you can configure the field to automatically complete entries that either start with or contain a text string.

Auto-complete fields configured to display a short list of entries, displaying all of the values on a single page. *Figure 5-2* shows the Select window for a short list auto-complete field.

Figure 5-2. Short list auto-complete



To configure a short list auto-complete field:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.
The Validations Workbench opens.
2. Create a new validation or open an existing validation.
The Validation window opens.
3. From the **Component Type** field, select **Auto Complete List**.
4. In the **Expected list length** field, select **Short**.

5. Click **Save**.



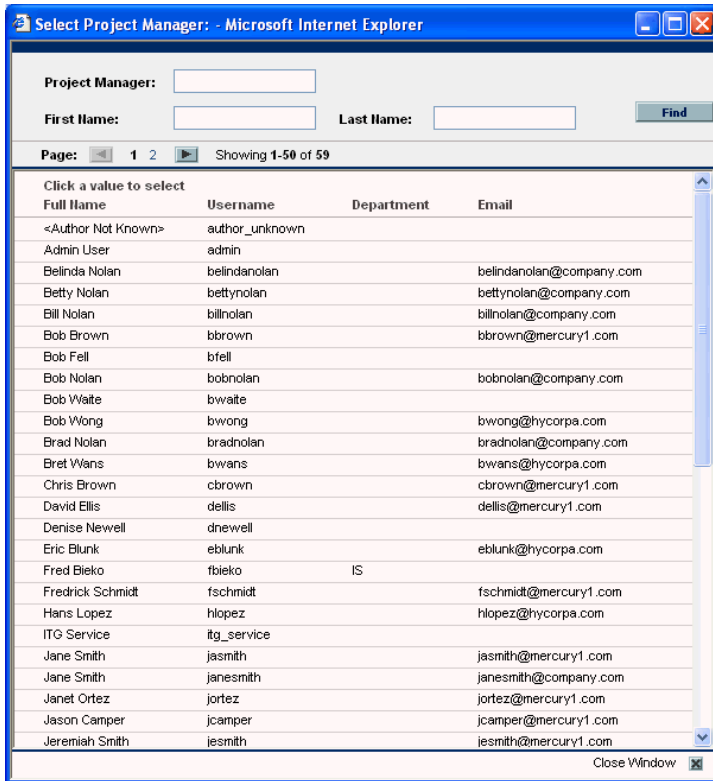
Auto-completes configured as short lists load all values when the window is opened. This can lead to a slower load time and an unfavorable user experience. For fields with many possible values, consider formatting the auto-complete using the long list format.

Configuring Long List Auto-Complete Field Validations

Auto-complete fields are used throughout the PPM Center to provide users with an efficient way to select field values from a set of valid choices. Auto-complete fields can be used for validations with a small or large number of choices. The auto-complete can be configured to behave differently depending on the expected number of values. For example, if you expect a large number of entries, the auto-complete window includes an interface that lets you page through your results. You can configure how the auto-complete feature for the field behaves. For example, you can configure the field to automatically complete entries that either start with or contain a text string.

Auto-complete fields configured to display a long list of entries, dividing the results between multiple pages. By default, 50 results are shown per page. End users can page through the results or further limit the results by specifying text in one of the available filter fields at the top of the page. *Figure 5-3 on page 84* shows the Select window for a long list auto-complete field.

Figure 5-3. Long list auto-complete



To configure a long list auto-complete field:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.
The Validations Workbench opens.
2. Open a validation.
The Validations window opens.
3. Create a new validation or open an existing validation.
The Validation window opens.
4. In the **Component Type** field, select **Auto Complete List**.
5. In the **Expected list length** field, select **Long**.

6. Click **Save**.

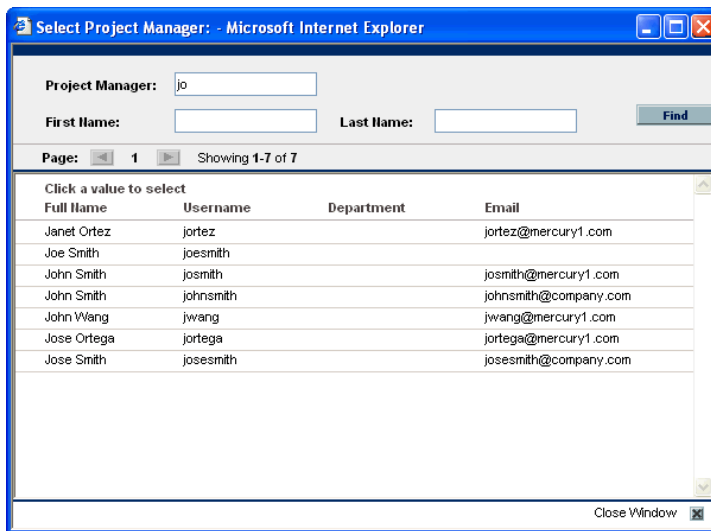
Auto-completes configured as long lists only load a limited set of values when the window is opened. For extremely long lists or lists that are at risk of loading slowly (for example the values are obtained from an alternate database), consider using the long list format.

All auto-completes that are validated by SQL - User must use the long list auto-complete format. This selection is automatically defaulted when the user selects SQL - User in the Validated By list in the Validation window.

Configuring Automatic Value Matching and Interactive Select Pages

This section provides instructions for configuring auto-complete fields to filter a list of possible values based on a matching character string. It also provides instructions for configuring the automatic value-limiting that occurs on the auto-complete's Select page. *Figure 5-4* shows an auto-complete field that has opened to display matching values.

Figure 5-4. Auto-complete field and matching values on the Select page



An Overview of Matching for "Starts with" or "Contains"

Auto-complete field behavior can be divided into the following areas:

- **Field behavior.** A user types a character in the field and type the `Tab` key. If an exact match is not available, the Select page opens.
- **Select page behavior.** For lists that are configured appropriately, when a user types a character or characters into the field at the top of the page, the results are automatically limited to display only matching entries.

For both the field and Select page behaviors, automatic value matching can be based on either “starts with” character matching or “contains” character matching. The following table summarizes this behavior:

Table 5-3. Automatic character matching field behavior

Character matching mode	Description of Behavior
Starts with	Type characters and then type the Tab key. The selection window opens and lists entries that begin with the specified characters.
Contains	Type characters, and then type the Tab key. The selection window opens and lists entries that contain the specified character string. This is the same behavior as a wild card search, which uses the% character at the beginning of the search text.

Table 5-4. Automatic character matching Select page behavior

Character Matching Selection Mode	Description of Behavior
Starts with	Type characters and the list is automatically filtered for entries that begin with those characters.
Contains	Type characters and the list is automatically filtered for entries that contain the character string.

To configure “starts with” matching from the auto-complete window to the selection window, add the following to the SQL WHERE clause:

```
UPPER(value) like UPPER('%?%') and (value like
upper(substr('?',1,1)) || '%' or value like
lower(substr('?',1,1)) || '%')
```

To configure “contains” matching from the auto-complete window to the selection window, add the following to the SQL WHERE clause:

```
UPPER(value) like UPPER('%?%') and (value like '%' ||
upper(substr('?',1,1)) || '%' or value like '%' ||
lower(substr('?',1,1)) || '%')
```

To configure “starts with” matching within the interactive selection window:

1. Open the auto-complete Validation window.
2. From the **Expected list length** field, select **Short**.

This feature is only available for short lists.

3. From the **Selection** option, select **Starts With**.

4. Save the validation.



This setting only controls the matching on the Select page. Matching in the auto-complete field is controlled by including specific clauses in the auto-complete's SQL. See above for details.

To configure “contains” matching within the interactive selection window:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Open the auto-complete Validation.

The Validation window opens.

3. From the **Expected list length** field, select **Short**.

This feature is only available for short lists.

4. From the **Selection** option, select **Contains**.

5. Save the validation.



This setting only controls the matching on the Select page. Matching in the auto-complete field is controlled by including specific clauses in the auto-complete's SQL. See above for details.

Configuration Tips

Consider the following tips as you configure the “starts with” versus “contains” functionality for auto-complete fields and the Select page:

- Auto-completes should be configured such that the field matching behavior works the same way as the Select page matching behavior. Specifically, if the auto-complete field uses the STARTING WITH clauses in the SQL, then the selection window should use the “Starts With” Selection Mode.
- Consider using the “Contains” selection mode for fields with multi-word values. For example, possible values for the request type auto-complete field are:

```
Development Bug  
Development Enhancement  
Development Issue  
Development Change Request  
IS Bug  
IS Enhancement  
IS Issue  
IS Change Request  
Support Issue
```

The “contains” mode can be useful here. The user knows that he must log a bug against one of the IS-supported financial applications. The user types “bug” into the auto-complete field and types the **Tab** key. The following items are returned:

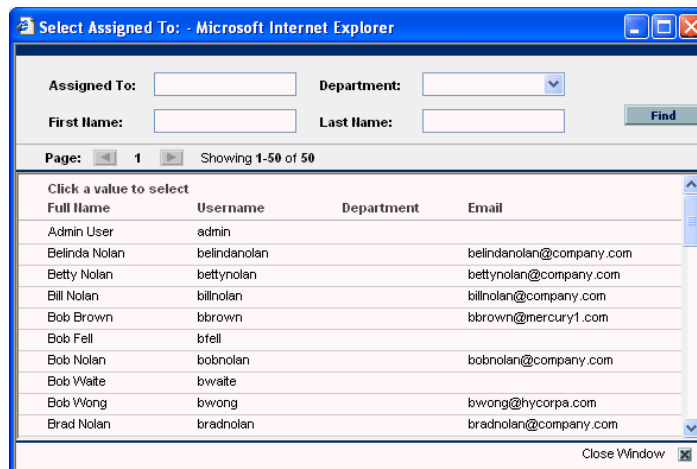
```
Development Bug
IS Bug
```

The user selects “IS Bug.” Without the “contains” feature enabled, typing “bug” would have returned the entire list. He might have also typed “Financial,” thinking that there might be a separate request type used for each type of supported application. This, too, would have returned the entire list. At that point, the user would be forced to try another “starts with” phrase or simply read the entire (potentially long) list.

Adding Search Fields to Long List Auto-Complete Validations

Auto-completes with a long list of values can be configured to display additional filter fields in the Select window. These fields can be used to search other properties than the primary values in the list. Users can enter values in the filter fields, and then click **Find** to display only the values that match the search criteria. *Figure 5-5* shows the Select window with additional filter fields.

Figure 5-5. Filter fields in the auto-complete select window



▶ Filter fields cannot be configured when validating your list by List, Command With Delimited Output, or Command With Fixed Width Output.

To add a filter field to the auto-complete validation:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Open the validation for the auto-complete.

Auto-complete validations must display **Auto Complete List** in the **Component Type** field.

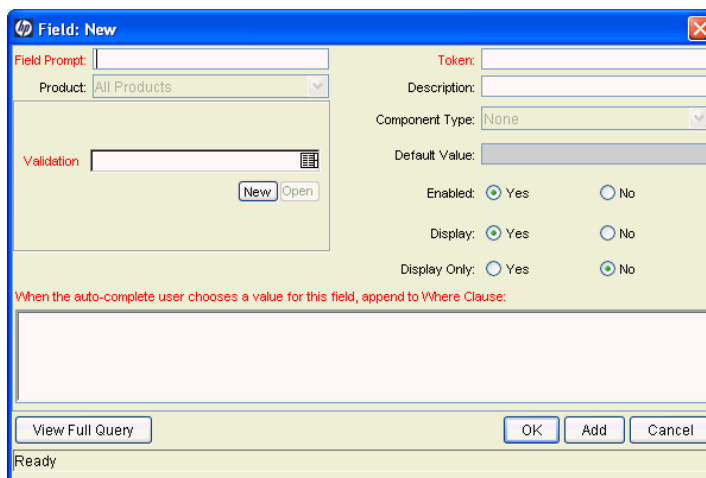
3. In the **Expected list length** field, select **Long**.

Only long formatted auto-completes can include filter fields.

4. Click the **Filter Fields** tab.

5. Click **New**.

The Field: New window opens.



6. Enter the required information and any optional information you want to provide.

Table 5-5 on page 90 lists the controls in the Field window.

Table 5-5. Fields in the Fields: New window

Field Name	Description
Field Prompt	Name displayed for the field in the auto-complete Select window.
Product	PPM Center product that uses the field.
Validation	Validation for the filter field. You can select any type of validation, except for auto-complete type validations. Valid values are appended to the WHERE clause in the SQL query that determines the ultimate auto-complete display.
New	Opens the Validation window, where you can construct a new validation for the filter field. Note that you cannot use an auto-complete type validation for the filter field.
Open	Opens the Validation window and displays the definition of the validation specified in the Validation field.
Token	Token for the field value. The token value is appended to the WHERE clause in the SQL query that determines the ultimate auto-complete display.
Description	Filter field description.
Component Type	Component type for the filter field, determined by its validation.
Default Value	Default value for the filter field, determined by its validation.
Enabled	Determines whether the filter field is enabled.
Display	Determines whether the filter field is visible to the user in the auto-complete's Select window.
Display Only	Determines whether the filter field is updatable. When Display Only is set to Yes , the field can not be updated.
When the auto-complete user chooses a value for this field, append to WHERE clause:	AND clause appended to the portlet's WHERE clause if the user enters a value in this filter field. Each filter field appends its term to the portlet query if the user enters a value in the Select window. For example, if the filter field uses the CRT-Priority-Enabled validation and a filter field token of P_PRIORITY, enter the following in this field: <code>AND R.PRIORITY_CODE = '[P.P_PRIORITY]'</code> Note: The value in this field must start with 'AND.'
View Full Query	Opens a window that displays the full query.

7. Click **OK**.

Filter fields offer users a powerful way to efficiently locate specific values in large lists. As you add filter fields to an auto-complete validation, consider the following:

- Ensure that the filter fields are functionally related to the listed values. For example, a validation that provides a list of request types can include a filter field for a specific Department associated with the request types.
- Consider reusing (copying) an auto-complete validation and modifying the filter fields to display a subset of the list. Use the **Displayed**, **Display Only**, and **Default** fields in the Filter Field window, to configure the auto-complete values to automatically limit the results.
- Performance can degrade if you join tables over database links. Use this functionality only for complex fields.

Configuring the Filter Field Layout

To modify the filter field layout:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.
The Validations Workbench opens.
2. Open the auto-complete validation that includes filter fields on the **Filter Fields** tab.
3. Click the **Filter Layout** tab.

The **Filter Layout** tab lists the primary field and all filter fields that have been defined for the auto-complete list. The primary field, which is named **Field Value**, holds the eventual selected value.

The screenshot shows a dialog box titled "Validation : CRT - Assigned To - Enabled". It has several sections:

- Name:** CRT - Assigned To - Enabled
- Description:** Returns a list of users filtered by knta_security_groups.used_by_requests_flag
- Enabled:** Use in Workflow?
- Component Type:** Auto Complete List
- Validated By:** SQL - User
- Expected list length:** Short Long
- Selection mode:** Starts With Contains
- Number of results per page:** 50
- Configuration | Filter Fields | Filter Layout** (selected)
- Table of Filter Fields:**

<input checked="" type="checkbox"/> Field Value	<input type="checkbox"/> Department
<input type="checkbox"/> First Name	<input type="checkbox"/> Last Name
<input type="checkbox"/> Direct Manager	<input type="checkbox"/> Title
<input type="checkbox"/> Member of Org Unit	<input type="checkbox"/> Location
<input type="checkbox"/> Member of Security Group	<input type="checkbox"/> Category
<input type="checkbox"/> Having Role	
- Field Width:** 1
- Component Lines:** [Control]
- Move Field:** [Up] [Down] [Left] [Right]
- Swap Mode:**
- Preview:** [Button]
- Used By:** [Button] **Ownership:** [Button]
- OK** [Button] **Save** [Button] **Cancel** [Button]
- Ready (Read-Only, Seed Data)**

4. Select the field that you would like to move.

To select more than one field, type the **shift** key while selecting a range. It is only possible to select a continuous set of fields (multiple selection using **ctrl** key is not supported).

5. Use the arrow pointers to move the fields to the desired location in the layout builder.



A field or a set of fields cannot be moved to an area where other fields already exist. The other field(s) must be moved out of the way first.

6. To switch the positions of two fields:

- a. Select the first field, and then and select the **Swap Mode** option.

An S is displayed in the checkbox area of the selected field.

- b. Double-click the second field that you want to reposition.

The two fields switch positions and the **Swap Mode** option is cleared.

7. To preview the layout, click **Preview**.

A window opens and shows the fields as they are to be displayed.



Rows with no fields are ignored. They are not displayed as blank lines.

Hidden fields are treated the same as blank fields, and do not affect the layout.

Configuring an Auto-Complete List of Users (Special Case)

User auto-completes or validations (Validated by: SQL-User) have the following three default filter fields:

- **Primary field** - this field takes the name of the auto-complete field
- **First name**
- **Last name**

The user auto-complete always appears in the long list format, which uses the paging interface to display the items. Additionally, user auto-completes display a different icon.

To configure a user auto-complete validation:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Create a new validation.

The Validation window opens.

3. From the **Component Type** field, select **Auto Complete List**.

4. From the **Validated By** field, select **SQL - User**.

5. Configure the SQL query that is to determine the users listed in the validation.

See *Configuring the Auto-Complete Values* on page 93 for details.

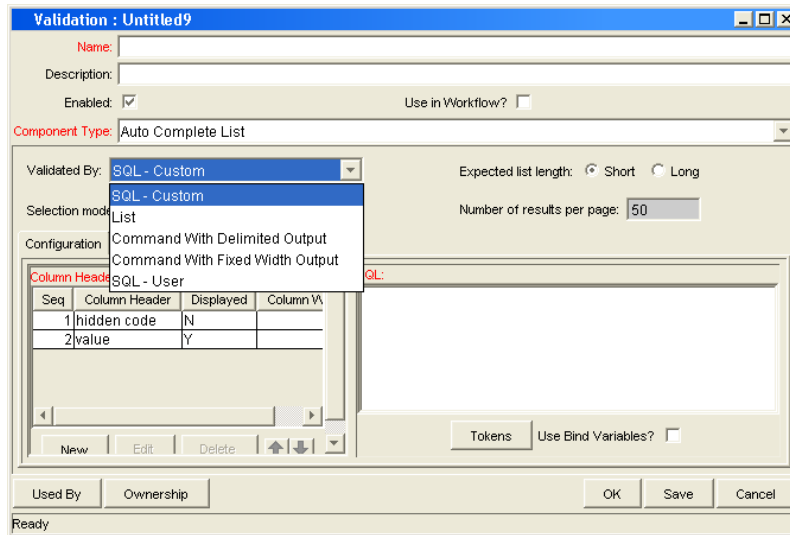
6. Click **Save**.

Configuring the Auto-Complete Values

The values in an auto-complete can be specified in the following ways. In the **Validate By** field, select one of the following:

- **List.** Used to enter specific values.
- **SQL.** Uses an SQL statement to build the contents of the list.
- **SQL - User.** Identical to SQL configuration, but includes a few additional preconfigured filter fields.
- **Command With Delimited Output.** Uses a system command to produce a character-delimited text string and uses the results to define the list.
- **Command With Fixed Width Output.** Uses a system command to produce a text file and parses the result on the basis of the width of columns, as well as the headers.

Figure 5-6. Auto-complete list



For more information on creating auto-completes validated by List or SQL, see *Configuring Static List Validations* on page 76 and *Configuring Dynamic List Validations* on page 78.

Configuring Validations by Commands With Delimited Output

Validations that are validated by commands with delimited output can be used to get data from an alternate source, and use that data to populate an auto-complete. This functionality provides additional flexibility when designing auto-completes.

Many enterprises need to use alternate sources of data within their applications. Examples of these sources are a flat file, an alternate database source, or output from a command line execution. Special commands may be used in conjunction with these alternate data sources, in the context of a validation, to provide a list of values.

To configure a validation by command with delimited output:

1. In the Validation Workbench, under **Validated By**, choose **Command With Delimited Output**, and then enter the delimiting character.
2. Under **New Command**, enter the command steps.

These can include PPM Center special commands. Include the special command `ksc_capture_output`, which captures and parses the delimited command output. If you place the `ksc_capture_output` special command between the `ksc_connect` and `ksc_disconnect` commands, the command is run on the remote system. Otherwise, the command is run locally on the PPM Server (like `ksc_local_exec`).

The following simple example uses a comma as the delimiter and includes the validation values red, blue, and green. The script places the validations into the `newfile.txt` file, and then uses the special command `ksc_capture_output` to process the text in the file.

```
ksc_begin_script[AS.PKG_TRANSFER_PATH]newfile.txt
red,red
blue,blue
green,green
ksc_end_script
ksc_capture_output cat[AS.PKG_TRANSFER_PATH]newfile.txt
```

Table 5-6 shows the Validation window for Command with Delimited Output.

Figure 5-7. Validation by command with delimited output

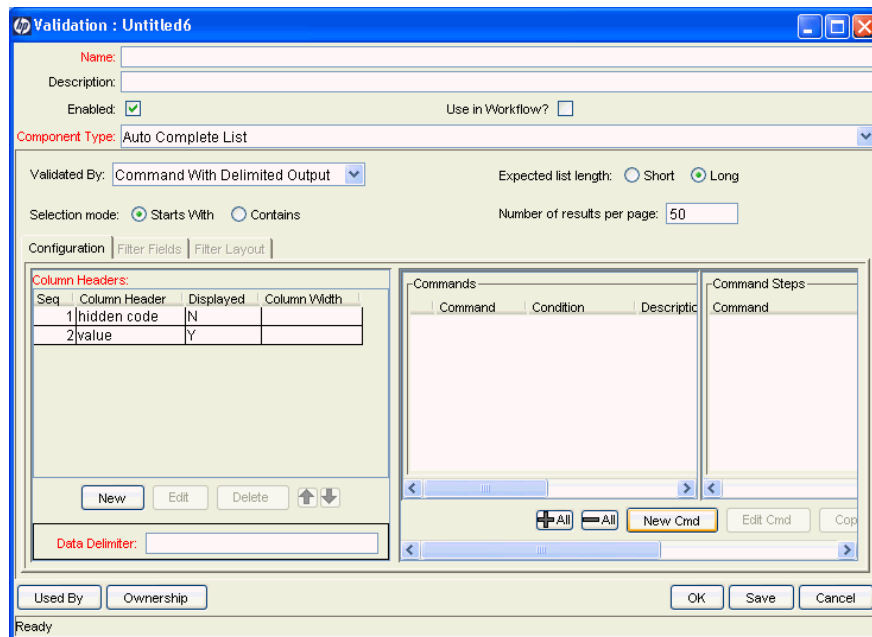


Table 5-6. Validation by command with delimited output

Field Name	Description
Commands	Field where new commands can be added to capture validation values.
Data Delimiter	Indicates the character or key by which the file is separated into the validation columns.

You can also define headers for the selected columns. These column headers are used in the window that opens when a value is selected from an auto-complete. To define a new header, click **New** in the **Column Header** section. *Table 5-7* shows the fields that you can enter for a column header. If you do not define a column header for each column in a command, a default header is used.

Table 5-7. Column headers

Field Name	Description
Column Header	The name of the column that is displayed in the auto-complete window.
Display	Determines whether the header is displayed in the validation.

Configuring Validations by Commands with Fixed-Width Output

Validations by Command with Fixed Width Output can be used to obtain data from an alternate source, and use that data to populate an auto-complete. This functionality provides additional flexibility when designing auto-completes.

Many enterprises need alternate data sources within their applications. Example sources are a flat file, an alternate database source, or output from a command-line execution. You can use special commands together with these alternate data sources, in the context of a validation, to provide a list of values on the fly.

In the Validation Workbench, under **Validated By**, choose **Command With Fixed Width Output** and enter the width information. Then, under **New Command**, type the command steps. These can include special commands. Include the special command `ksc_capture_output` in your commands. This command captures and parses the delimited command output. If you place the `ksc_capture_output` between `ksc_connect` and `ksc_disconnect`, the command is run on the remote system. Otherwise, it is run locally on the PPM Server (as `ksc_local_exec` is).

The following example includes the validations red, blue, and green. The column width is set to 6. The script places the validations into the `newfile.txt` file.

```
ksc_begin_script[AS.PKG_TRANSFER_PATH]newfile.txt
red red
blue blue
green green
ksc_end_script
ksc_capture_output cat[AS.PKG_TRANSFER_PATH]newfile.txt
```

Figure 5-8. Validation by command with fixed width output

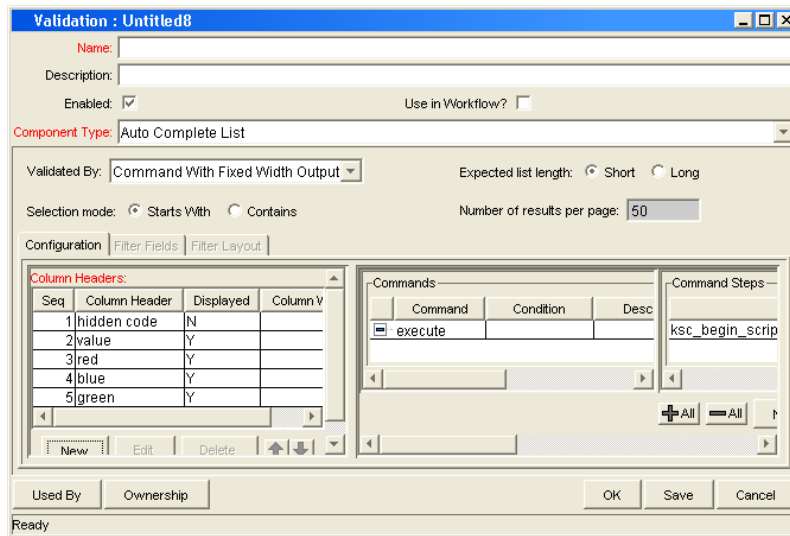


Table 5-8. Validation by command with fixed width output

Field Name	Description
Commands	Field where new commands can be added to capture validation values.

Headers can also be defined for the columns selected. These column headers are used in the window that opens when a value is selected from an auto-complete. To define a new column header, click **New** in the **Column Header** section. [Table 5-9](#) shows the fields can be entered for a column header. If a column header is not defined for each column in a command, a default name is used.

Table 5-9. Column headers

Field Name	Description
Column Header	The name of the column that is displayed in the auto-complete.
Display	Whether or not the column is displayed. The first column is never displayed and the second column is always displayed.
Column Width	The number of characters in each column of the output generated as a result of the command.

Configuring User-Defined Multi-Select Auto-Complete Fields

A number of auto-completes in the PPM Workbench have been pre-configured to allow users to open a separate window for selecting multiple values from a list. Users can also define custom auto-completes to have multi-select capability when creating various product entities.

The user-defined multi-select capability is supported for:

- User data fields
- Report type fields
- Request type fields
- Project template fields

The user-defined Multi-Select capability is not supported for:

- Request header types
- Object types

In order to use this feature when creating a new entity, users must:

- Select a validation for the new entity that has **Auto-Complete List** as the Component Type. This enables the **Multi-Select Enabled** field in the Field: New window.
- In the Field: New window, users must click **Yes** for the **Multi-Select Enabled** option.

The step-by-step procedure for defining multi-select capability in user data, report type, request type, or project template fields is very similar. The procedure for enabling this capability for request type field is shown below as an example.

To define a multi-select auto-complete for a request type:

1. From the PPM Workbench shortcut bar, select **Demand Mgmt > Request Types**.

The Request Types Workbench opens.

2. Open a Request Type.

The Request Type window opens.

3. Click **New**.

The Field: New window opens.

4. Select a validation of type **Auto-Complete List** from the **Validation** field.

The **Multi-Select Enabled** option is enabled.

5. To the right of **Multi-Select Enabled**, click **Yes**.

The Possible Conflicts window opens and displays a warning not to use a multi-select auto-complete for advanced queries, workflow transitions and reports. If this field is not to be used in advanced queries, workflow transitions or reports, click **Yes**.

6. Configure any other optional settings for the new request type.

7. Click **OK**.

The field is now enabled for multi-select auto-complete.

Example of Token Evaluation and Validation by Command with Delimited Output

The validation functionality can be extended to include field-dependent token evaluation. You can configure validations to change dynamically, depending on the client-side value entered in another field.

To use field dependent token evaluation, you must configure a validation in conjunction with an object type, request type, report type, project template, or user data definition. Consider the following example of how to set up an object type using field-dependent tokens.

1. Generate a validation and set the following parameters as shown here:

- a. Name: **demo_client_token_parsing**
- b. Component Type: **Auto Complete List**
- c. Validated By: **Command With Delimited Output**
- d. Data Delimiter: | (bar)
- e. Command
 - Command: **Validate_from_file**
 - Steps

```
ksc_connect_source_server SOURCE_ENV="Your Env"  
ksc_capture_output cat [P.P_FILENAME]  
ksc_exit
```

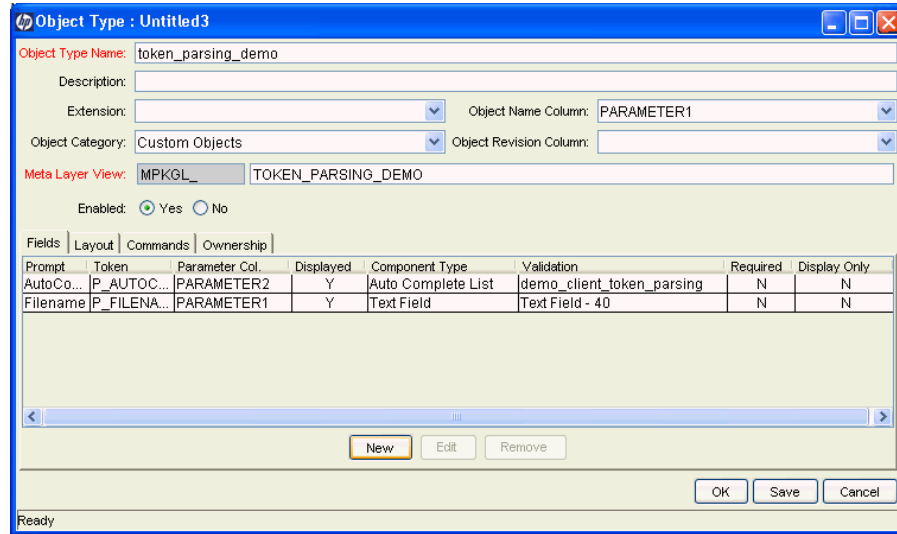
The screenshot shows a configuration window titled "Validation : demo_client_token_parsing". The "Name" field is set to "demo_client_token_parsing". The "Component Type" is "Auto Complete List". The "Validated By" is "Command With Delimited Output". The "Expected list length" is "Long". The "Selection mode" is "Starts With". The "Number of results per page" is "50". The "Data Delimiter" is "|". The "Commands" section shows a command named "Validate_from_file" with the following steps: "ksc_connect_source_server", "ksc_capture_output cat [P.P_FILENAME]", and "ksc_exit".

Seq	Column Header	Displayed	Column Width
1	hidden code	N	
2	value	Y	

Command	Condition	Command Steps
Validate_from_file		ksc_connect_source_server ksc_capture_output cat [P.P_FILENAME] ksc_exit

When called, this validation connects to an environment called “Your Env” and retrieves data from a file specified by the token P_FILENAME. The file resides in the directory specified in the Base Path in the Environment window.

2. Generate an object type named `token_parsing_demo`.



a. Generate a new field with the following parameter settings:

- Name: Filename
- Token: P_FILENAME
- Validation: Text Field - 40

b. Generate a new field with the following parameter settings:

- Name: AutoComp
- Token: P_AUTOCOMP
- Validation: demo_client_token_parsing (the validation defined in [step 2](#))

3. For this example to return any values in the auto-complete, a file must be generated in the directory specified in the Base Path in the Environment Detail of “Your Env” environment. Generate a file named `parse_test1.txt` that contains the following delimited data:

```
DELIMITED_TEXT1|Parameter 1  
DELIMITED_TEXT2|Parameter 2  
DELIMITED_TEXT3|Parameter 3  
DELIMITED_TEXT4|Parameter 4
```

The object type `token_parsing_demo` can now use this token evaluation.

To test the configuration sample:

1. From the PPM Workbench shortcut bar, select **Deploy Mgmt > Packages**.

The Packages Workbench opens.

2. Open a new package.

3. In the Package window, select a workflow, and click **Add Line**.

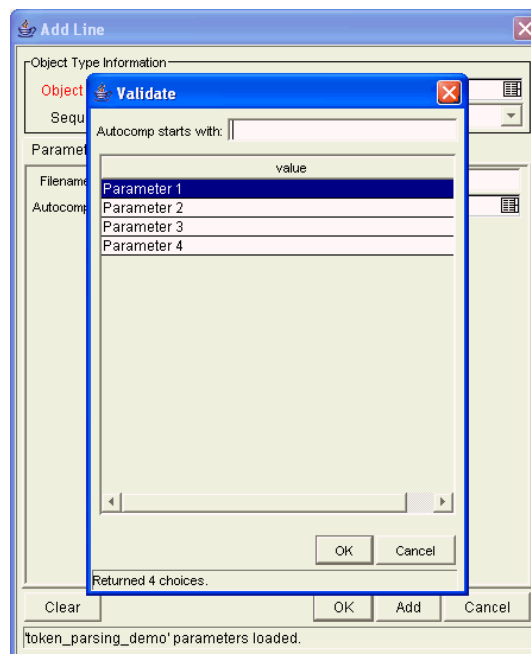
The Add Line window opens.

4. In the **Object Type** field, select `token_parsing_demo`.

The **Filename** and **AutoComp** fields are displayed:

5. In the **Filename** field, type `parse_test1.txt`.

6. In the **AutoComp** field, select `parse_test1.txt`.



Configuring Text Field Validations

Text fields displayed on a single line. Text fields can be configured to display the data according to a certain format. For example, you can configure a text field to accept and format a ten-digit telephone number or display a specific number of decimal places for a percentage.

To create a text field validation:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Open a validation.

The Validations window opens.

3. In the **Name** field, type the validation name.

4. In the **Component Type** field, select **Text Field**.

5. In the **Data Mask** field, select the data mask that represents the format you want for the field data.

For more information, see *Text Data Masks for Validations* on page 104.

6. Configure the selected data mask. (Optional)

For information about data masks, see *Text Data Masks for Validations* on page 104.

7. To view the results of your data mask settings:

- a. In the **Sample Input** field, enter a value to preview based on your settings.
- b. Click **Format**.

The Formatted Output window displays the results.

8. Click **OK**.

Text Data Masks for Validations

PPM Center includes a number of preconfigured data masks that can be used when creating text field validations. Each of these data masks can be configured to meet your specific data requirements. The data masks delivered with PPM Center are described in [Table 5-10](#).

Table 5-10. Data mask formats (page 1 of 2)

Data Mask	Description
Alphanumeric	Field allows all alphanumeric characters. You can specify the maximum field length for fields using this validation.
Alphanumeric Uppercase	Field allows alphanumeric characters and formats all characters as uppercase text. You can specify the maximum field length for fields using this validation.
Numeric	<p>Field allows only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none">■ Range of values (maximum and minimum) that the field accepts■ Whether to display a zero if the field contains no data■ Whether to use a group separator such as a comma to display large numbers■ Negative number format■ Maximum number of decimal places <p>For more detailed information, see Configuring the Numeric Data Mask on page 105.</p>
Currency	<p>Field is used to display currency data and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none">■ Range of valid values (maximum and minimum) for the field■ Whether a zero is displayed if no data is entered■ Whether a group separator such as a comma is used to display large numbers■ Negative number display■ Number of decimal places <p>For more detailed information, see Configuring the Currency Data Mask on page 107.</p>

Table 5-10. Data mask formats (page 2 of 2)

Data Mask	Description
Percentage	<p>Field is used to display percentages and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none"> ■ Range of valid values (maximum and minimum) for the field ■ Whether a zero is displayed if no data is entered ■ Whether a group separator such as a comma is used to display large numbers ■ Negative number display ■ Number of decimal places <p>For more detailed information, see Configuring the Percentage Data Mask on page 109.</p>
Telephone	<p>Field is used to display telephone numbers and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none"> ■ Format - specify the number of digits included, and the delimiter to be used between groups of numbers. For example, you can specify dashes (-) or periods (.) between numbers (555-555-5555 or 555.555.5555). ■ Maximum and minimum number of digits <p>For more detailed information, see Configuring the Telephone Data Mask on page 110.</p>
Custom	<p>Field allows a range of custom inputs. You can customize the field to accept digits, letters, spaces, and custom delimiters. For more detailed information, see Configuring a Custom Data Mask on page 112.</p>

Configuring the Numeric Data Mask

The numeric data mask allows only numeric characters. When creating a validation using this data mask, you can specify the following field characteristics:

- Range of values (maximum and minimum) accepted
- Whether to display a zero if the field contains no data
- Whether to use a group separator such as a comma to display large numbers
- Negative number format
- Maximum number of decimal places accepted

Figure 5-9 shows the fields that you can configure for this data mask. Table 5-11 provides descriptions of these configurable fields.

Figure 5-9. Validation window for the numeric data mask

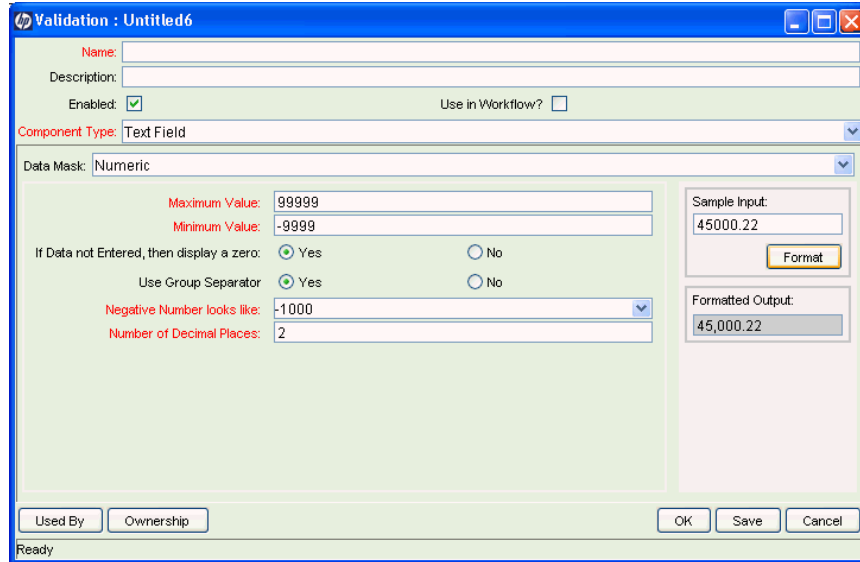


Table 5-11. Fields for configuring the numeric data mask for text fields (page 1 of 2)

Field Name	Description
Maximum Value	Largest value allowed for this field. You can specify a positive or negative number.
Minimum Value	Smallest accepted value for the field. You can specify positive or negative number.
If Data not Entered, then display a zero	Determines whether a field with no data displays a zero.

Table 5-11. Fields for configuring the numeric data mask for text fields
(page 2 of 2)

Field Name	Description
Use Group Separator	Determines if the field uses a group separator (such as a comma) to divide characters within large numbers (for example, whether 1000000 is displayed as 1,000,000). The default character used as the separator depends on the locale setting for the machine, but you can use the Regional Settings window in the PPM Workbench (select Edit > Regional Settings) to change the delimiter.
Negative Number looks like	Used to select one of the following four formats for the display of negative numbers: <ul style="list-style-type: none"> ■ (1000)—parentheses and black text ■ (1000)—parentheses and red text ■ -1000—minus character (-) and black text ■ -1000—minus character (-) and red text
Number of Decimal Places	Determines the maximum number of decimal places used to display values.

Configuring the Currency Data Mask

The currency data mask allows only numeric characters and is used to display currency data. When creating a validation using this data mask, you can specify the following characteristics:

- Range of values (maximum and minimum) allowed for this field
- Whether or not a zero is displayed when data is not entered into the field
- Whether a group separator such as a comma is used to display large numbers
- Negative number display
- Number of decimal places

Figure 5-10 on page 108 shows the fields that you can configure for this data mask. *Table 5-12* on page 108 provides descriptions of these fields.

Figure 5-10. Validation window for the currency data mask

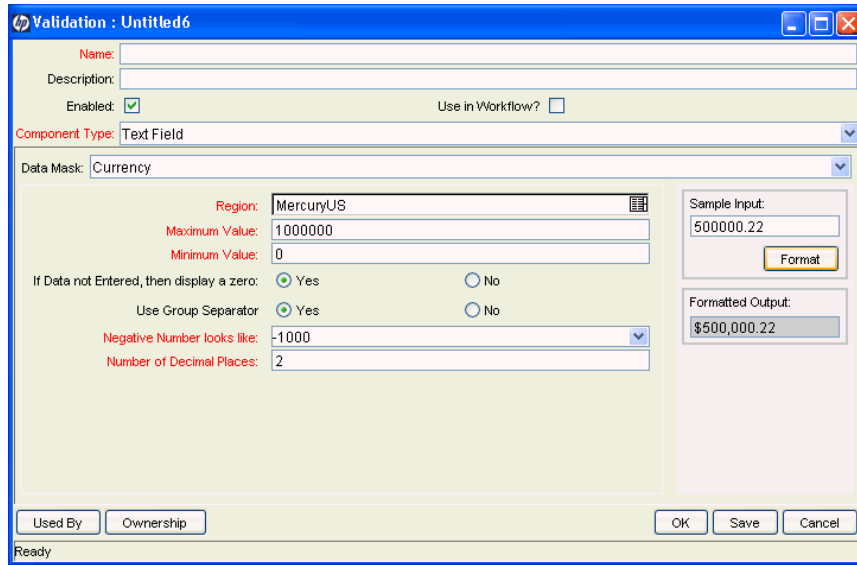


Table 5-12. Fields configuring the currency data mask for text fields

Field Name	Description
Maximum Value	Largest value allowed for this field. You can enter a positive or negative number.
Minimum Value	Smallest value allowed for this field. You can enter a positive or negative number.
If Data not Entered, then display a zero	Determines whether a field that contains no data displays a zero.
Use Group Separator	Determines if the field should use a group separator (such as a comma) to divide characters within large numbers. For example: 1000000 versus 1,000,000. The character used for the separator defaults based on the machine's local, but can be configured in the Regional Settings window in the PPM Workbench. Select Edit > Regional Settings to access this window.
Negative Number looks like	Determines the text used to display negative numbers. The four possible options are: <ul style="list-style-type: none"> ■ (1000)—parentheses and black text ■ (1000)—parentheses and red text ■ -1000—minus character (-) and black text ■ -1000—minus character (-) and red text
Number of Decimal Places	The maximum number of decimal places displayed.

The `INSTALLATION_CURRENCY` server parameter determines the currency symbol displayed in the field and the position of the text in the field. For example, the following parameter setting specifies the dollar currency sign, and right-aligned text:

```
INSTALLATION_CURRENCY=$;RIGHT
```

For help with changing this setting, contact your system administrator.

Configuring the Percentage Data Mask

The percentage data mask allows only numeric characters and is used to display percentages. When creating a validation using this data mask, the following characteristics can be specified:

- Range of values (maximum and minimum) allowed for this field
- Whether or not a zero is displayed when data is not entered into the field
- Whether a group separator such as a comma is used to display large numbers
- Negative number format
- Maximum number of decimal places

Figure 5-11 shows the fields that you can configure for this data mask, and *Table 5-13* on page 110 provides descriptions of these fields.

Figure 5-11. Validation window for the percentage data mask

The screenshot shows a dialog box titled "Validation : Untitled6". It contains the following fields and options:

- Name:** [Empty text field]
- Description:** [Empty text field]
- Enabled:** (checked)
- Use in Workflow?:** (unchecked)
- Component Type:** Text Field
- Data Mask:** Percentage
- Maximum Value:** 100
- Minimum Value:** 0
- If Data not Entered, then display a zero:** Yes, No
- Use Group Separator:** Yes, No
- Negative Number looks like:** -1000
- Number of Decimal Places:** 2
- Sample Input:** 50.22
- Formatted Output:** 50.22%
- Buttons:** Used By, Ownership, OK, Save, Cancel

Table 5-13. Fields configuring the percentage data mask for text fields

Field Name	Description
Maximum Value	Largest value allowed for this field. You can specify a positive or negative value.
Minimum Value	Smallest value allowed for this field. You can specify a positive or negative value.
If Data not Entered, then display a zero	Determines whether the field displays a zero if no value is entered.
Use Group Separator	Determines whether a group separator such as a comma is used to display large numbers (for example, 1000000 versus 1,000,000). The default character used for the separator is based on the local machine setting, but you can modify the setting from the PPM Workbench, in the Regional Settings window. To open this window in the PPM Workbench, select Edit > Regional Settings .
Negative Number looks like	Determines the text used to display negative numbers. The four possible options are: <ul style="list-style-type: none"> ■ (1000)—parentheses and black text ■ (1000)—parentheses and red text ■ -1000—minus character (-) and black text ■ -1000—minus character (-) and red text
Number of Decimal Places	Determines the maximum number of decimal places accepted.

Configuring the Telephone Data Mask

Use the percentage data mask to specify telephone number display. As you create a validation using this data mask, you can specify the following characteristics for the field:

- Specify the number of digits to include, and the delimiter to use between groups of numbers. For example, you can specify dashes (-) or periods (.) between numbers (555-555-5555 or 555.555.5555).
- Maximum and minimum number of digits.

Figure 5-12 on page 111 shows the fields that you can configure for this data mask, and *Table 5-14 on page 111* provides descriptions of these fields.

Figure 5-12. Validation window for the telephone data mask

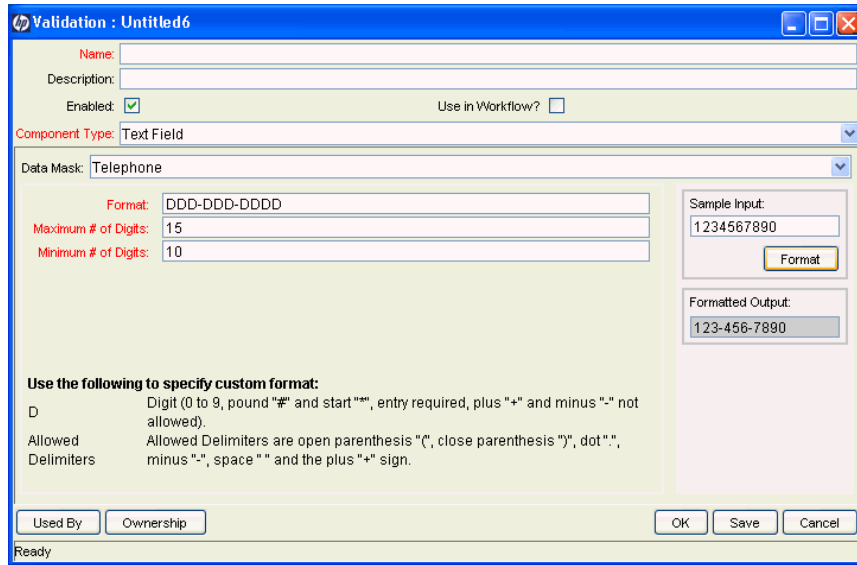


Table 5-14. Fields configuring the telephone data mask for text fields

Field Name	Description
Format	<p>The rule that determines how digits are formatted, including the use of spaces or delimiters. The format definition can include the following delimiters:</p> <ul style="list-style-type: none"> ■ Parentheses () ■ Period (.) ■ Dash (-) ■ Space ■ Plus character (+) <p>For telephone format examples, see Table 5-15.</p>
Maximum # of Digits	The maximum number of digits that the field accepts.
Minimum # of Digits	The minimum number of digits that the field accepts.

Table 5-15. Sample telephone data mask formats

Format Rule	User Input	Output
D-DDD-DDD-DDDD	15555555555	1-555-555-5555
DDD DDD DDDD	5555555555	555 555 5555
(DDD) DDD-DDDD	5555555555	(555) 555-5555

Special behavior applies to the extra characters if you define a format that lets users enter a range of number of characters. Extra characters are always grouped with the first set of characters. For example, if you configure the telephone data mask with a minimum of ten characters and a maximum of 15 characters, the results are as follows:

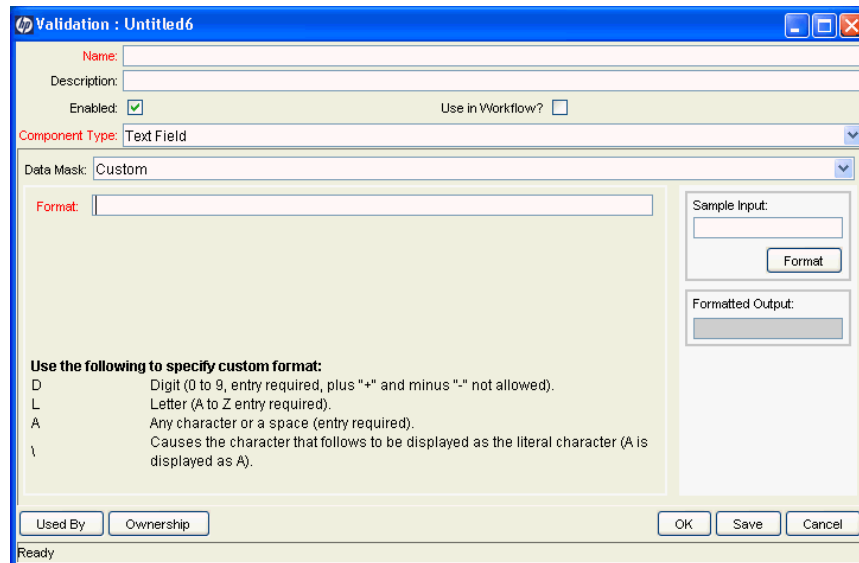
```
Format: DDD-DDD-DDDD  
Min: 10  
Max: 15  
Input: 1234567890  
Output: 123-456-7890  
Input 2: 12345678901  
Output 2: 1234-567-8901
```

Configuring a Custom Data Mask

You can define a custom data mask that allows a range of inputs, and specify the format for the input. You can customize the field to accept numeric values, alphabetic characters, spaces, and custom delimiters.

Figure 5-13 shows the fields that you can configure for this data mask.

Figure 5-13. Validation window for the custom data mask



To configure a custom format, in the **Format** field, type a combination of the following symbols.

- Use D to specify that the user must enter a numeric value between 0 and 9.
- Use L to specify that the user must enter an alphabetic character between A and Z.
- Use A to specify that the user must type a character or space.
- Use a \ (backslash) to specify that the next character is to be displayed as the literal character. For example: “\A” is displayed as “A”.

Table 5-16 lists two examples of custom formatting.

Table 5-16. Sample custom data mask formats

Format Rule	User Input	Output
DDD\ -DD\ -DDDD	555555555	555-55-5555
AA\ -DDD	BC349	BC-349

Configuring Directory Chooser Validations

The **Directory Chooser** field can be used to select a valid directory from an environment. HP Deployment Management connects to the first source environment on a workflow and allows navigation through the directory structure and the selection of a directory from the list.

As you implement the Directory Chooser, note the following:

- You can only use the **Directory Chooser** field on an object type.
- On every object type that a Directory Chooser is chosen, it is also necessary to have a field whose token is `P_FILE_LOCATION` and whose validation is DLV - File Location. The possible values for this field are **Client** and **Server**. If **Client** is chosen, the Directory Chooser connects to the Client Base Path of the source environment. If **Server** is chosen, the Directory Chooser connects to the Server Base Path of the source environment.

Configuring File Chooser Validations

A **File Chooser** field can be used by object types to select a valid file from an environment. HP Deployment Management connects to the first source environment on a workflow and provides the ability to view all files within a specific directory and select one from the list.

On every object type that a File Chooser is chosen, it is necessary to define the following fields:

- The first is a field for the file location for the directory chooser, described in the previous section.
- The second is a field whose token is P_SUB_PATH. This field is the directory from which the file is selected and is usually a directory chooser field.

Figure 5-14. Validation window for static environment override in file chooser

Table 5-17. File chooser field

Field Name	Description
Base File Name Only	Defines whether the base file name only (without its suffix) or the complete name is displayed.
Environment Override Behavior	Used to select files from a specific environment other than the default environment.

The Environment Override Behavior drop-down list contains three options: **Default Behavior**, **Static Environment Override**, and **Token-Based Environment Override**.

Static Environment Override lets you override one environment at a time. The fields for static environment override are shown in *Figure 5-14* on page 114 and described in *Table 5-18*.

Table 5-18. Static environment override

Field Name	Description
Overriding Environment	Selects the environment to be overridden.
Overriding Server Basepath	The server basepath of the environment can be overridden.
Overriding Client Basepath	The client basepath of the environment are overridden.

Token-based Environment Override provides the ability to select a token that is to resolve to the overriding environment. The fields for **Token-based Environment Override** are shown in *Figure 5-15* and defined in *Table 5-19* on page 116.

Figure 5-15. Validation window for token-based environment override in file chooser

Table 5-19. Token-based environment override

Field Name	Description
Environment Token	Select the token that is to resolve to the overriding environment.
Overriding Server Basepath	Specify a basepath to override the server basepath of the environment to be resolved by the token.
Overriding Client Basepath	Client basepath of the environment that is to be resolved by the token can be overridden.

Configuring Date Field Validations

Date fields can accept a variety of formats. The current date field validations are separated into two categories: all systems and systems using only the English language. These formats are described in [Table 5-20 on page 117](#).

Table 5-20. Date field formats

Field Name	Systems	Description
Date Format	All	<p>Formats for the date part of the field. Choices are:</p> <ul style="list-style-type: none"> ■ Long. January 2, 1999 ■ Medium. 02-Jan-99 ■ Short. 1/2/99 ■ None. Date is not displayed.
Date Format	English Only	<p>Available formats for the date section of the field are:</p> <ul style="list-style-type: none"> ■ MM/DD/YY (06/16/99) ■ DD-MON-YY (16-Jun-99) ■ MONTH DD, YYYY (June 16, 1999) ■ Day, Month DD, YYYY (Monday, June 16, 1999) ■ DD-MON (16-JUN, defaults to current year) ■ DD-MON-YYYY (16-JUN-1999) ■ MM-DD-YYYY (06-16-1999) ■ MM-DD-YY (06-16-99) ■ DD (Defaults to the current month and year) ■ MM/DD (06/16, defaults to current year) ■ MM/DD/YYYY (06/16/1999)
Time Format	All	<p>Available formats for the time section of the field are:</p> <ul style="list-style-type: none"> ■ Long. 12:00:00 PM PST ■ Medium. 12:00:00 PM ■ Short. 12:00 PM ■ None. Time is not displayed.

Configuring 1800 Character Text Areas

Standard text areas are either 40 or 200 characters. You can, however, create a Text Area validation with a character length of 1800.

To create a validation with a character length of 1800:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Search for **Text Area - 1800**.

3. In the results tab, select **Text Area - 1800**.
4. Click **Copy**.
5. Rename the validation.

You can use the Text Area validation (1800 characters long) as you define a custom field in the product.

► You can only create a text field or area of length 40, 200, 1800, or 4000.

Configuring the Table Component

The table component is used to enter multiple records into a single field on a request. You can configure the table component to include multiple columns of varied data types. This component also supports rules for populating elements within the table and provides functionality for capturing column totals.

For example, XYZ Corporation creates a request type to request quotes and parts for hardware. Each entry of this type has four elements: Product, Quantity, Price, and Total. XYZ creates a Table Component field called **Hardware Information** to collect this information.

When the user logs a request for new hardware, the request displays the **Hardware Information** field. The user opens the Hardware Information window and selects a product, which triggers a rule to populate the fields in the **Price** and **Total** columns. He submits the request, which now contains all of the information required to successfully order the hardware.

Figure 5-16. Hardware information window

Hardware Information

Select the Product and Quantity of the items you wish to order.

Seq	Products	Quantity	Price	Total
<input type="checkbox"/> 1	PC	3	1200	3600
<input type="checkbox"/> 2	PC	2	1200	2400

You can only add fields of this component to request types, request header types, and request user data.

Configuring Table Components

To create a table component field:

1. From the PPM Workbench shortcut bar, select **Configuration > Validations**.

The Validations Workbench opens.

2. Click **New Validation**.

The Validation window opens.

3. From the **Component Type** list, select **Table Component**.

The screenshot shows a 'Validation : Untitled6' dialog box. It contains the following elements:

- Name:** [Empty text field]
- Description:** [Empty text field]
- Enabled:**
- Use in Workflow?:**
- Component Type:** Table Component (dropdown menu)
- User Instructions:** [Empty text area]
- Meta Layer View:** MREQ_ [Text field]
- Table Columns:** A tabbed interface with 'Form Layout' and 'Rules' sub-tabs. The 'Table Columns' sub-tab is active, showing a table with the following headers: Column Seq., Column Header, Column Token, Parameter Col., Enabled, Component Type, Validation, Editable, and Required. The table body is currently empty.
- Buttons:** New, Edit, Remove, Used By, Ownership, OK, Save, and Cancel.

4. Enter a validation name and description.
5. Enter any user instructions to display at the top of the table entry page.
6. Create the table columns, as follows:

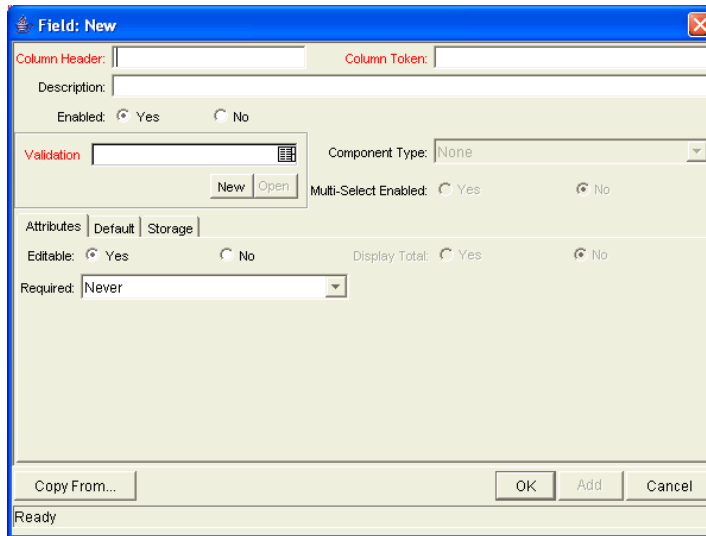
- a. Click **New** in the **Table Columns** tab.

The Field window opens.

- b. Define the type of information to store that column.

This may require that you create a validation for the column.
You cannot use file attachments in a table component column.

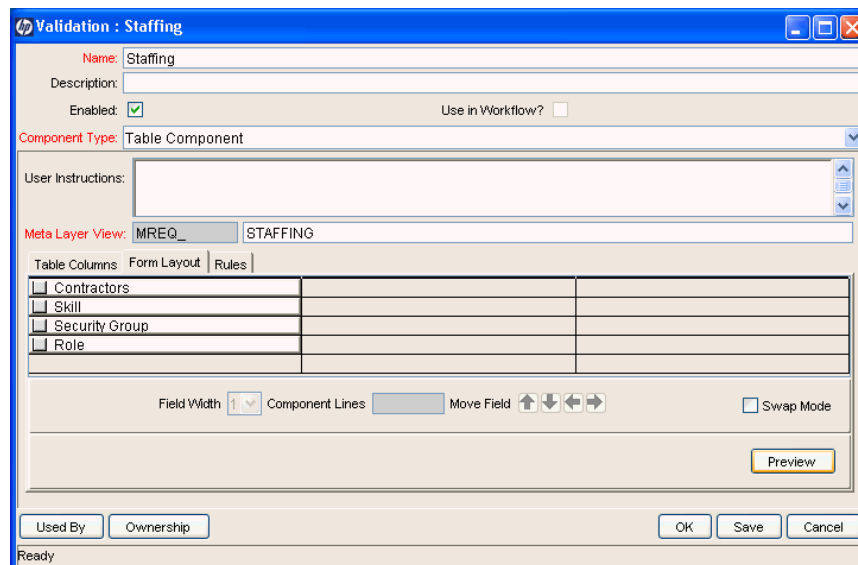




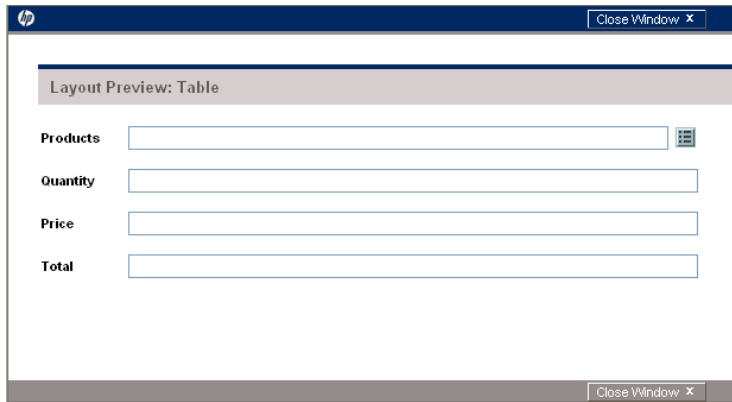
- c. Specify the attributes (editable or required) and any default behavior.
- d. To save the column information and add another column, click **Add**.
- e. To close the Field window after you finish adding columns, click **OK**.

7. Configure the form layout, as follows:

- a. Click the **Form Layout** tab.
- b. To move a field, select it, and then use the arrow pointers to change its position in a given direction.



- c. To see the layout you configured, click **Preview**.



The preview loads a window in the PPM Workbench, but the table component itself is only available to those using the standard (HTML) interface.

8. To set up rules for advanced defaulting behavior or calculating column totals, configure any required table logic, as follows:
 - a. Click the **Rules** tab.
 - b. Click **New**.
 - c. Create a rule.



For detailed instructions on how to create a rule, see [Configuring Table Rules on page 121](#).

9. Click **OK**.

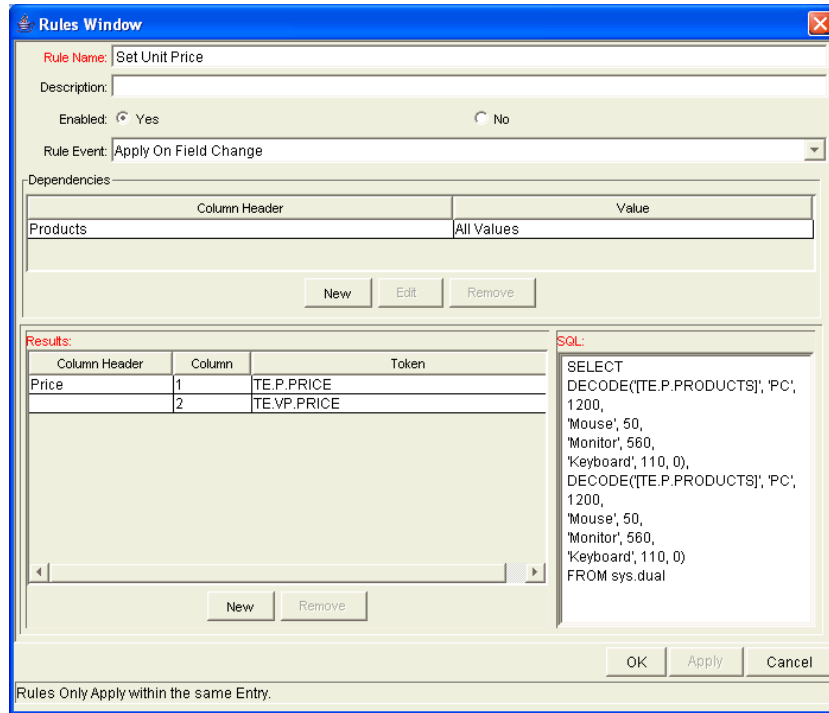
The new **Table Component** field can be included on a request type, request header type or request user data field.

Configuring Table Rules

Table rules are configured in the same manner as advanced request type rules. Essentially, you can configure fields (columns) in the table to default to certain values based on an event or value in another field in the table. Because the table component rules are configured using an SQL statement, you are given enormous flexibility for the data that is populated in the table cells.

Table rules are configured using the **Rules** tab on the Validation window.

Figure 5-17. Rules window accessed from the Rules tab



Example of Using a Table Component on an Order Form

The following example illustrates the table component rules functionality.

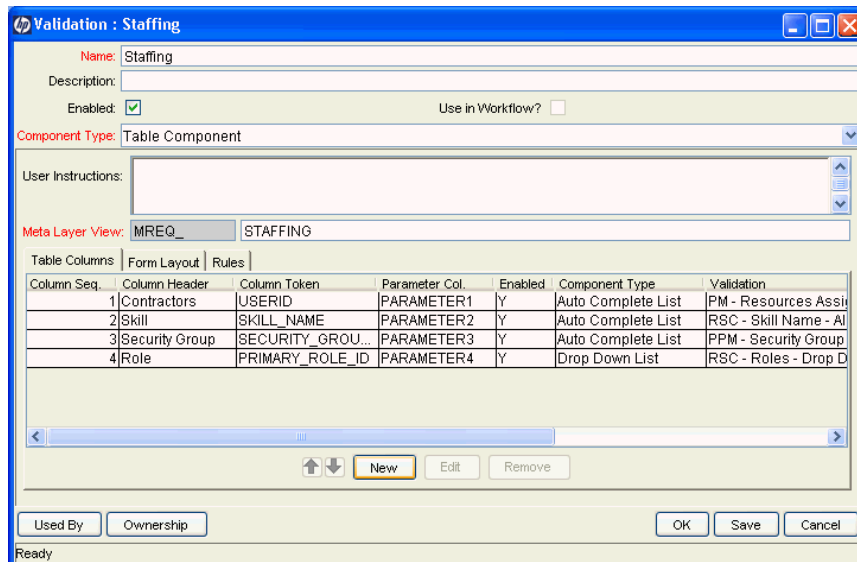
XYZ Corporation uses a request for creating and tracking employee computer hardware equipment orders. XYZ has included a table component field on their request type for gathering the order information. When the employee selects a Product, the Unit Price is automatically updated. Then, when they update the Quantity, the total line cost is automatically calculated and displayed in the table.

To enable this functionality, XYZ first has to configure a new validation with the following specifications:

Table 5-21. Example, table component validation settings

Setting	Value / Description
Validation Name	Product Order Information
Component Type	Table Component
Column 1	Column Header = Products Column Token = PRODUCTS Validation = Auto-complete with the following list values: PC, MOUSE, MONITOR, KEYBOARD
Column 2	Column Header = Quantity Column Token = QUANTITY Validation = Numeric Text Field
Column 3	Column Header = Price Column Token = PRICE Validation = Numeric Text Field
Column 4	Column Header = Total Column Token = TOTAL Validation = Numeric Text Field

Figure 5-18. Validations window



After you define the columns for the validation, you can set up the rules.

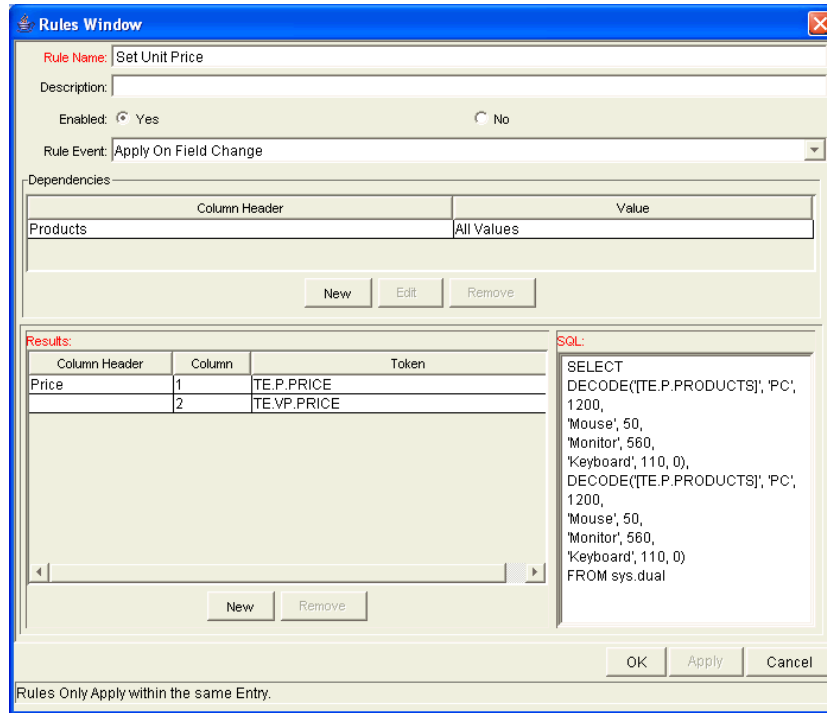
Example of Setting Unit Prices

XYZ Corporation uses the rule described in *Table 5-22* and shown in *Figure 5-19* on page 125 to set the default unit price based on the product selected.

Table 5-22. Example - Set Unit Price rule settings

Setting	Value / Description
Rule Name	Set Unit Price
Rule Event	Apply on Field Change
Dependencies	Column = Products All Values = Yes
Results	Column Header = Price
SQL	<pre>SELECT DECODE('[TE.P.PRODUCTS]', 'PC', 1200, 'Mouse', 50, 'Monitor', 560, 'Keyboard', 110, 0), DECODE('[TE.P.PRODUCTS]', 'PC', 1200, 'Mouse', 50, 'Monitor', 560, 'Keyboard', 110, 0) FROM sys.dual</pre>

Figure 5-19. Rules window



Example of Calculating Totals

XYZ Corporation uses the following rule to set the calculate and display the total line price in the Total column based on the values in the **Products** and **Quantity** fields.

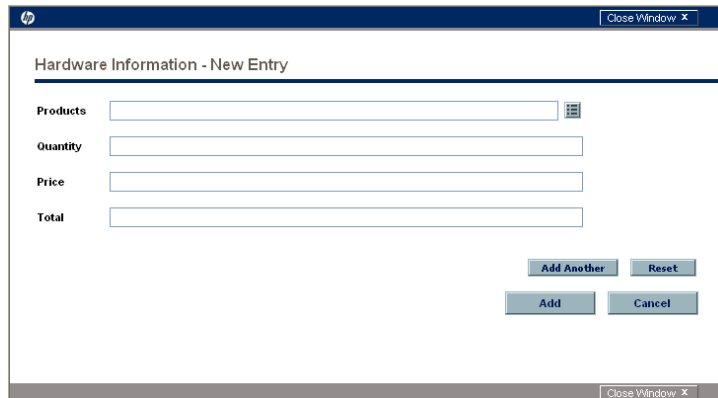
Table 5-23. Example - Calculate Total rule settings

Setting	Value / Description
Rule Name	Calculate Total
Rule Event	Apply on Field Change
Dependencies	Column = Price [All Values = Yes] Column = Quantity [All Values = Yes]
Results	Column Header = Total
SQL	<pre>SELECT [TE.P.PRICE] * [TE.P.QUANTITY], [TE.P.PRICE] * [TE.P.QUANTITY] from sys.dual</pre>

Using Table Components

Add a field to a request type that is validated by this table component validation. After a user opens the window to enter information, the table rules are applied to each row created.

Figure 5-20. Hardware information window



Using Tokens in Table Components

Each column in the table component has an associated token. You can use these tokens in the same manner as other field tokens, such as for commands, notifications, or advanced field defaulting. For detailed information about referencing tokens related to table components, see [Chapter 4, Using Tokens](#), on page 47.

Calculating Column Totals

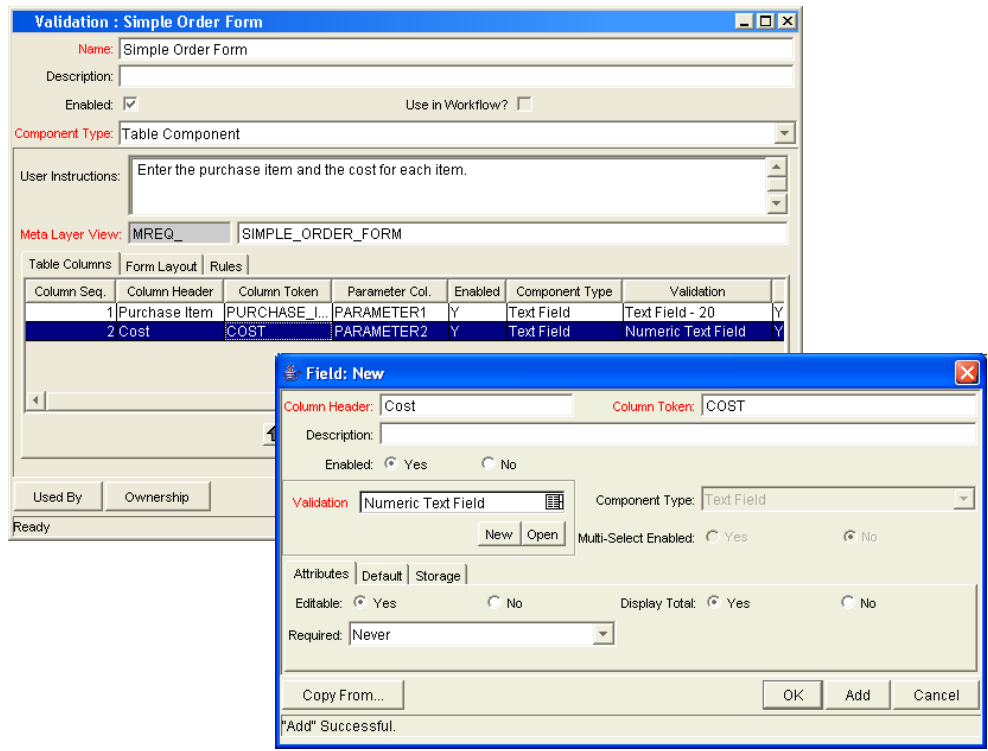
You can configure columns that are validated by a number to calculate the total for that column. This is configured in the validation's Field window. The following example illustrates how to configure a column to calculate and display the column total.

XYZ Corporation uses a request for creating and tracking simple employee equipment orders. XYZ has included a table component field on their request type for gathering the order information. Employees enter the Purchase Items and Cost for each item. The table component automatically calculates the total cost for the Cost column.

XYZ creates a validation with the following settings:

- Component Type = **Table Component**
- Column 1 = Purchase Item (text field)
- Column 2 = Cost (number). In the Field window for the Cost column, select Display Total = **Yes**. The **Display Total** field is only enabled if the field's validation is a number.

Figure 5-21. Sample validation for a Simple Order table component.



XYZ Corporation includes adds a field to their Order request type that uses this validation. If a user creates a request of that type, he can click the table component icon next to the field to open the order form. The total for the Cost column is displayed at the bottom of the table.

Figure 5-22. Sample table component displaying a column total

Simple Order Form
Enter the purchase item and the cost for each item.

Seq	Purchase Item	Cost
<input type="checkbox"/> 1	Flatscreen Monitor	1800
<input type="checkbox"/> 2	Cable	40
<input type="checkbox"/> 3	Monitor Switch	80
Total		1920

Check All Clear All Add Edit Copy Delete Done Cancel

A Tokens

In This Appendix:

- *Overview of Tokens*
- *Application Server Tokens*
- *Budget Tokens*
- *Contact Tokens*
- *Distribution Tokens*
- *Document Management Tokens*
- *Environment Tokens*
 - *Environment > Dest Env Tokens*
 - *Environment > Dest Env > App Tokens*
 - *Environment > Dest Env > Env Tokens*
 - *Environment > Env Tokens*
 - *Environment > Env > App Tokens*
 - *Environment > Env > Env Tokens*
 - *Environment > Source Env Tokens*
 - *Environment > Source Env > App Tokens*
 - *Environment > Source Env > Env Tokens*
- *Command Tokens*
- *Financial Benefit Tokens*
- *Notification Tokens*
- *Organization Unit Tokens*
- *Package Tokens*
 - *Package > Package Line Tokens*
 - *Package > Pending Reference Tokens*
- *Package Line Tokens*
- *Program Tokens*
- *Project Tokens*
- *Project Detail Tokens*
- *Release Tokens*
 - *Release > Distribution Tokens*
- *Request Tokens*
 - *Request > Pending Reference Tokens*
 - *Request > Field Tokens*
- *Request Detail Tokens*
 - *Request Detail > Field Tokens*
- *Resource Pool Tokens*
- *Security Group Tokens*
- *Skill Tokens*

- *Staffing Profile Tokens*
 - *Step TXN (Transaction) Tokens*
 - *System Tokens*
 - *Task Tokens*
 - *Tasks > Pending Tokens*
 - *Time Management Notification Tokens*
 - *User Tokens*
 - *Validation Tokens*
 - *Validation > Value Tokens*
 - *Workflow Tokens*
 - *Workflow > Workflow Step Tokens*
 - *Workflow Step Tokens*
 - *Request > Field Tokens*
 - *CMBD Application Tokens*
 - *Demand Management SLA Tokens*
 - *Demand Management Scheduling Tokens*
 - *MAM Impact Analysis Tokens*
 - *Portfolio Management Asset Tokens*
 - *Portfolio Management Project Tokens*
 - *Portfolio Management Proposal Tokens*
 - *Program Issue Tokens*
 - *Program Reference Tokens*
 - *Project Issue Tokens*
 - *Project Reference Tokens*
 - *Project Risk Tokens*
 - *Project Scope Change Tokens*
 - *Quality Center Defect Information Tokens*
 - *Quality Center Information Tokens*
 - *Resource Management Work Item Tokens*
-

Overview of Tokens

PPM Center uses variables to facilitate the creation of general objects that can be applied to a variety of contexts. These variables are called tokens.

The Token Builder generates tokens in the explicit entity format by providing a list of possible values. When such a list is available, the Context Value auto-complete field at the bottom of the Token Builder is enabled and the appropriate prefix is assigned. You then select the token from the list of provided tokens.

Application Server Tokens

Table A-1. Application server tokens

Prefix	Tokens	Description
AS	PKG_TRANSFER_PATH	Temporary directory used for files during command executions.

Other application server properties tokens are generated from the parameters in the `server.conf` file. For a description of each server parameter, see the *System Administration Guide and Reference*.

Budget Tokens

Table A-2. Budget tokens (page 1 of 2)

Prefix	Tokens	Description
BGT	ACTIVE_FLAG	Active flag for the budget.
BGT	BUDGET_ID	ID of the budget (defined in the table KCST_BUDGETS).
BGT	BUDGET_IS_FOR_ENTITY_NAME	Entity name (work plan, program, or org unit) to which the budget is linked.
BGT	BUDGET_IS_FOR_ID	ID of the work plan/program/org unit to which the budget is linked.
BGT	BUDGET_IS_FOR_NAME	Name of the work plan/program/org unit to which the budget is linked.

Table A-2. Budget tokens (page 2 of 2)

Prefix	Tokens	Description
BGT	BUDGET_NAME	Name of the budget.
BGT	BUDGET_ROLLS_UP_TO_ID	ID of the budget into which this budget rolls up.
BGT	BUDGET_ROLLS_UP_TO_NAME	Name of the budget into which this budget rolls up.
BGT	BUDGET_URL	URL used to view this budget.
BGT	CREATED_BY	Username of the user who created the budget.
BGT	CREATION_DATE	Date the budget was created.
BGT	DESCRIPTION	Budget description.
BGT	END_PERIOD	Budget end period.
BGT	INITIATION_REQ	Budget initiation request ID.
BGT	PERIOD_SIZE	Budget period size.
BGT	REGION	Region associated with the budget.
BGT	START_PERIOD	Budget start period.
BGT	STATUS_CODE	Budget status code.
BGT	STATUS_NAME	Budget status name.

Contact Tokens

Table A-3. Contact tokens

Prefix	Tokens	Description
CON	COMPANY	Company ID for which the contact works.
CON	COMPANY_NAME	Name of the company for which the contact works.
CON	CONTACT_ID	Contact ID (defined in the table KCRT_CONTACTS).
CON	CREATED_BY	ID of the user who created the contact.
CON	CREATION_DATE	Date the contact was created.
CON	EMAIL_ADDRESS	Email address of the contact.
CON	FIRST_NAME	First name of the contact.
CON	FULL_NAME	Full name of the contact.
CON	LAST_NAME	Last name of the contact.
CON	LAST_UPDATED_BY	ID of the user who last updated the contact.
CON	LAST_UPDATE_DATE	Date the contact was last updated.
CON	PHONE_NUMBER	Phone number of the contact.
CON	USERNAME	Contact username (if applicable). This can be the username for an external system, and not PPM Center.
CON	USER_ID	UserID of the contact, if the contact is a PPM Center user.

Distribution Tokens

Table A-4. Distribution tokens

Prefix	Tokens	Description
DIST	CREATED_BY	ID of the user who created the distribution.
DIST	CREATED_BY_USERNAME	PPM Center username for the user who created the distribution.
DIST	DESCRIPTION	Release description.
DIST	DISTRIBUTION_ID	Distribution ID (defined in table KREL_DISTRIBUTION).
DIST	DISTRIBUTION_NAME	Distribution name.
DIST	DISTRIBUTION_STATUS	Distribution workflow status.
DIST	FEEDBACK_FLAG	Determines whether the distribution has fed back a specified value to the package lines being distributed.
DIST	FEEDBACK_VALUE	Value to be returned to the original package lines.
DIST	LAST_UPDATED_BY	ID of the user who last updated the distribution.
DIST	LAST_UPDATED_BY_USERNAME	PPM Center username for the user who last updated the distribution.
DIST	LAST_UPDATE_DATE	Date the distribution was last updated.
DIST	RELEASE_ID	ID of the release that created this distribution.
DIST	RELEASE_NAME	Name of the release that created this distribution.
DIST	WORKFLOW	workflow used to process the distribution.

Document Management Tokens

Table A-5. Document Management tokens

Prefix	Tokens	Description
DMS	DOC_LINK	Resolves to a URL which, when clicked, opens the latest version of the document. Forces user authentication before the document is delivered.
DMS	DOC_HISTORY	Resolves to a URL which, when clicked, displays a view of the version history of the document. Forces user authentication before the information is delivered.
DMS	AUTHOR	Resolves to the author field stored with the document.
DMS	DESCRIPTION	Resolves to the description field stored with the document.
DMS	LAST_CHECK_IN_DATE	Resolves to the timestamp of the last check-in.
DMS	LAST_CHECKED_IN_BY_NAME	Resolves to the full name of the PPM Center user who added or last checked in the document.
DMS	LAST_CHECKED_IN_BY	Resolves to the ID of the PPM Center user who added or last checked in the document.

Environment Tokens

If any PPM Center Extensions are installed, there are more environment tokens with the prefix “AC.” For information about these tokens, see the PPM Center Extensions documentation.

Environment > Dest Env Tokens

Table A-6. Environment > Dest Env tokens (page 1 of 3)

Prefix	Tokens	Description
DEST_ENV	CLIENT_BASE_PATH	Base (root) path of the client.
DEST_ENV	CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
DEST_ENV	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
DEST_ENV	CLIENT_NAME	DNS name or IP address of the client computer.
DEST_ENV	CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
DEST_ENV	CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
DEST_ENV	CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
DEST_ENV	CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
DEST_ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.
DEST_ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
DEST_ENV	CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
DEST_ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
DEST_ENV	CREATED_BY	ID of the user who created the environment.
DEST_ENV	CREATION_DATE	Date the environment was created.

Table A-6. Environment > Dest Env tokens (page 2 of 3)

Prefix	Tokens	Description
DEST_ENV	DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DEST_ENV	DATABASE_TYPE	Validation value code of the database type.
DEST_ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
DEST_ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DEST_ENV	DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
DEST_ENV	DB_NAME	DNS name or IP address of the database server.
DEST_ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
DEST_ENV	DB_PASSWORD	Password PPM Center uses to log on to or access the database. This value is encrypted.
DEST_ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
DEST_ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
DEST_ENV	DB_VERSION	Database version (for example, 8.1.7).
DEST_ENV	DESCRIPTION	Environment description.
DEST_ENV	ENABLED_FLAG	Flag that indicates whether the environment is enabled and available for use in workflows.
DEST_ENV	ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
DEST_ENV	ENVIRONMENT_NAME	Environment name.

Table A-6. Environment > Dest Env tokens (page 3 of 3)

Prefix	Tokens	Description
DEST_ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.
DEST_ENV	LAST_UPDATE_DATE	Date the environment was last updated.
DEST_ENV	LOCATION	Environment location.
DEST_ENV	MSSQL_DB_NAME	For a Microsoft® SQL Server database type, the database name used to access the database from the command line.
DEST_ENV	SERVER_BASE_PATH	Base (root) path of the server.
DEST_ENV	SERVER_CON_PROTOCOL	Protocol used to connect to this server.
DEST_ENV	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
DEST_ENV	SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
DEST_ENV	SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
DEST_ENV	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
DEST_ENV	SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
DEST_ENV	SERVER_NAME	DNS name or IP address of the server computer.
DEST_ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
DEST_ENV	SERVER_PASSWORD	Password PPM Center uses to log on to or access the server. This value is encrypted.
DEST_ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
DEST_ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
DEST_ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Environment > Dest Env > App Tokens

Table A-7. Environment > Dest Env > App tokens (page 1 of 3)

Prefix	Tokens	Description
DEST_ENV.APP	APP_CODE	Short name (code) for the application.
DEST_ENV.APP	APP_NAME	Descriptive name for the application.
DEST_ENV.APP	CLIENT_BASE_PATH	Application-specific base (root) path of the client.
DEST_ENV.APP	CLIENT_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the client.
DEST_ENV.APP	CLIENT_USERNAME	Application-specific username PPM Center uses to log on to or access the client.
DEST_ENV.APP	CLIENT_CON_PROTOCOL	Application-specific protocol used to connect to this client.
DEST_ENV.APP	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connection protocol.
DEST_ENV.APP	CLIENT_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this client.
DEST_ENV.APP	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
DEST_ENV.APP	CREATED_BY	ID of the user who created the application.
DEST_ENV.APP	CREATION_DATE	Date the application was created.
DEST_ENV.APP	DB_LINK	For Oracle database type, the application-specific database link from the PPM Center schema to the database schema for the environment.
DEST_ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.

Table A-7. Environment > Dest Env > App tokens (page 2 of 3)

Prefix	Tokens	Description
DEST_ENV.APP	DB_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the database.
DEST_ENV.APP	DB_USERNAME	Application-specific username or schema name that PPM Center uses to log on to or access the database.
DEST_ENV.APP	DESCRIPTION	Application description.
DEST_ENV.APP	ENABLED_FLAG	Flag that indicates whether the application is enabled and available for selection in package lines.
DEST_ENV.APP	ENVIRONMENT_APP_ID	ID of the application in the table KENV_ENVIRONMENT_APPS.
DEST_ENV.APP	ENVIRONMENT_ID	ID of the environment with which the application is associated.
DEST_ENV.APP	ENVIRONMENT_NAME	Name of the environment with which the application is associated.
DEST_ENV.APP	LAST_UPDATED_BY	ID of the user who last updated the application.
DEST_ENV.APP	LAST_UPDATE_DATE	Date the application was last updated.
DEST_ENV.APP	SERVER_CON_PROTOCOL	Application-specific protocol used to connect to this server.
DEST_ENV.APP	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
DEST_ENV.APP	SERVER_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this server.
DEST_ENV.APP	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
DEST_ENV.APP	SERVER_BASE_PATH	Application-specific base (root) path of the server.

Table A-7. Environment > Dest Env > App tokens (page 3 of 3)

Prefix	Tokens	Description
DEST_ENV.APP	SERVER_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the server.
DEST_ENV.APP	SERVER_USERNAME	Application-specific username PPM Center uses to log on to or access the server.
DEST_ENV.APP	WORKBENCH_ENVIRONMENT_URL	URL of the environment window in the PPM Workbench.

Environment > Dest Env > Env Tokens

Table A-8. Environment > Dest Env > Env tokens (page 1 of 4)

Prefix	Tokens	Description
DEST_ENV.ENV	CLIENT_BASE_PATH	Base (root) path of the client.
DEST_ENV.ENV	CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
DEST_ENV.ENV	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
DEST_ENV.ENV	CLIENT_NAME	DNS name or IP address of the client computer.
DEST_ENV.ENV	CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
DEST_ENV.ENV	CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
DEST_ENV.ENV	CLIENT_PASSWORD	Encrypted Password that PPM Center uses to log on to or access the client.
DEST_ENV.ENV	CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
DEST_ENV.ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.

Table A-8. Environment > Dest Env > Env tokens (page 2 of 4)

Prefix	Tokens	Description
DEST_ENV.ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
DEST_ENV.ENV	CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
DEST_ENV.ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
DEST_ENV.ENV	CREATED_BY	ID of the user who created the environment.
DEST_ENV.ENV	CREATION_DATE	Date the environment was created.
DEST_ENV.ENV	DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DEST_ENV.ENV	DATABASE_TYPE	Validation value code of the database type.
DEST_ENV.ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
DEST_ENV.ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DEST_ENV.ENV	DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
DEST_ENV.ENV	DB_NAME	DNS name or IP address of the database server.
DEST_ENV.ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
DEST_ENV.ENV	DB_PASSWORD	Encrypted password that PPM Center uses to log on to or access the database.
DEST_ENV.ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net listens for remote SQL connections on the database server.

Table A-8. Environment > Dest Env > Env tokens (page 3 of 4)

Prefix	Tokens	Description
DEST_ENV.ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
DEST_ENV.ENV	DB_VERSION	Database version (such as 8.1.7).
DEST_ENV.ENV	DESCRIPTION	Environment description.
DEST_ENV.ENV	ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
DEST_ENV.ENV	ENVIRONMENT_ID	The ID of the environment in the table KENV_ENVIRONMENTS.
DEST_ENV.ENV	ENVIRONMENT_NAME	Environment name.
DEST_ENV.ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.
DEST_ENV.ENV	LAST_UPDATE_DATE	Date the environment was last updated.
DEST_ENV.ENV	LOCATION	Environment location.
DEST_ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
DEST_ENV.ENV	SERVER_BASE_PATH	Base (root) path of the server.
DEST_ENV.ENV	SERVER_CON_PROTOCOL	Protocol used to connect to this server.
DEST_ENV.ENV	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
DEST_ENV.ENV	SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
DEST_ENV.ENV	SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
DEST_ENV.ENV	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.

Table A-8. Environment > Dest Env > Env tokens (page 4 of 4)

Prefix	Tokens	Description
DEST_ENV.ENV	SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
DEST_ENV.ENV	SERVER_NAME	DNS name or IP address of the server computer.
DEST_ENV.ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
DEST_ENV.ENV	SERVER_PASSWORD	Password PPM Center uses to log on to or access the server. This value is encrypted.
DEST_ENV.ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
DEST_ENV.ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
DEST_ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Environment > Env Tokens

Table A-9. Environment > Env tokens (page 1 of 3)

Prefix	Tokens	Description
ENV	CLIENT_BASE_PATH	Base (root) path of the client.
ENV	CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
ENV	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
ENV	CLIENT_NAME	DNS name or IP address of the client computer.
ENV	CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
ENV	CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
ENV	CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
ENV	CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.
ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
ENV	CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
ENV	CREATED_BY	ID of the user who created the environment.
ENV	CREATION_DATE	Date the environment was created.
ENV	DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
ENV	DATABASE_TYPE	Validation value code of the database type.
ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.

Table A-9. Environment > Env tokens (page 2 of 3)

Prefix	Tokens	Description
ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
ENV	DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
ENV	DB_NAME	DNS name or IP address of the database server.
ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
ENV	DB_PASSWORD	Encrypted password that PPM Center uses to log on to or access the database.
ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
ENV	DB_VERSION	Database version (such as 8.1.7).
ENV	DESCRIPTION	Environment description.
ENV	ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
ENV	ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
ENV	ENVIRONMENT_NAME	Environment name.
ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.
ENV	LAST_UPDATE_DATE	Date the environment was last updated.
ENV	LOCATION	Environment location.
ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
ENV	SERVER_BASE_PATH	Base (root) path of the server.

Table A-9. Environment > Env tokens (page 3 of 3)

Prefix	Tokens	Description
ENV	SERVER_CON_PROTOCOL	Protocol used to connect to this server.
ENV	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
ENV	SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
ENV	SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
ENV	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
ENV	SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
ENV	SERVER_NAME	DNS name or IP address of the server computer.
ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
ENV	SERVER_PASSWORD	Encrypted password that PPM Center uses to log on to or access the server.
ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Environment > Env > App Tokens

Table A-10. Environment > Env > App tokens (page 1 of 3)

Prefix	Tokens	Description
ENV.APP	APP_CODE	Short name (code) for the application.
ENV.APP	APP_NAME	Descriptive name for the application.
ENV.APP	CLIENT_BASE_PATH	Application-specific base (root) path of the client.

Table A-10. Environment > Env > App tokens (page 2 of 3)

Prefix	Tokens	Description
ENV.APP	CLIENT_PASSWORD	Encrypted application-specific password PPM Center uses to log on to or access the client. This value is encrypted.
ENV.APP	CLIENT_USERNAME	Application-specific username PPM Center uses to log on to or access the client.
ENV.APP	CLIENT_CON_PROTOCOL	Application-specific protocol used to connect to this client.
ENV.APP	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connection protocol.
ENV.APP	CLIENT_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this client.
ENV.APP	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
ENV.APP	CREATED_BY	ID of the user who created the application.
ENV.APP	CREATION_DATE	Date the application was created.
ENV.APP	DB_LINK	For Oracle database type, the application-specific database link from the PPM Center schema to the database schema for the environment.
ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.
ENV.APP	DB_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the database.
ENV.APP	DB_USERNAME	Application-specific username or schema name that PPM Center uses to log on to or access the database.
ENV.APP	DESCRIPTION	Application description.
ENV.APP	ENABLED_FLAG	Flag that indicates whether the application is enabled and available for selection in package lines.

Table A-10. Environment > Env > App tokens (page 3 of 3)

Prefix	Tokens	Description
ENV.APP	ENVIRONMENT_APP_ID	ID of the application in the table KENV_ENVIRONMENT_APPS.
ENV.APP	ENVIRONMENT_ID	ID of the environment with which the application is associated.
ENV.APP	ENVIRONMENT_NAME	Name of the environment with which the application is associated.
ENV.APP	LAST_UPDATED_BY	ID of the user who last updated the application.
ENV.APP	LAST_UPDATE_DATE	Date the application was last updated.
ENV.APP	SERVER_CON_PROTOCOL	Application-specific protocol used to connect to this server.
ENV.APP	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
ENV.APP	SERVER_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this server.
ENV.APP	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
ENV.APP	SERVER_BASE_PATH	Application-specific base (root) path of the server.
ENV.APP	SERVER_PASSWORD	Application-specific password PPM Center uses to log on to or access the server. This value is encrypted.
ENV.APP	SERVER_USERNAME	Application-specific username that PPM Center uses to log on to or access the server.
ENV.APP	WORKBENCH_ENVIRONMENT_URL	URL of the environment window in the PPM Workbench.

Environment > Env > Env Tokens

Table A-11. Environment > Env > Env tokens (page 1 of 3)

Prefix	Tokens	Description
ENV.ENV	CLIENT_BASE_PATH	Base (root) path of the client.
ENV.ENV	CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
ENV.ENV	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
ENV.ENV	CLIENT_NAME	DNS name or IP address of the client computer.
ENV.ENV	CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
ENV.ENV	CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
ENV.ENV	CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
ENV.ENV	CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
ENV.ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.
ENV.ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
ENV.ENV	CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
ENV.ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
ENV.ENV	CREATED_BY	ID of the user who created the environment.
ENV.ENV	CREATION_DATE	Date the environment was created.
ENV.ENV	DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
ENV.ENV	DATABASE_TYPE	Validation value code of the database type.

Table A-11. Environment > Env > Env tokens (page 2 of 3)

Prefix	Tokens	Description
ENV.ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
ENV.ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
ENV.ENV	DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
ENV.ENV	DB_NAME	DNS name or IP address of the database server.
ENV.ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
ENV.ENV	DB_PASSWORD	Password PPM Center uses to log on to or access the database. This value is encrypted.
ENV.ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
ENV.ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
ENV.ENV	DB_VERSION	Database version (such as 8.1.7).
ENV.ENV	DESCRIPTION	Environment description.
ENENV.ENV	ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
ENV.ENV	ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
ENV.ENV	ENVIRONMENT_NAME	Environment name.
ENV.ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.
ENV.ENV	LAST_UPDATE_DATE	Date the environment was last updated.

Table A-11. Environment > Env > Env tokens (page 3 of 3)

Prefix	Tokens	Description
ENV.ENV	LOCATION	Environment location.
ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, database name used to access the database from the command line.
ENV.ENV	SERVER_BASE_PATH	Base (root) path of the server.
ENV.ENV	SERVER_CON_PROTOCOL	Protocol used to connect to this server.
ENV.ENV	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
ENV.ENV	SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
ENV.ENV	SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
ENV.ENV	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
ENV.ENV	SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
ENV.ENV	SERVER_NAME	DNS name or IP address of the server computer.
ENV.ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
ENV.ENV	SERVER_PASSWORD	Password PPM Center uses to log on to or access the server. This value is encrypted.
ENV.ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
ENV.ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Environment > Source Env Tokens

Table A-12. Environment > Source Env tokens (page 1 of 3)

Prefix	Tokens	Description
SOURCE_ ENV	CLIENT_BASE_PATH	Base (root) path of the client.
SOURCE_ ENV	CLIENT_CON_ PROTOCOL	Protocol used to connect to this client.
SOURCE_ ENV	CLIENT_CON_ PROTOCOL_MEANING	Visible value of the client connect protocol.
SOURCE_ ENV	CLIENT_NAME	DNS name or IP address of the client computer.
SOURCE_ ENV	CLIENT_NT_DOMAIN	Domain name for a client running Windows.
SOURCE_ ENV	CLIENT_ENABLED_ FLAG	Flag that indicates whether the client portion of the environment is enabled.
SOURCE_ ENV	CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
SOURCE_ ENV	CLIENT_SQL_ COMMAND	Default command line SQL*Plus command name.
SOURCE_ ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.
SOURCE_ ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
SOURCE_ ENV	CLIENT_TRANSFER_ PROTOCOL	Protocol used to transfer files to or from this client.
SOURCE_ ENV	CLIENT_TRANSFER_ PROTOCOL_MEANING	Visible value of the client transfer protocol.
SOURCE_ ENV	CREATED_BY	ID of the user who created the environment.
SOURCE_ ENV	CREATION_DATE	Date the environment was created.
SOURCE_ ENV	DATABASE_ENABLED_ FLAG	Flag that indicates whether the database portion of the environment is enabled.
SOURCE_ ENV	DATABASE_TYPE	Validation value code of the database type.

Table A-12. Environment > Source Env tokens (page 2 of 3)

Prefix	Tokens	Description
SOURCE_ ENV	DB_CONNECT_STRING	For Oracle database type, connect string used to access the database from the command line.
SOURCE_ ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
SOURCE_ ENV	DB_LINK	For Oracle database type, database link from the PPM Center schema to the environment's database schema.
SOURCE_ ENV	DB_NAME	DNS name or IP address of the database server.
SOURCE_ ENV	DB_ORACLE_SID	For Oracle database type, SID of the database (often the same as the DB_CONNECT_STRING).
SOURCE_ ENV	DB_PASSWORD	Password PPM Center uses to log on to or access the database. This value is encrypted.
SOURCE_ ENV	DB_PORT_NUMBER	For Oracle database type, port number on which SQL*Net is listening for remote SQL connections on the database server.
SOURCE_ ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
SOURCE_ ENV	DB_VERSION	Database version (such as 8.1.7).
SOURCE_ ENV	DESCRIPTION	Environment description.
SOURCE_ ENV	ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
SOURCE_ ENV	ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
SOURCE_ ENV	ENVIRONMENT_NAME	Environment name.
SOURCE_ ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.

Table A-12. Environment > Source Env tokens (page 3 of 3)

Prefix	Tokens	Description
SOURCE_ ENV	LAST_UPDATE_DATE	Date the environment was last updated.
SOURCE_ ENV	LOCATION	Environment location.
SOURCE_ ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
SOURCE_ ENV	SERVER_BASE_PATH	Base (root) path of the server.
SOURCE_ ENV	SERVER_CON_ PROTOCOL	Protocol used to connect to this server.
SOURCE_ ENV	SERVER_CON_ PROTOCOL_MEANING	Visible value of the server connection protocol.
SOURCE_ ENV	SERVER_SQL_ COMMAND	Default command line SQL*Plus command name.
SOURCE_ ENV	SERVER_TRANSFER_ PROTOCOL	Protocol used to transfer files to or from this server.
SOURCE_ ENV	SERVER_TRANSFER_ PROTOCOL_MEANING	Visible value of the server transfer protocol.
SOURCE_ ENV	SERVER_ENABLED_ FLAG	Flag that indicates whether the server portion of the environment is enabled.
SOURCE_ ENV	SERVER_NAME	DNS name or IP address of the server computer.
SOURCE_ ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
SOURCE_ ENV	SERVER_PASSWORD	Encrypted password PPM Center uses to log on to or access the server.
SOURCE_ ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
SOURCE_ ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
SOURCE_ ENV	WORKBENCH_ ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Environment > Source Env > App Tokens

Table A-13. Environment > Source Env > App tokens (page 1 of 3)

Prefix	Token	Description
SOURCE_ ENV.APP	APP_CODE	Short name (code) for the application.
SOURCE_ ENV.APP	APP_NAME	Descriptive name for the application.
SOURCE_ ENV.APP	CLIENT_BASE_PATH	Application-specific base (root) path of the client.
SOURCE_ ENV.APP	CLIENT_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the client.
SOURCE_ ENV.APP	CLIENT_USERNAME	Application-specific username PPM Center uses to log on to or access the client.
SOURCE_ ENV.APP	CLIENT_CON_ PROTOCOL	Application-specific protocol used to connect to this client.
SOURCE_ ENV.APP	CLIENT_CON_ PROTOCOL_MEANING	Visible value of the client connection protocol.
SOURCE_ ENV.APP	CLIENT_TRANSFER_ PROTOCOL	Application-specific protocol used to transfer files to and from this client.
SOURCE_ ENV.APP	CLIENT_TRANSFER_ PROTOCOL_MEANING	Visible value of the client transfer protocol.
SOURCE_ ENV.APP	CREATED_BY	ID of the user who created the application.
SOURCE_ ENV.APP	CREATION_DATE	Date the application was created.
SOURCE_ ENV.APP	DB_LINK	For Oracle database type, application-specific database link from the PPM Center schema to the database schema for the environment.
SOURCE_ ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.

Table A-13. Environment > Source Env > App tokens (page 2 of 3)

Prefix	Token	Description
SOURCE_ ENV.APP	DB_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the database.
SOURCE_ ENV.APP	DB_USERNAME	Application-specific username or schema name that PPM Center uses to log on to or access the database.
SOURCE_ ENV.APP	DESCRIPTION	Application description.
SOURCE_ ENV.APP	ENABLED_FLAG	Flag that indicates whether the application is enabled and available for selection in package lines.
SOURCE_ ENV.APP	ENVIRONMENT_APP_ID	ID of the application in the table KENV_ENVIRONMENT_APPS.
SOURCE_ ENV.APP	ENVIRONMENT_ID	ID of the environment with which the application is associated.
SOURCE_ ENV.APP	ENVIRONMENT_NAME	Name of the environment with which the application is associated.
SOURCE_ ENV.APP	LAST_UPDATED_BY	ID of the user who last updated the application.
SOURCE_ ENV.APP	LAST_UPDATE_DATE	Date the application was last updated.
SOURCE_ ENV.APP	SERVER_CON_ PROTOCOL	Application-specific protocol used to connect to this server.
SOURCE_ ENV.APP	SERVER_CON_ PROTOCOL_MEANING	Visible value of the server connection protocol.
SOURCE_ ENV.APP	SERVER_TRANSFER_ PROTOCOL	Application-specific protocol used to transfer files to and from this server.
SOURCE_ ENV.APP	SERVER_TRANSFER_ PROTOCOL_MEANING	Visible value of the server transfer protocol.
SOURCE_ ENV.APP	SERVER_BASE_PATH	Application-specific base (root) path of the server.

Table A-13. Environment > Source Env > App tokens (page 3 of 3)

Prefix	Token	Description
SOURCE_ENV.APP	SERVER_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the server.
SOURCE_ENV.APP	SERVER_USERNAME	Application-specific username PPM Center uses to log on to or access the server.
SOURCE_ENV.APP	WORKBENCH_ENVIRONMENT_URL	URL of the environment window in the PPM Workbench.

Environment > Source Env > Env Tokens

Table A-14. Environment Source Env > Env tokens (page 1 of 4)

Prefix	Tokens	Description
SOURCE_ENV.ENV	CLIENT_BASE_PATH	Base (root) path of the client.
SOURCE_ENV.ENV	CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
SOURCE_ENV.ENV	CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
SOURCE_ENV.ENV	CLIENT_NAME	DNS name or IP address of the client computer.
SOURCE_ENV.ENV	CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
SOURCE_ENV.ENV	CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
SOURCE_ENV.ENV	CLIENT_PASSWORD	Encrypted password PPM Center uses to log on to or access the client.
SOURCE_ENV.ENV	CLIENT_SQL_COMMAND	Default command-line SQL*Plus command name.
SOURCE_ENV.ENV	CLIENT_TYPE_CODE	Validation value code of the client machine type.

Table A-14. Environment Source Env > Env tokens (page 2 of 4)

Prefix	Tokens	Description
SOURCE_ENV.ENV	CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
SOURCE_ENV.ENV	CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
SOURCE_ENV.ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
SOURCE_ENV.ENV	CREATED_BY	ID of the user who created the environment.
SOURCE_ENV.ENV	CREATION_DATE	Date the environment was created.
SOURCE_ENV.ENV	DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
SOURCE_ENV.ENV	DATABASE_TYPE	Validation value code of the database type.
SOURCE_ENV.ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
SOURCE_ENV.ENV	DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
SOURCE_ENV.ENV	DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
SOURCE_ENV.ENV	DB_NAME	DNS name or IP address of the database server.
SOURCE_ENV.ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
SOURCE_ENV.ENV	DB_PASSWORD	Encrypted password PPM Center uses to log on to or access the database.

Table A-14. Environment Source Env > Env tokens (page 3 of 4)

Prefix	Tokens	Description
SOURCE_ENV.ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
SOURCE_ENV.ENV	DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
SOURCE_ENV.ENV	DB_VERSION	Database version (such as 8.1.7).
SOURCE_ENV.ENV	DESCRIPTION	Environment description.
SOURCE_ENV.ENV	ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
SOURCE_ENV.ENV	ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
SOURCE_ENV.ENV	ENVIRONMENT_NAME	Environment name.
SOURCE_ENV.ENV	LAST_UPDATED_BY	ID of the user who last updated the environment.
SOURCE_ENV.ENV	LAST_UPDATE_DATE	Date the environment was last updated.
SOURCE_ENV.ENV	LOCATION	Environment location.
SOURCE_ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
SOURCE_ENV.ENV	SERVER_BASE_PATH	Base (root) path of the server.
SOURCE_ENV.ENV	SERVER_CON_PROTOCOL	Protocol used to connect to this server.
SOURCE_ENV.ENV	SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
SOURCE_ENV.ENV	SERVER_SQL_COMMAND	Default command line SQL*Plus command name.

Table A-14. Environment Source Env > Env tokens (page 4 of 4)

Prefix	Tokens	Description
SOURCE_ENV.ENV	SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
SOURCE_ENV.ENV	SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SOURCE_ENV.ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
SOURCE_ENV.ENV	SERVER_NAME	DNS name or IP address of the server computer.
SOURCE_ENV.ENV	SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
SOURCE_ENV.ENV	SERVER_PASSWORD	Password that PPM Center uses to log on to or access the server. This value is encrypted.
SOURCE_ENV.ENV	SERVER_TYPE_CODE	Validation value code of the server machine type.
SOURCE_ENV.ENV	SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
SOURCE_ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

Command Tokens

Table A-15. Command tokens

Prefix	Tokens	Description
EXEC	EXIT_CODE	Exit code of a command execution.
EXEC	OUTPUT	Last line of output from a command execution.

You can use the command execution tokens, `[EXEC.OUTPUT]` and `[EXEC.EXIT_CODE]` in the following contexts:

- Inside command step segments that use the `ksc_connect` and `ksc_exit` special commands.
- Immediately after command step segments that use the `ksc_local_exec` special command.

For example, the following code segment demonstrates how to use both of these command execution tokens to retrieve the output and exit code immediately upon execution. The tokens are used immediately after the `ksc_local_exec` special command.

```
ksc_local_exec pwd
ksc_set MY_PATH="[EXEC.OUTPUT]"
ksc_set MY_EXIT_CODE="[EXEC.EXIT_CODE]"
ksc_local_exec echo '[MY_PATH]/bin'
ksc_local_exec echo '[MY_EXIT_CODE]'
```

Financial Benefit Tokens

Table A-16. Financial Benefit tokens

Prefix	Tokens	Description
FBEN	ACTIVE_FLAG	Active flag of the financial benefit.
FBEN	BENEFIT_ID	ID of the financial benefit.
FBEN	BENEFIT_IS_FOR_ENTITY_NAME	Entity name to which the financial benefit is linked.
FBEN	BENEFIT_IS_FOR_ID	ID of the asset, project, or proposal to which the financial benefit is linked.
FBEN	BENEFIT_IS_FOR_NAME	Name of the asset, project, or proposal to which the financial benefit is linked.
FBEN	BENEFIT_NAME	Name of the financial benefit.
FBEN	BENEFIT_URL	URL to view the financial benefit.
FBEN	CREATED_BY	Username of the user who created the financial benefit.
FBEN	CREATION_DATE	Date when the financial benefit was created.
FBEN	DESCRIPTION	Description of the financial benefit.
FBEN	END_PERIOD	End period of the financial benefit.
FBEN	INITIATION_REQ	Initiation request ID of the financial benefit.
FBEN	PERIOD_SIZE	Period size of the financial benefit.
FBEN	REGION	Region associated with the financial benefit.
FBEN	START_PERIOD	Start period of the financial benefit.
FBEN	STATUS_CODE	Status code of the financial benefit.
FBEN	STATUS_NAME	Status name of the financial benefit.

Notification Tokens

Table A-17. Notification tokens

Prefix	Tokens	Description
NOTIF	CC_USERS	List of users on the Cc: header of the notification.
NOTIF	CHANGED_FIELD	Field that changed to trigger a notification.
NOTIF	EXCEPTION_RULE	Exception rule that was met by the task exception that caused the notification to be sent.
NOTIF	EXCEPTION_RULE_NAME	Name of the task exception that caused the notification to be sent.
NOTIF	EXCEPTION_VIOLATION	Specific violation of the exception that caused the notification to be sent.
NOTIF	NEW_VALUE	New value of the changed field.
NOTIF	NOTIFICATION_DETAILS	Notification details for linked tokens.
NOTIF	OLD_VALUE	Previous value of the changed field.
NOTIF	TO_USERS	List of users on the To: header of the notification.

Organization Unit Tokens

Table A-18. Organization Unit tokens (page 1 of 2)

Prefix	Tokens	Description
ORG	BUDGET_ID	ID of the budget linked to this org unit.
ORG	BUDGET_NAME	Name of the budget linked to this org unit.
ORG	CATEGORY_CODE	Lookup code of the org unit category (lookup type = RSC - org unit Category)
ORG	CATEGORY_NAME	Category name of the org unit.
ORG	CREATED_BY	ID of the user who created the org unit.

Table A-18. Organization Unit tokens (page 2 of 2)

Prefix	Tokens	Description
ORG	CREATED_BY_USERNAME	Name of the user who created the org unit.
ORG	CREATION_DATE	Date on which the org unit was created.
ORG	DEPARTMENT_CODE	Lookup code of the org unit department (lookup type = DEPT)
ORG	DEPARTMENT_NAME	Department name of the org unit.
ORG	LOCATION_CODE	Lookup code of the org unit location (lookup type = RSC - Location)
ORG	LOCATION_NAME	Location name of the org unit.
ORG	MANAGER_ID	ID of the org unit manager.
ORG	MANAGER_USERNAME	Name of the org unit manager.
ORG	ORG_UNIT_ID	Org unit ID (defined in table KRSC_ORG_UNITS).
ORG	ORG_UNIT_NAME	Org unit name.
ORG	PARENT_ORG_UNIT_ID	Parent org unit ID.
ORG	PARENT_ORG_UNIT_NAME	Parent org unit name.
ORG	REGIONAL_CALENDAR	Name of the regional calendar for the org unit.
ORG	REGION	Region associated with the Org Unit.
ORG	TYPE_CODE	Lookup code of the org unit category (lookup type = RSC - org unit Category)
ORG	TYPE_NAME	Type name of the org unit.

Package Tokens

Table A-19. Package tokens (page 1 of 3)

Prefix	Tokens	Description
PKG	ASSIGNED_TO_EMAIL	Email address of the user to whom the package is assigned.
PKG	ASSIGNED_TO_GROUP_ID	ID of the security group to which the package is assigned.
PKG	ASSIGNED_TO_GROUP_NAME	Security group to which the package is assigned.
PKG	ASSIGNED_TO_USERNAME	Name of the user to whom the package is assigned.
PKG	ASSIGNED_TO_USER_ID	ID of the user to whom the package is assigned.
PKG	CREATED_BY	ID of the user who created the package.
PKG	CREATED_BY_EMAIL	Email address of the user who created the package.
PKG	CREATED_BY_USERNAME	PPM Center username of the user who created the package.
PKG	CREATION_DATE	Date the package was created.
PKG	DESCRIPTION	Package description.
PKG	ID	Package ID in the table KDLV_PACKAGES.
PKG	LAST_UPDATED_BY	ID of the user who last updated the package.
PKG	LAST_UPDATED_BY_EMAIL	Email address of the user who last updated the package.
PKG	LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the package.
PKG	LAST_UPDATE_DATE	Date the package was last updated.
PKG	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent note.
PKG	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
PKG	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.

Table A-19. Package tokens (page 2 of 3)

Prefix	Tokens	Description
PKG	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PKG	NOTES	All notes for the package.
PKG	NUMBER	Package name/number.
PKG	PACKAGE_GROUP_CODE	Package group code.
PKG	PACKAGE_GROUP_NAME	Package group name.
PKG	PARENT_REQUEST_ID	ID of the request that created this package (if applicable).
PKG	PRIORITY	Package priority.
PKG	PRIORITY_CODE	Validation value code for the package priority.
PKG	PRIORITY_NAME	Validation value meaning of the package priority.
PKG	PRIORITY_SEQ	Package priority sequence.
PKG	PROJECT_CODE	Validation value code of the work plan to which the package belongs.
PKG	PROJECT_NAME	Validation value meaning of the work plan to which the package belongs.
PKG	SUBMIT_DATE	Date on which the package was submitted.
PKG	REQUESTED_BY_EMAIL	Email address of the user who requested the package.
PKG	REQUESTED_BY_USERNAME	PPM Center username of the user who requested the package.
PKG	REQUESTED_BY_USER_ID	ID of the user who requested the package.
PKG	PACKAGE_ID	ID of the package in the table KDLV_PACKAGES.
PKG	PACKAGE_NO_LINK	Standard hyperlink to the package in HTML-formatted notifications.
PKG	PACKAGE_TYPE	Validation value meaning of the package type.

Table A-19. Package tokens (page 3 of 3)

Prefix	Tokens	Description
PKG	PACKAGE_TYPE_CODE	Validation value code for the package type.
PKG	PACKAGE_URL	URL of the package in the standard interface.
PKG	PERCENT_COMPLETE	Percent complete of the package.
PKG	RUN_GROUP	Package run group.
PKG	STATUS	Validation value meaning for the package status.
PKG	STATUS_CODE	Validation value code for the package status.
PKG	WORKBENCH_PACKAGE_NO_LINK	Package URL in the PPM Workbench.
PKG	WORKBENCH_PACKAGE_URL	Package screen URL in the PPM Workbench.
PKG	WORKFLOW_ID	ID of the workflow that the package uses.
PKG	WORKFLOW_NAME	Name of the workflow that the package uses.

Package > Package Line Tokens

Table A-20. Package > Package Line tokens (page 1 of 2)

Prefix	Tokens	Description
PKG. PKGL	APP_CODE	Application code for the package line.
PKG. PKGL	APP_NAME	Name of the application for the package line.
PKG. PKGL	ID	ID of the package line in the table KDLV_PACKAGE_LINES.
PKG. PKGL	OBJECT_CATEGORY_CODE	Validation value code of the object type category of the line.
PKG. PKGL	OBJECT_CATEGORY_NAME	Validation value meaning of the object type category of the line.
PKG. PKGL	OBJECT_NAME	Object name of the package line.

Table A-20. Package > Package Line tokens (page 2 of 2)

Prefix	Tokens	Description
PKG. PKGL	OBJECT_REVISION	Value of the object revision column (if any) as specified by the object type of the package line.
PKG. PKGL	OBJECT_TYPE	Object type of the package line.
PKG. PKGL	OBJECT_TYPE_ID	ID of the object type of the package line.
PKG. PKGL	PACKAGE_LINE_ID	ID of the package line.
PKG. PKGL	SEQ	Sequence of the package line (relative to other lines in the same package).
PKG. PKGL	WORKBENCH_OBJECT_TYPE_URL	URL to access the object type window for this object type in the PPM Workbench.

Package > Pending Reference Tokens

Table A-21. Package > Pending Reference tokens (page 1 of 2)

Prefix	Tokens	Description
PKG.PEND	ID	ID of the entity that is blocked by the package.
PKG.PEND	NAME	Name of the entity that is blocked by the package.
PKG.PEND	DETAIL	Detail information for the entity that is blocked by the package.
PKG.PEND	DESCRIPTION	Description of the entity that is blocked by the package.
PKG.PEND	STATUS_ID	ID of the state or code of the status of the entity blocked by the package.
PKG.PEND	STATUS_NAME	Name of the status (or state) of the entity blocked by the package.
PKG.PEND	STATE	Name of the state of the entity of the request blocked by the package.
PKG.PEND	ASSIGNED_TO_USERNAME	Name of the assigned user (or resource) of the entity blocked by the package.

Table A-21. Package > Pending Reference tokens (page 2 of 2)

Prefix	Tokens	Description
PKG.PEND	ASSIGNED_TO_USER_ID	Username of the assigned user (or resource) of the entity blocked by the package.
PKG.PEND	ASSIGNED_TO_GROUP_NAME	Name of the assigned group (or resource group) of the entity that is blocked by the package.
PKG.PEND	ASSIGNED_TO_GROUP_ID	ID of the assigned group (or resource group) of the entity that is blocked by the package.
PKG.PEND	RESOURCE_USERNAME	Name of the resource associated with the entity that is blocked by the package.
PKG.PEND	RESOURCE_ID	Username of the assigned user (or resource) associated with the entity that is blocked by the package.
PKG.PEND	RESOURCE_GROUP_NAME	Name of the assigned group (or resource group) associated with the entity blocked by the package.
PKG.PEND	RESOURCE_GROUP_ID	ID of the assigned group (or resource group) associated with the entity that is blocked by the package.
PKG.PEND	PERCENT_COMPLETE	Current percent complete value associated with the entity that is blocked by the package.
PKG.PEND	ENTITY_TYPE_ID	ID of the type of entity that is blocked by the package.
PKG.PEND	ENTITY_TYPE_NAME	Name of the type of entity blocked by the package.

Package Line Tokens

Table A-22. Package Line tokens

Prefix	Tokens	Description
PKGL	APP_CODE	Application code for the package line.
PKGL	APP_NAME	Name of the application for the package line.
PKGL	ID	ID of the package line in the table KDLV_PACKAGE_LINES.
PKGL	OBJECT_CATEGORY_CODE	Validation value code of the object type category of the line.
PKGL	OBJECT_CATEGORY_NAME	Validation value meaning of the object type category of the line.
PKGL	OBJECT_NAME	Object name of the package line.
PKG	OBJECT_REVISION	Value of the object revision column (if any) as specified by the object type of the package line.
PKGL	OBJECT_TYPE	Object type of the package line.
PKGL	OBJECT_TYPE_ID	ID of the object type of the package line.
PKGL	PACKAGE_LINE_ID	ID of the package line.
PKGL	SEQ	Sequence of the package line (relative to other lines in the same package).
PKGL	WORKBENCH_OBJECT_TYPE_URL	URL to access the object type window for this object type in the PPM Workbench.

Program Tokens

Table A-23. Program tokens (page 1 of 2)

Prefix	Tokens	Description
PRG	CREATED_BY	ID of the user who created the program.
PRG	CREATED_BY_USERNAME	Name of the user who created the program.
PRG	LAST_UPDATED_BY	ID of the user who last updated the program.

Table A-23. Program tokens (page 2 of 2)

Prefix	Tokens	Description
PRG	LAST_UPDATED_BY_USERNAME	Name of the user who last updated the program.
PRG	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
PRG	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
PRG	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
PRG	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PRG	PROGRAM_MANAGER	ID(s) of the user(s) assigned to manage the program.

Project Tokens

Table A-24. Project tokens (page 1 of 5)

Prefix	Tokens	Description
PRJ	ACTUAL_DURATION	Actual duration of the work plan.
PRJ	ACTUAL_EFFORT	Actual effort associated with the work plan.
PRJ	ACTUAL_FINISH_DATE	Actual finish date of the work plan.
PRJ	ACTUAL_START_DATE	Actual start date of the work plan.
PRJ	BUDGET_ID	ID of the budget linked to the work plan.
PRJ	BUDGET_NAME	Name of the budget linked to the work plan.
PRJ	CONFIDENCE_CODE	Code of the confidence value entered by the user.
PRJ	CONFIDENCE_NAME	Name of the confidence value entered by the user.
PRJ	CREATED_BY	User who created the work plan.
PRJ	CREATED_BY_EMAIL	Email address of the user who created the work plan.

Table A-24. Project tokens (page 2 of 5)

Prefix	Tokens	Description
PRJ	CREATED_BY_USERNAME	Username of the person who created the work plan.
PRJ	CREATION_DATE	Creation date of the work plan.
PRJ	DEPARTMENT_CODE	Code of the department value entered by the user.
PRJ	DEPARTMENT_NAME	Name of the department value entered by the user.
PRJ	DESCRIPTION	Description of the work plan.
PRJ	ESTIMATED_REMAINING_DURATION	Estimated time remaining for the work plan.
PRJ	ESTIMATED_REMAINING_EFFORT	Estimated remaining effort involved in the work plan.
PRJ	ESTIMATED_FINISH_DATE	Estimated finish date of the work plan.
PRJ	LAST_UPDATE_DATE	Date on which the work plan was last updated.
PRJ	LAST_UPDATED_BY	Last person to update the work plan.
PRJ	LAST_UPDATED_BY_EMAIL	Email address of the last person to update the project plan.
PRJ	LAST_UPDATED_BY_USERNAME	Username of the last person to update the work plan.
PRJ	MASTER_PROJECT_ID	ID of the master project.
PRJ	MASTER_PROJECT_NAME	Name of the master project.
PRJ	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent note.
PRJ	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
PRJ	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
PRJ	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PRJ	MOST_RECENT_NOTE_TYPE	Type of the most recent note (USER or FIELD CHANGE).

Table A-24. Project tokens (page 3 of 5)

Prefix	Tokens	Description
PRJ	MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_AUTHOR_USERNAME	Username of the author of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_AUTHORED_DATE	Date of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_TEXT	Text of the most recent user note.
PRJ	PARENT_PROJECT_ID	ID of the parent work plan.
PRJ	PARENT_PROJECT_NAME	Name of the parent work plan.
PRJ	PERCENT_COMPLETE	Percent of the work plan completed.
PRJ	PRIORITY	Priority of the work plan.
PRJ	PROGRAM_ID	Delimited list of program ids of all programs associated with the project.
PRJ	PROGRAM_ID_MANAGER	Delimited list of manager ids of all programs associated with the project.
PRJ	PROGRAM_ID_MANAGER_USERNAME	Delimited list of manager usernames of all programs associated with the project.
PRJ	PROGRAM_NAME	Delimited list of program names of all programs associated with the project.
PRJ	PROJECT_ID	Number that uniquely identifies the work plan (same as PROJECT_NUMBER) in the table KDRV_PROJECTS.
PRJ	PROJECT_MANAGER	Manager of the work plan. Use this token to get the project manager information from the project container.
PRJ	PROJECT_MANAGER_EMAIL	Email address of the project manager.
PRJ	PROJECT_MANAGER_USERNAME	Username of the project manager.
PRJ	PROJECT_NAME	Work plan name.

Table A-24. Project tokens (page 4 of 5)

Prefix	Tokens	Description
PRJ	PROJECT_NAME_LINK	Standard hyperlink to the work plan in HTML-formatted notifications.
PRJ	PROJECT_NUMBER	Number that uniquely identifies the work plan (same as PROJECT_ID).
PRJ	PROJECT_PATH	Work plan path. This is a hierarchy of parent work plans that contain this work plan.
PRJ	PROJECT_REQUEST_ID	Request ID for this project. Returns the request_id associated with the project. You can feed this into a request token to get at all the request details.
PRJ	PROJECT_STAKEHOLDER	Delimited list of user ids of the project stakeholders.
PRJ	PROJECT_STAKEHOLDER_EMAIL	Delimited list of emails of the project stakeholders.
PRJ	PROJECT_STAKEHOLDER_USERNAME	Delimited list of usernames of the project stakeholders.
PRJ	PROJECT_STATE	Work plan state.
PRJ	PROJECT_TEMPLATE	Name of the project template used to create the project plan.
PRJ	PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
PRJ	PROJECT_URL	URL for the Project Overview page of the work plan.
PRJ	REGIONAL_CALENDAR	Name of the regional calendar for the work plan
PRJ	SCHEDULED_EFFORT	Scheduled effort defined in the work plan.
PRJ	SCHEDULED_DURATION	Scheduled duration for the work plan.
PRJ	SCHEDULED_FINISH_DATE	Finish date scheduled for the work plan.

Table A-24. Project tokens (page 5 of 5)

Prefix	Tokens	Description
PRJ	SCHEDULED_START_DATE	Start date scheduled for the work plan.
PRJ	SUMMARY_CONDITION	Summary condition of the work plan.
PRJ	WORKBENCH_PROJECT_URL	URL used to access this work plan in the PPM Workbench.

Project Detail Tokens

Table A-25. Project Detail tokens

Prefix	Tokens	Description
PRJD	PROJECT_DETAIL_ID	Project detail ID of the work plan in the table KDRV_PROJECTS.
PRJD	PROJECT_ID	Project ID of the work plan in the table KDRV_PROJECTS.

Parameters are accessible with this prefix (similar to request detail):

[PRJD.P.CUSTOM_TOKEN] .

Release Tokens

Table A-26. Release tokens (page 1 of 2)

Prefix	Tokens	Description
REL	RELEASE_ID	ID of the release in the KREL_RELEASES table.
REL	RELEASE_NAME	Release name.
REL	RELEASE_STATUS	Release status.
REL	CREATED_BY	ID of the user who created the release.
REL	CREATED_BY_USERNAME	PPM Center username of the user who created the release.
REL	LAST_UPDATED_BY	ID of the user who last updated the release.

Table A-26. Release tokens (page 2 of 2)

Prefix	Tokens	Description
REL	LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the release.
REL	LAST_UPDATE_DATE	Date on which the release was last updated.
REL	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
REL	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
REL	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
REL	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
REL	RELEASE_MANAGER	PPM Center user designated as the release manager.
REL	RELEASE_TEAM	Group of PPM Center users associated with the release.
REL	RELEASE_GROUP	High-level categorization of the release.
REL	DESCRIPTION	Release description.
REL	NOTES	Notes contained within the release.

Release > Distribution Tokens

Table A-27. Release > Distribution tokens (page 1 of 2)

Prefix	Tokens	Description
REL.DIST	CREATED_BY	User ID of the user who created the distribution.
REL.DIST	CREATE_BE_USERNAME	Username of the user who created the distribution.
REL.DIST	DESCRIPTION	Description of the release.
REL.DIST	DISTRIBUTION_ID	Internal identifier of the distribution for the release.
REL.DIST	DISTRIBUTION_NAME	Name of the distribution for the release.

Table A-27. Release > Distribution tokens (page 2 of 2)

Prefix	Tokens	Description
REL.DIST	DISTRIBUTION_STATUS	Status of the distribution for the release.
REL.DIST	FEEDBACK_FLAG	Feedback flag for the release.
REL.DIST	FEEDBACK_VALUE	Feedback value for the release.
REL.DIST	LAST_UPDATED_BY	User ID of the user who updated the distribution.
REL.DIST	LAST_UPDATED_BY_USERNAME	Username of the user who last updated the distribution.
REL.DIST	LAST_UPDATE_DATE	Last update date of the distribution.
REL.DIST	RELEASE_ID	Internal identifier of the release.
REL.DIST	RELEASE_NAME	Name of the release.
REL.DIST	WORKFLOW	Workflow assigned to the release.

Request Tokens

Table A-28. Request tokens (page 1 of 4)

Prefix	Tokens	Description
REQ	APPLICATION_CODE	Validation value code for the application to which the request is assigned.
REQ	APPLICATION_NAME	Validation value meaning of the application to which the request is assigned.
REQ	ASSIGNED_TO_EMAIL	Email address of the user to whom the request is assigned.
REQ	ASSIGNED_TO_GROUP_ID	ID of the security group to which the request is assigned.
REQ	ASSIGNED_TO_GROUP_NAME	Name of the security group to which the request is assigned.
REQ	ASSIGNED_TO_USERNAME	PPM Center username of the user to whom the request is assigned.
REQ	ASSIGNED_TO_NAME	Full name of the assigned user.

Table A-28. Request tokens (page 2 of 4)

Prefix	Tokens	Description
REQ	ASSIGNED_TO_USER_ID	ID of the user to whom the request is assigned.
REQ	COMPANY	Company employing the user who created the request.
REQ	COMPANY_NAME	Name of the company employing the user who created the request.
REQ	CONTACT_EMAIL	Email address of the contact for the request.
REQ	CONTACT_NAME	Full name of the contact for the request.
REQ	CONTACT_PHONE_NUMBER	Phone number of the contact for the request.
REQ	CREATED_BY	ID of the user who created the request.
REQ	CREATED_BY_EMAIL	Email address of the user who created the request.
REQ	CREATED_BY_NAME	Full name of the created by user.
REQ	CREATED_BY_USERNAME	PPM Center username of the user who last updated the request.
REQ	CREATION_DATE	Date on which the request was created.
REQ	DEPARTMENT_CODE	Validation value code of the department for the request.
REQ	DEPARTMENT_NAME	Validation value meaning of the department for the request.
REQ	DESCRIPTION	Request description.
REQ	LAST_UPDATED_BY	ID of the user who last updated the request.
REQ	LAST_UPDATED_BY_EMAIL	Email address of the user who last updated the request.
REQ	LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the request.
REQ	LAST_UPDATE_DATE	Date on which the request was last updated.
REQ	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.

Table A-28. Request tokens (page 3 of 4)

Prefix	Tokens	Description
REQ	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
REQ	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
REQ	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
REQ	MOST_RECENT_NOTE_TYPE	Type of the most recent note (USER or FIELD CHANGE).
REQ	MOST_RECENT_NOTE_CONTEXT	In the case of requests, this is the request status; blank in all other cases.
REQ	MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent user note.
REQ	MOST_RECENT_USER_NOTE_AUTHOR_USERNAME	Username of the author of the most recent user note.
REQ	MOST_RECENT_USER_NOTE_AUTHORED_DATE	Date of the most recent user note.
REQ	MOST_RECENT_USER_NOTE_TEXT	Text of the most recent user note.
REQ	MOST_RECENT_USER_NOTE_TYPE	Type of the most recent user note (USER or FIELD CHANGE).
REQ	MOST_RECENT_USER_NOTE_CONTEXT	Request status.
REQ	NOTES	All notes for the request.
REQ	PERCENT_COMPLETE	Percent of the request that is completed.
REQ	PRIORITY_CODE	Validation value code of the request priority.
REQ	PRIORITY_NAME	Validation value meaning of the request priority.
REQ	PROJECT_CODE	Validation value code of the work plan to which the request belongs.
REQ	PROJECT_NAME	Validation value meaning of the work plan to which the request belongs.

Table A-28. Request tokens (page 4 of 4)

Prefix	Tokens	Description
REQ	SUBMIT_DATE	Date on which the request was submitted.
REQ	REQUEST_GROUP_CODE	Request group code.
REQ	REQUEST_GROUP_NAME	Request group name.
REQ	REQUEST_ID	ID of the request in the table KCRT_REQUESTS.
REQ	REQUEST_ID_LINK	Standard hyperlink to display for the request in HTML-formatted notifications.
REQ	REQUEST_SUB_TYPE_ID	ID of the sub-type for the request.
REQ	REQUEST_SUB_TYPE_NAME	Name of the sub-type for the request.
REQ	REQUEST_TYPE_ID	ID of the request type of the request.
REQ	REQUEST_TYPE_NAME	Name of the request type.
REQ	REQUEST_URL	URL of the request in the standard interface.
REQ	STATUS_ID	ID of the request status.
REQ	STATUS_NAME	Request status.
REQ	WORKBENCH_REQUEST_TYPE_URL	URL of the request type in the PPM Workbench.
REQ	WORKBENCH_REQUEST_URL	URL of the request in the PPM Workbench.
REQ	WORKFLOW_ID	ID of the workflow that the request uses.
REQ	WORKFLOW_NAME	Name of the workflow that the request uses.

Request > Pending Reference Tokens

Table A-29. Request > Pending Reference tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.PEND	ID	ID of the entity that the request is blocking.
REQ.PEND	NAME	Name of the entity that the request is blocking.
REQ.PEND	DETAIL	Detail information for the entity that the request is blocking.
REQ.PEND	DESCRIPTION	Description of the entity that the request is blocking.
REQ.PEND	STATUS_ID	ID of the state or code of the status of the entity that the request is blocking.
REQ.PEND	STATUS_NAME	Name of the status (or state) of the entity that the request is blocking.
REQ.PEND	STATE	Name of the state of the entity of the request that is blocked by the request.
REQ.PEND	ASSIGNED_TO_USERNAME	Name of the assigned user (or resource) of the entity that the request is blocking.
REQ.PEND	ASSIGNED_TO_USER_ID	Username of the assigned user (or resource) of the entity that the request is blocking.
REQ.PEND	ASSIGNED_TO_GROUP_NAME	Name of the assigned group (or resource group) of the entity that the request is blocking.
REQ.PEND	ASSIGNED_TO_GROUP_ID	ID of the assigned group (or resource group) of the entity that the request is blocking.
REQ.PEND	RESOURCE_USERNAME	Name of the resource associated with the entity that the request is blocking.
REQ.PEND	RESOURCE_ID	Username of the assigned user (or resource) associated with the entity that the request is blocking.
REQ.PEND	RESOURCE_GROUP_NAME	Name of the assigned group (or resource group) associated with the entity that is blocked by the request.

Table A-29. Request > Pending Reference tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.PEND	RESOURCE_GROUP_ID	ID of the assigned group (or resource group) associated with the entity that is being blocked by the request.
REQ.PEND	PERCENT_COMPLETE	Current percent complete value associated with the entity that the request is blocking.
REQ.PEND	ENTITY_TYPE_ID	ID of the type of entity that the request is blocking.
REQ.PEND	ENTITY_TYPE_NAME	Name of the type of entity that the request is blocking.

Request > Field Tokens

The request field tokens are the tokens associated with field groups. Field groups are attached to request header types to enable additional pre-configured fields on requests. For more information concerning request field tokens, see [Request > Field Tokens on page 203](#).

Request Detail Tokens

Table A-30. Request Detail tokens

Prefix	Tokens	Description
REQD	CREATED_BY	ID of the user who created the request detail.
REQD	CREATION_DATE	Date on which the request detail was created.
REQD	LAST_UPDATED_BY	ID of the user who last updated the request detail.
REQD	LAST_UPDATE_DATE	Date on which request detail was last updated.
REQD	REQUEST_DETAIL_ID	ID for the request detail in the table KCRT_REQUEST_DETAILS.
REQD	REQUEST_ID	ID of the request for the request detail.
REQD	REQUEST_TYPE_ID	ID of the request type for the request detail.

The `REQD` prefix is typically used for accessing custom fields, such as:
`[REQD.P.CUSTOM_TOKEN]`.

Request Detail > Field Tokens

Within the token builder, Request Detail Field is an empty folder.

Resource Pool Tokens

Table A-31. Resource Pool tokens

Prefix	Tokens	Description
RSCP	CREATED_BY	The username of the user who created the resource pool.
RSCP	CREATION_DATE	The date on which the resource pool was created.
RSCP	DESCRIPTION	The resource pool description.
RSCP	END_PERIOD	The resource pool end period.
RSCP	PERIOD_SIZE	The resource pool period size.
RSCP	RESOURCE_POOL_URL	The URL used to view the resource pool.
RSCP	RSC_POOL_ID	The ID of the resource pool in table KRSC_RSC_POOLS.
RSCP	RSC_POOL_IS_FOR_ENTITY_NAME	The entity name to which the resource pool is linked (program or org unit).
RSCP	RSC_POOL_IS_FOR_ID	The ID of the program or org unit to which the resource pool is linked.
RSCP	RSC_POOL_IS_FOR_NAME	The name of the program or org unit to which the resource pool is linked.
RSCP	RSC_POOL_NAME	The resource pool name.

Security Group Tokens

Table A-32. Security Group tokens

Prefix	Tokens	Description
SG	CREATED_BY	ID of the user who created the security group.
SG	CREATION_DATE	The date on which the security group was created.
SG	DESCRIPTION	The security group description.
SG	LAST_UPDATED_BY	ID of the user who last updated the security group.
SG	LAST_UPDATE_DATE	Date on which the security group was last updated.
SG	SECURITY_GROUP_ID	ID of the security group in the table KNTA_SECURITY_GROUPS.
SG	SECURITY_GROUP_NAME	Security group name.

Skill Tokens

Table A-33. Skill tokens

Prefix	Tokens	Description
SKL	CREATED_BY	User ID of the user who created the skill.
SKL	CREATED_BY_USERNAME	Name of the user who created the skill.
SKL	CREATION_DATE	Date on which the skill was created.
SKL	SKILL_CATEGORY_CODE	Lookup code for the skill Category (lookup type = RSC - skill Category).
SKL	SKILL_CATEGORY_NAME	Name of the skill category.
SKL	SKILL_ID	ID of the skill in table KRSC_SKILLS.
SKL	SKILL_NAME	Skill name.

Staffing Profile Tokens

Table A-34. Staffing Profile tokens

Prefix	Tokens	Description
STFP	CREATED_BY	Username of the user who created the staffing profile.
STFP	CREATION_DATE	Date on which the staffing profile was created.
STFP	DESCRIPTION	Description of the staffing profile.
STFP	END_PERIOD	End period of the staffing profile.
STFP	STAFFING_PROFILE_URL	URL to view this staffing profile.
STFP	STAFF_PROF_ID	ID of the staffing profile in table KRSC_STAFF_PROFS.
STFP	STAFF_PROF_IS_FOR_ENTITY_NAME	Entity name to which the staffing profile is linked.
STFP	STAFF_PROF_IS_FOR_ID	ID of the work plan, program or org unit to which the staffing profile is linked.
STFP	STAFF_PROFL_IS_FOR_NAME	Name of the work plan, program or org unit to which the staffing profile is linked (work plan, program, or org unit).
STFP	STAFF_PROF_NAME	Staffing profile name.
STFP	START_PERIOD	Staffing profile start period.
STFP	STATUS_CODE	Staffing profile status code.
STFP	STATUS_NAME	Staffing profile status name.

Step TXN (Transaction) Tokens

Table A-35. Step TXN (Transaction) tokens (page 1 of 2)

Prefix	Tokens	Description
WST	CONCURRENT_REQUEST_ID	Identifier for the Oracle Concurrent Request for the Workflow Step Transaction.
WST	CREATED_BY	User ID of the user who created the Workflow Step Transaction.
WST	CREATION_DATE	Date the Workflow Step Transaction was created.
WST	ERROR_MESSAGE	Any system level error message associated with the Workflow Step Transaction.
WST	EXECUTION_BATCH_ID	Execution batch ID for the Workflow Step Transaction.
WST	HIDDEN_STATUS	Hidden status of the Workflow Step Transaction.
WST	LAST_UPDATED_BY	User ID of the user who last updated the Workflow Step Transaction.
WST	LAST_UPDATED_BY_EMAIL	Email address of the user who last updated the Workflow Step Transaction.
WST	LAST_UPDATED_BE_USERNAME	Username of the user who last updated the Workflow Step Transaction.
WST	LAST_UPDATE_DATE	Date the Workflow Step Transaction was last updated.
WST	STATUS	Status of the Workflow Step Transaction.
WST	STEP_TRANSACTION_ID	Transaction ID for the Workflow Step Transaction.
WST	TIMEOUT_DATE	Date of the last timeout on the Workflow Step Transaction.
WST	USER_COMMENT	Any comments a user added to the Workflow Step Transaction.
WST	WORKFLOW_ID	Workflow ID for the Workflow Step Transaction.
WST	WORKFLOW_STEP_ID	Workflow step ID for the Workflow Step Transaction.

Table A-35. Step TXN (Transaction) tokens (page 2 of 2)

Prefix	Tokens	Description
WST	NEW_HIDDEN_STATUS	New hidden status of the Workflow Step Transaction.
WST	OLD_HIDDEN_STATUS	Old hidden status of the Workflow Step Transaction.
WST	NEW_STATUS	New status of the Workflow Step Transaction.
WST	OLD_STATUS	Old status of the Workflow Step Transaction.

System Tokens

Table A-36. System tokens

Prefix	Tokens	Description
SYS	DATE	Date on which the token is parsed.
SYS	FULL_NAME	Full name of the PPM Center user.
SYS	ITG_TIME_STAMP	Date and time stamp when the token is parsed. You can use this token with the <code>ksc_store</code> command.
SYS	NEWLINE	New line character.
SYS	TIME_STAMP	Date and time stamp. (Deprecated)
SYS	UNIQUE_IDENTIFIER	Used to obtain a unique number from the database. It can be used to generate unique filenames, for example. It is often necessary to use with the <code>ksc_set</code> special command.
SYS	UNIX_NEWLINE	UNIX new line character.
SYS	USERNAME	PPM Center username for the user currently logged on to PPM Center.
SYS	USER_ID	ID of the user currently logged on to PPM Center.

Task Tokens

Table A-37. Tasks tokens (page 1 of 3)

Prefix	Tokens	Description
TSK	ACTUAL_DURATION	Actual task duration.
TSK	ACTUAL_EFFORT	Actual effort associated with the task.
TSK	ACTUAL_FINISH_DATE	Actual date the task finished.
TSK	ACTUAL_START_DATE	Actual date the task started.
TSK	CONFIDENCE_CODE	Confidence code that the user entered.
TSK	CONFIDENCE_NAME	Confidence name that the user entered.
TSK	CONSTRAINT_DATE	Task constraint date.
TSK	CREATED_BY	User who created the task.
TSK	CREATED_BY_EMAIL	Email address of the user who created the task.
TSK	CREATED_BY_USERNAME	Username of the user who created the task.
TSK	CREATION_DATE	Date the task was created.
TSK	DEPARTMENT_CODE	Department code value the user entered.
TSK	DEPARTMENT_NAME	Department name the user entered.
TSK	DESCRIPTION	TASK description.
TSK	ESTIMATED_REMAINING_DURATION	Estimated time remaining to complete the task.
TSK	ESTIMATED_REMAINING_EFFORT	Estimated remaining effort involved in the task.
TSK	ESTIMATED_FINISH_DATE	Estimated finish date of the task.
TSK	HAS_EXCEPTIONS	Flag to show whether or not the task has exceptions.
TSK	LAST_UPDATE_DATE	Date on which the task was last updated.
TSK	LAST_UPDATED_BY	Last user to update the task.
TSK	LAST_UPDATED_BY_EMAIL	Email address of the last user to update the task.
TSK	LAST_UPDATED_BY_USERNAME	Username of the last person to update the task.

Table A-37. Tasks tokens (page 2 of 3)

Prefix	Tokens	Description
TSK	MASTER_PROJECT_ID	ID of the master project.
TSK	MASTER_PROJECT_NAME	Name of the master project.
TSK	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
TSK	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
TSK	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
TSK	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
TSK	PARENT_PROJECT_ID	ID of the parent work plan.
TSK	PARENT_PROJECT_NAME	Name of the parent work plan.
TSK	PERCENT_COMPLETE	Percentage of the task completed.
TSK	PRIORITY	Task priority.
TSK	PROJECT_PATH	Work plan path. Hierarchy of parent work plans that contain this task.
TSK	PROJECT_TEMPLATE	Name of the project template used to create the work plan that contains the task.
TSK	PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
TSK	RESOURCE_ID	ID of the resource assigned to the task.
TSK	RESOURCE_EMAIL	Email address of the resource.
TSK	RESOURCE_GROUP_ID	ID of the resource group assigned to the task.
TSK	RESOURCE_GROUP_NAME	Name of the resource group assigned to the task.
TSK	RESOURCE_USERNAME	Username of the resource.
TSK	SCHEDULED_EFFORT	Scheduled effort involved in the task.
TSK	SCHEDULED_DURATION	Duration scheduled for the task.

Table A-37. Tasks tokens (page 3 of 3)

Prefix	Tokens	Description
TSK	SCHEDULED_FINISH_DATE	Finish date scheduled for the task.
TSK	SCHEDULED_START_DATE	Start date scheduled for the task.
TSK	SCHEDULING CONSTRAINT	Scheduling constraint for the task.
TSK	TASK_ID	The number that uniquely identifies the task (same as TASK_NUMBER). This corresponds to the PROJECT_ID column in table KDRV_PROJECTS.
TSK	TASK_NAME	Task name.
TSK	TASK_NAME_LINK	Standard hyperlink to the task in HTML-formatted notifications.
TSK	TASK_NUMBER	Number that uniquely identifies the task (same as TASK_ID).
TSK	TASK_STATE	The task state.
TSK	TASK_URL	URL for the task Detail page.
TSK	WORKBENCH_TASK_URL	URL to access this task in the PPM Workbench.

Tasks > Pending Tokens

Table A-38. Tasks > Pending tokens (page 1 of 2)

Prefix	Tokens	Description
TSK.PEND	ID	ID of the entity that is blocked by the task.
TSK.PEND	NAME	Name of the entity that is blocked by the task.
TSK.PEND	DETAIL	Detail information for the entity that is blocked by the task as shown in the References field.
TSK.PEND	DESCRIPTION	Description of the entity that is blocked by the task.
TSK.PEND	STATUS_ID	ID of the state or the code of the status of the entity that is being blocked by the task.
TSK.PEND	STATUS_NAME	Name of the status (or state) of the entity that is blocked by the task.

Table A-38. Tasks > Pending tokens (page 2 of 2)

Prefix	Tokens	Description
TSK.PEND	STATE	Name of the state of the entity blocked by the task.
TSK.PEND	ASSIGNED_TO_USERNAME	Name of the assigned user (or resource) of the entity blocked by the task.
TSK.PEND	ASSIGNED_TO_USER_ID	Username of the assigned user (or resource) of the entity blocked by the task.
TSK.PEND	ASSIGNED_TO_GROUP_NAME	Name of the assigned group (or resource group) of the entity blocked by the task.
TSK.PEND	ASSIGNED_TO_GROUP_ID	ID of the assigned group (or resource group) of the entity that is being blocked by the task.
TSK.PEND	RESOURCE_USERNAME	Name of the resource associated with the entity that is blocked by the task.
TSK.PEND	RESOURCE_ID	Username of the resource (or assigned user) associated with the entity blocked by the task.
TSK.PEND	RESOURCE_GROUP_NAME	Name of the resource group (or assigned user) associated with the entity blocked by the task.
TSK.PEND	RESOURCE_GROUP_ID	ID of the resource group (or assigned group) associated with the entity blocked by the task.
TSK.PEND	PERCENT_COMPLETE	Current percent complete value associated with the entity blocked by the task.
TSK.PEND	ENTITY_TYPE_ID	ID of the type of entity blocked by the task.
TSK.PEND	ENTITY_TYPE_NAME	Name of the type of entity that is being blocked by the task.

Time Management Notification Tokens

Table A-39. Time Management Notification tokens

Prefix	Tokens	Description
TMG	CREATE_TIME_SHEET_URL	URL for creating a new time sheet time period.
TMG	OPEN_TIME_SHEET_URL	URL for opening time sheet.
TMG	TIME_PERIOD	Time period.
TMG	TIME_SHEET_DESCRIPTION	Time sheet description.

User Tokens

Table A-40. User tokens (page 1 of 3)

Prefix	Tokens	Description
USR	AUTHENTICATION_MODE_CODE	Authentication mode for the user (such as LDAP).
USR	AUTHENTICATION_MODE_NAME	Authentication mode for the user (such as LDAP).
USR	COMPANY	Company that employs the user.
USR	COMPANY_NAME	Name of the company that employs the user.
USR	CREATED_BY	ID of the user who created the user.
USR	CREATED_BY_FULL_NAME	Full name of the “created by” user.
USR	CREATED_BY_USERNAME	PPM Center username of the user who created the user.
USR	CREATION_DATE	Date on which the user was created.
USR	DEPARTMENT_CODE	Lookup code of the department the user belongs to (lookup type = DEPT).
USR	DEPARTMENT_NAME	Name of the department to which the user belongs.
USR	EMAIL_ADDRESS	Email address of the user.

Table A-40. User tokens (page 2 of 3)

Prefix	Tokens	Description
USR	END_DATE	Date on which the user is made inactive in the application.
USR	FIRST_NAME	First name of the user.
USR	LAST_NAME	Last name of the user.
USR	LAST_UPDATED_BY	ID of the user who last updated the user.
USR	LAST_UPDATE_BY_FULL_NAME	Full name of the last updated by user.
USR	LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the user.
USR	LAST_UPDATE_DATE	Date the user was last updated.
USR	LOCATION_CODE	Lookup code of the user's location (lookup type = RSC - Location).
USR	LOCATION_NAME	Name of the user's location.
USR	MANAGER_USERNAME	Username of the user's manager.
USR	MANAGER_USER_ID	ID of the user's manager.
USR	PASSWORD	Password for the user to use to log on to PPM Center. This value is encrypted.
USR	PASSWORD_EXPIRATION_DATE	Date the password needs to be reset for the user.
USR	PASSWORD_EXPIRATION_DAYS	Number of days until the password must be reset for the user.
USR	PHONE_NUMBER	Phone number of the user.
USR	PRIMARY_ROLE_ID	ID of the primary role associated with the user.
USR	PRIMARY_ROLE_NAME	Name of the primary role associated with the user.
USR	REGION	Region associated with the user.
USR	REGIONAL_CALENDAR	Name of the regional calendar for the user.
USR	RESOURCE_CATEGORY_CODE	Lookup code of resource category (lookup type = RSC - Category) to which the user belongs.

Table A-40. User tokens (page 3 of 3)

Prefix	Tokens	Description
USR	RESOURCE_CATEGORY_NAME	Name of the category to which the user belongs.
USR	RESOURCE_TITLE_CODE	Lookup code of the user's resource title (lookup type = RSC - Resource Title).
USR	RESOURCE_TITLE_NAME	Name of the user's resource title.
USR	START_DATE	Date the user is made active in the application.
USR	USERNAME	Username for the user to use to log on to PPM Center.
USR	USER_ID	ID of the user in the table KNTA_USERS.
USR	WORKLOAD_CAPACITY	Workload capacity of the user (% of FTE)

Validation Tokens

Table A-41. Validation tokens (page 1 of 2)

Prefix	Tokens	Description
VAL	COMPONENT_TYPE	Component type associated with the validation.
VAL	CREATED_BY	ID of the user who created the validation.
VAL	CREATION_DATE	Date the validation was created.
VAL	DESCRIPTION	Description of the validation.
VAL	LAST_UPDATED_BY	ID of the user who last updated the validation.
VAL	LAST_UPDATE_DATE	Date the validation was last updated.
VAL	LOOKUP_TYPE	Lookup type associated with the validation (if applicable).
VAL	VALIDATION_ID	ID of the validation in the table KNTA_VALIDATIONS.

Table A-41. Validation tokens (page 2 of 2)

Prefix	Tokens	Description
VAL	VALIDATION_NAME	Name of the validation.
VAL	VALIDATION_SQL	SQL statement associated with the validation (if applicable).
VAL	WORKBENCH_VALIDATION_URL	URL for the validation in the PPM Workbench.

Validation > Value Tokens

Table A-42. Validation > Value tokens

Prefix	Tokens	Description
VAL.VALUE	CREATED_BY	ID of the user who created the value.
VAL.VALUE	CREATION_DATE	Date the value was created.
VAL.VALUE	DEFAULT_FLAG	Flag to indicate whether the value is the default value for the associated lookup type.
VAL.VALUE	DESCRIPTION	Description of the value.
VAL.VALUE	ENABLED_FLAG	Flag that indicates whether the value is available for selection in a list.
VAL.VALUE	LAST_UPDATED_BY	ID of the user who last updated the value.
VAL.VALUE	LAST_UPDATE_DATE	Date the value was last updated.
VAL.VALUE	LOOKUP_CODE	Code associated with the value.
VAL.VALUE	LOOKUP_TYPE	Value lookup type.
VAL.VALUE	MEANING	Meaning associated with the value.
VAL.VALUE	SEQ	Sequence relative to other values in the associated lookup type in which this value is to be displayed as a list item.

Workflow Tokens

Table A-43. Workflow tokens

Prefix	Tokens	Description
WF	CREATED_BY	ID of the user who created the workflow.
WF	CREATION_DATE	Date on which the workflow was created.
WF	DESCRIPTION	Workflow description.
WF	ENABLED_FLAG	Flag that indicates whether the workflow is enabled for use in packages and/or requests.
WF	FIRST_WORKFLOW_STEP_ID	ID of the first workflow step in the workflow.
WF	FIRST_WORKFLOW_STEP_NAME	Name of the first workflow step in the workflow.
WF	ICON_NAME	Name of the workflow step icon.
WF	LAST_UPDATED_BY	ID of the user who last updated the workflow.
WF	LAST_UPDATE_DATE	Date the workflow was last updated.
WF	PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow.
WF	REOPEN_WORKFLOW_STEP_ID	ID of the reopened workflow step.
WF	REOPEN_WORKFLOW_STEP_NAME	Name of the reopened workflow step.
WF	SUBWORKFLOW_FLAG	Specifies whether this workflow can be used as a subworkflow.
WF	WORKFLOW_ID	ID of the workflow defined in the table KWFL_WORKFLOWS.
WF	WORKFLOW_NAME	Name of the workflow.
WF	WORKBENCH_WORKFLOW_URL	URL to open the workflow in the PPM Workbench.

Workflow > Workflow Step Tokens

Table A-44. Workflow > Workflow Step tokens (page 1 of 3)

Prefix	Tokens	Description
WF.WFS	ACTION_BUTTON_LABEL	Label displayed on the package or request action button for the workflow step.
WF.WFS	AVERAGE_LEAD_TIME	Average lead time in days defined for the workflow step.
WF.WFS	CREATED_BY	ID of the user who created the workflow step.
WF.WFS	CREATION_DATE	Date the workflow step was created.
WF.WFS	DESCRIPTION	Workflow step description.
WF.WFS	DEST_ENV_GROUP_ID	ID of the destination environment group for the workflow step.
WF.WFS	DEST_ENV_GROUP_NAME	Name of the destination environment group for the workflow step.
WF.WFS	DEST_ENVIRONMENT_ID	ID of destination environment for the workflow step.
WF.WFS	DEST_ENVIRONMENT_NAME	Name of the destination environment for the workflow step.
WF.WFS	ENABLED_FLAG	Flag that indicates whether the workflow step is enabled and can be traversed in a package or request.
WF.WFS	GL_ARCHIVE_FLAG	For GL object migration, a flag that indicates whether to save the GL object being migrated to the HP GL Migrator archive.
WF.WFS	INFORMATION_URL	Workflow step information URL.
WF.WFS	JUMP_RECEIVE_LABEL_CODE	Code for a Jump/Receive workflow step.
WF.WFS	JUMP_RECEIVE_LABEL_NAME	Name of a Jump/Receive workflow step.
WF.WFS	LAST_UPDATED_BY	ID of the user who last updated the workflow step.
WF.WFS	LAST_UPDATE_DATE	Date the workflow step was last updated.

Table A-44. Workflow > Workflow Step tokens (page 2 of 3)

Prefix	Tokens	Description
WF.WFS	OM_ARCHIVE_FLAG	For AOL object migration, a flag that indicates whether to save the AOL object being migrated to the Object*Migrator archive.
WF.WFS	PARENT_ASSIGNED_TO_GROUP_ID	ID of the security group that the current package or request is assigned to (determined by context at time of evaluation).
WF.WFS	PARENT_ASSIGNED_TO_GROUP_NAME	Security group that the current package or request is assigned to (determined by context at time of evaluation).
WF.WFS	PARENT_ASSIGNED_TO_USERNAME	Name of the user to whom the current package or request is assigned (determined by context at time of evaluation).
WF.WFS	PARENT_ASSIGNED_TO_USER_ID	ID of the user to whom the current package or request is assigned (determined by context at time of evaluation).
WF.WFS	PARENT_STATUS	Validation value code of the status of the request that is using the workflow step.
WF.WFS	PARENT_STATUS_NAME	Validation value meaning of the status of the request that is using the workflow step.
WF.WFS	PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow containing the workflow step.
WF.WFS	RESULT_WORKFLOW_PARAMETER_ID	ID of the workflow parameter to which the result of the workflow step is written.
WF.WFS	RESULT_WORKFLOW_PARAMETER_NAME	Name of the workflow parameter to which the result of the workflow step is written.
WF.WFS	SORT_ORDER	Display sequence of the workflow step relative to all other steps in the workflow.

Table A-44. Workflow > Workflow Step tokens (page 3 of 3)

Prefix	Tokens	Description
WF.WFS	SOURCE_ENV_GROUP_ID	ID of the source environment group for the workflow step.
WF.WFS	SOURCE_ENV_GROUP_NAME	Name of the source environment group for the workflow step.
WF.WFS	SOURCE_ENVIRONMENT_ID	ID of the source environment for the workflow step.
WF.WFS	SOURCE_ENVIRONMENT_NAME	Name of the source environment for the workflow step.
WF.WFS	STEP_NAME	Workflow step name.
WF.WFS	STEP_NO	Display sequence of the workflow step relative to all other steps in the workflow.
WF.WFS	STEP_SOURCE_NAME	Name of the workflow step source.
WF.WFS	STEP_TYPE_NAME	Name of the workflow step source type.
WF.WFS	WORKFLOW_ID	ID of the workflow containing the workflow step.
WF.WFS	WORKFLOW_NAME	Name of the workflow containing the workflow step.
WF.WFS	WORKFLOW_STEP_ID	ID of the workflow step in the table KWFL_WORKFLOW_STEPS.

Workflow Step Tokens

Table A-45. Workflow Step tokens (page 1 of 4)

Prefix	Tokens	Description
WFS	ACTION_BUTTON_LABEL	Label displayed on the package or request action button for the workflow step.
WFS	AVERAGE_LEAD_TIME	Average lead time in days defined for the workflow step.
WFS	CREATED_BY	ID of the user who created the workflow step.

Table A-45. Workflow Step tokens (page 2 of 4)

Prefix	Tokens	Description
WFS	CREATION_DATE	Date the workflow step was created.
WFS	DESCRIPTION	Description of the workflow step.
WFS	DEST_ENV_GROUP_ID	ID of the destination environment group for the workflow step.
WFS	DEST_ENV_GROUP_NAME	Name of the destination environment group for the workflow step.
WFS	DEST_ENVIRONMENT_ID	ID of destination environment for the workflow step.
WFS	DEST_ENVIRONMENT_NAME	Name of the destination environment for the workflow step.
WFS	ENABLED_FLAG	Flag that indicates whether the workflow step is enabled and can be traversed in a package or request.
WFS	GL_ARCHIVE_FLAG	For GL object migration, a flag that indicates whether to save the GL object being migrated to the HP GL Migrator archive.
WFS	INFORMATION_URL	Workflow step information URL.
WFS	JUMP_RECEIVE_LABEL_CODE	Code for a Jump/Receive workflow step.
WFS	JUMP_RECEIVE_LABEL_NAME	Name of a Jump/Receive workflow step.
WFS	LAST_UPDATED_BY	ID of the user who last updated the workflow step.
WFS	LAST_UPDATE_DATE	Date the workflow step was last updated.
WFS	OM_ARCHIVE_FLAG	For AOL object migration, the flag that indicates whether to save the AOL object being migrated to the Object*Migrator archive.
WFS	PARENT_ASSIGNED_TO_GROUP_ID	ID of the security group to which the current package or request is assigned (determined by context at time of evaluation).

Table A-45. Workflow Step tokens (page 3 of 4)

Prefix	Tokens	Description
WFS	PARENT_ASSIGNED_TO_GROUP_NAME	Security group to which the current package or request is assigned to (determined by context at time of evaluation).
WFS	PARENT_ASSIGNED_TO_USERNAME	Name of the user to whom the current package or request is assigned (determined by context at time of evaluation).
WFS	PARENT_ASSIGNED_TO_USER_ID	ID of the user to whom the current package or request is assigned (determined by context at time of evaluation).
WFS	PARENT_STATUS	Validation value code of the status of the request that is using the workflow step.
WFS	PARENT_STATUS_NAME	Validation value meaning of the status of the request that is using the workflow step.
WFS	PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow containing the workflow step.
WFS	RESULT_WORKFLOW_PARAMETER_ID	ID of the workflow parameter to which the result of the workflow step is written.
WFS	RESULT_WORKFLOW_PARAMETER_NAME	Name of the workflow parameter to which the result of the workflow step is written.
WFS	SORT_ORDER	Display sequence of the workflow step relative to all other steps in the workflow.
WFS	SOURCE_ENV_GROUP_ID	ID of the source environment group for the workflow step.
WFS	SOURCE_ENV_GROUP_NAME	Name of the source environment group for the workflow step.
WFS	SOURCE_ENVIRONMENT_ID	ID of the source environment for the workflow step.
WFS	SOURCE_ENVIRONMENT_NAME	Name of the source environment for the workflow step.
WFS	STEP_NAME	W workflow step name.
WFS	STEP_NO	Display sequence of the workflow step relative to all other steps in the workflow.

Table A-45. Workflow Step tokens (page 4 of 4)

Prefix	Tokens	Description
WFS	STEP_SOURCE_NAME	Name of the workflow step source.
WFS	STEP_TYPE_NAME	Name of the workflow step source type.
WFS	WORKFLOW_ID	ID of the workflow containing the workflow step.
WFS	WORKFLOW_NAME	Name of the workflow containing the workflow step.
WFS	WORKFLOW_STEP_ID	ID of the workflow step in the table KWFL_WORKFLOW_STEPS.

Request > Field Tokens

The request field tokens are the tokens associated with field groups. Field groups are attached to request header types to enable additional pre-configured fields on requests. Field groups are often delivered as a part of PPM Center best practice functionality. You only have access to field groups associated with products that are licensed at your site.

CMDB Application Tokens

Table A-46. CMDB Application tokens

Prefix	Tokens	Description
REQ.P	KNTA_CMDB_APPLICATION	CMDB application referenced by the request.

Demand Management SLA Tokens

Table A-47. Demand Management SLA tokens

Prefix	Tokens	Description
REQ.P	KNTA_SLA_LEVEL	SLA level.
REQ.P	KNTA_SLA_VIOLATION_DATE	SLA violation date.
REQ.P	KNTA_SLA_SERV_REQUESTED_ON	Service request date.
REQ.P	KNTA_SLA_SERV_SATISFIED_ON	Service satisfied date.

Demand Management Scheduling Tokens

Table A-48. Demand Management Scheduling tokens

Prefix	Tokens	Description
REQ.P	KNTA_EST_START_DATE	Estimated start date.
REQ.P	KNTA_EFFORT	Estimated effort.
REQ.P	KNTA_REJECTED_DATE	Reject date.
REQ.P	KNTA_DEMAND_SATISFIED_DATE	Demand satisfied date.

MAM Impact Analysis Tokens

Table A-49. MAM Impact Analysis tokens

Prefix	Tokens	Description
REQ.P	KNTA_MAM_RFC_ID	MAM RFC ID number.
REQ.P	KNTA_MAM_IMPACT_RESULT	MAM impact results.

Portfolio Management Asset Tokens

Table A-50. Portfolio Management Asset tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_ASSET_DEPENDENCIES	Asset Dependencies.
REQ.P	KNTA_BUSINESS_UNIT	Business Unit.
REQ.P	KNTA_PROJECT_NAME	Asset Name.

Table A-50. Portfolio Management Asset tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_PROJECT_HEALTH	Asset Health.
REQ.P	KNTA_PROJECT_CLASS	Project Class.
REQ.P	KNTA_ASSET_CLASS	Asset Class.
REQ.P	KNTA_BUSINESS_OBJECTIVE	Business Objective.
REQ.P	KNTA_PROJECT_MANAGER	Project Manager.
REQ.P	KNTA_PROJECT_PLAN	Work Plan.
REQ.P	KNTA_BUDGET	Budget.
REQ.P	KNTA_FINANCIAL_BENEFIT	Financial Benefit.
REQ.P	KNTA_STAFFING_PROFILE	Staffing Profile.
REQ.P	KNTA_NPV	Net Present Value.
REQ.P	KNTA_VALUE_RATING	Value Rating.
REQ.P	KNTA_RISK_RATING	Risk Rating.
REQ.P	KNTA_ROI	Return on Investment.
REQ.P	KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
REQ.P	KNTA_TOTAL_SCORE	Total Score.
REQ.P	KNTA_DISCOUNT_RATE	Discount Rate.

Portfolio Management Project Tokens

Table A-51. Portfolio Management Project tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_BUSINESS_UNIT	Business Unit.
REQ.P	KNTA_PROJECT_DEPENDENCIES	Project Dependencies.
REQ.P	KNTA_PROJECT_NAME	Project Name.
REQ.P	KNTA_PROJECT_HEALTH	Project Health.
REQ.P	KNTA_PROJECT_CLASS	Project Class.
REQ.P	KNTA_ASSET_CLASS	Asset Class.
REQ.P	KNTA_BUSINESS_OBJECTIVE	Business Objective.
REQ.P	KNTA_PROJECT_PLAN	Work Plan.

Table A-51. Portfolio Management Project tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_PROJECT_MANAGER	Project Manager.
REQ.P	KNTA_BUDGET	Budget.
REQ.P	KNTA_FINANCIAL_BENEFIT	Financial Benefit.
REQ.P	KNTA_STAFFING_PROFILE	Staffing Profile.
REQ.P	KNTA_NPV	Net Present Value.
REQ.P	KNTA_VALUE_RATING	Value Rating.
REQ.P	KNTA_RISK_RATING	Risk Rating.
REQ.P	KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
REQ.P	KNTA_ROI	Return on Investment.
REQ.P	KNTA_TOTAL_SCORE	Total Score.
REQ.P	KNTA_DISCOUNT_RATE	Discount Rate.
REQ.P	KNTA_PLAN_START_DATE	Start Date.
REQ.P	KNTA_PLAN_FINISH_DATE	Finish Date.

Portfolio Management Proposal Tokens

Table A-52. Portfolio Management Proposal tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_BUSINESS_UNIT	Business Unit.
REQ.P	KNTA_PROJECT_NAME	Project Name.
REQ.P	KNTA_PROJECT_CLASS	Project Class.
REQ.P	KNTA_ASSET_CLASS	Asset Class.
REQ.P	KNTA_BUSINESS_OBJECTIVE	Business Objective.
REQ.P	KNTA_PROJECT_PLAN	Project Plan.
REQ.P	KNTA_PROJECT_DEPENDENCIES	Project Dependencies.
REQ.P	KNTA_PROJECT_MANAGER	Project Manager.
REQ.P	KNTA_PROJECT_TYPE	Project Type.
REQ.P	KNTA_BUDGET	Budget.
REQ.P	KNTA_FINANCIAL_BENEFIT	Expected Benefit.

Table A-52. Portfolio Management Proposal tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_STAFFING_PROFILE	Staffing Profile.
REQ.P	KNTA_NET_PRESENT_VALUE	Net Present Value.
REQ.P	KNTA_VALUE_RATING	Value Rating.
REQ.P	KNTA_RISK_RATING	Risk Rating.
REQ.P	KNTA_RETURN_ON_INVESTMENT	Return on Investment.
REQ.P	KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
REQ.P	KNTA_TOTAL_SCORE	Total Score.
REQ.P	KNTA_DISCOUNT_RATE	Discount Rate.
REQ.P	KNTA_PLAN_START_DATE	Start Date.
REQ.P	KNTA_PLAN_FINISH_DATE	Finish Date.

Program Issue Tokens

Within token builder, Program Issue is an empty folder.

Program Reference Tokens

Table A-53. Program Reference tokens

Prefix	Tokens	Description
REQ.P	KNTA_PROGRAM_REFERENCE	Program reference.

Project Issue Tokens

Table A-54. Project Issue tokens

Prefix	Tokens	Description
non-app	KNTA_ESCALATION_LEVEL	Escalation level.

Project Reference Tokens

Table A-55. Project Issue tokens

Prefix	Tokens	Description
REQ.P	KNTA_MASTER_PROJ_REF	Master project reference.

Project Risk Tokens

Table A-56. Project Issue tokens

Prefix	Tokens	Description
REQ.P	KNTA_IMPACT_LEVEL	Impact level.
REQ.P	KNTA_PROBABILITY	Probability level.

Project Scope Change Tokens

Table A-57. Project Scope Change tokens

Prefix	Tokens	Description
non-app	KNTA_CR_LEVEL	Critical level.
non-app	KNTA_IMPACT_LEVEL	Impact level.

Quality Center Defect Information Tokens

Table A-58. Quality Center Defect Information tokens

Prefix	Tokens	Description
REQ.P	KNTA_QC_DEECT_DOMAIN	Quality Center domain name.
REQ.P	KNTA_QC_DEFECT_PROJECT	Quality Center project.
REQ.P	KNTA_QC_DEFECT_ASSIGNED_TO	Quality Center defect assigned to.
REQ.P	KNTA_QC_DEFECT_NO	Quality Center defect number.
REQ.P	KNTA_QC_DEFECT_STATUS	Quality Center defect status.
REQ.P	KNTA_QC_DEFECT_ATT_URL	Quality Center defect ATT URL.
REQ.P	KNTA_QC_DEFECT_INT_MSG	Quality Center defect instant message.
REQ.P	KNTA_QC_DEFECT_INSTANCE	Quality Center defect instance.

Quality Center Information Tokens

Table A-59. Quality Center Information tokens

Prefix	Tokens	Description
REQ.P	KNTA_QC_DOMAIN	Quality Center domain name.
REQ.P	KNTA_QC_ASSIGNED_TO	Quality Center assigned to user.
REQ.P	KNTA_QC_PROJECT	Quality Center project.
REQ.P	KNTA_QC_REQUIREMENT_NO	Quality Center requirement number.
REQ.P	KNTA_QC_REQUIREMENT_STATUS	Quality Center requirement status.
REQ.P	KNTA_QC_REQUIREMENT_ATT_URL	Quality Center requirement ATT URL.
REQ.P	KNTA_QC_REQUIREMENT_INT_MSG	Quality Center requirement instant messaging.
REQ.P	KNTA_QC_INSTANCE	Quality Center instance.
REQ.P	KNTA_QC_DASHBOARD_SUBJECT	Quality Center dashboard subject.
REQ.P	KNTA_QC_REQUIREMENT_COVERAGE	Quality Center requirement coverage.
REQ.P	KNTA_QC_OPEN_DEFECTS	Quality Center open defects.

Resource Management Work Item Tokens

Table A-60. Resource Management Work Item tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_USR_SCHED_START_DATE	Scheduled Start Date
REQ.P	KNTA_USR_ACTUAL_START_DATE	Actual Start Date
REQ.P	KNTA_USR_SCHED_FINISH_DATE	Scheduled Finish Date
REQ.P	KNTA_USR_ACTUAL_FINISH_DATE	Actual Finish Date
REQ.P	KNTA_SCHED_DURATION	Scheduled Duration
REQ.P	KNTA_ACTUAL_DURATION	Actual Duration
REQ.P	KNTA_SCHED_EFFORT	Scheduled Effort
REQ.P	KNTA_ACTUAL_EFFORT	Actual Effort

Table A-60. Resource Management Work Item tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_WORKLOAD	Workload
REQ.P	KNTA_WORKLOAD_CATEGORY	Workload Category
REQ.P	KNTA_ROLE	Role
REQ.P	KNTA_ALLOW_EXTERNAL_UPDATE	Allow External Update of Actual Effort
REQ.P	KNTA_SCHED_START_DATE	Scheduled Start Date
REQ.P	KNTA_ACTUAL_START_DATE	Actual Start Date
REQ.P	KNTA_SCHED_FINISH_DATE	Scheduled Finish Date
REQ.P	KNTA_ACTUAL_FINISH_DATE	Actual Finish Date
REQ.P	KNTA_SCHED Eff_OVER_DUR	Scheduled Effort Over Duration

Index

A

- application server properties 131
- auto-complete
 - command with delimited output 94
 - command with fixed width output 96
 - configuring the values 93
 - example 100
 - list of users 92
 - long lists 83
 - search fields 88
 - short lists 82
 - user-defined multi-select 98

C

- command conditions 25, 35
 - examples 25
- command language 24
- command tokens 162
- command with delimited output 94
- command with fixed width output 96
- commands
 - create new command 29
 - creating 39
 - special 24
 - triggering from workflow 22
 - validation 81
- component types 69
 - auto-complete 98
 - directory chooser 113
 - file chooser 114
 - file chooser (static environment override) 115
 - file chooser (token-based environment override) 115
- configuring
 - filter field layout 91
- contact 131, 133

- creating 74
 - custom data masks 112
- currency data mask 107
- custom data masks
 - creating 112

D

- date field
 - valid format 117
- directory chooser 113
- dynamic list validations 78
 - command 81
 - SQL 79

E

- entity token
 - application server properties 131
 - command execution 162
 - contacts 131, 133
 - demand management fields 204
 - distributions 134
 - document management 135
 - environment applications 139
 - environments 136, 141
 - extension 64
 - notifications 163, 164, 177
 - organization units 164
 - package lines 168, 171
 - package pending 169
 - program 171, 176
 - project plan 172
 - releases 176
 - requests 178
 - requests pending 182
 - resource pools 184
 - security groups 185
 - skills 185
 - staffing profile 186, 187, 193

- tasks 189
- tasks pending 191
- users 193
- workflow steps 198, 200
- workflows 197

entity tokens

- validation values 196
- validations 195

F

field group tokens 203

- asset 204
- demand management 204
- PMO 203, 204, 208, 209
- program reference 207, 208
- project 205
- proposal 206
- work item 209

fields

- preview layout 92

file chooser 114

- static environment override 115
- token-based environment override 115

filter field layout

- configuring 91

format masks

- for text fields 104

formats

- for tokens 51

N

- nesting tokens 56

New Command window 29, 39

numeric data mask 105

O

object types

- commands and workflow 22

P

percentage data mask 109

R

request field tokens 59

- prefixes 59
- table components 59

S

special command

- parameters tab 34

special command builder 37

- using to build steps 43

special commands 24

- about 34
- building steps with command builder 43
- language 35
- nesting 44
- using 42
- viewing 36

SQL validations 79

- tips 80

static list validations

- configuring 76

swap mode 92

system special commands 24

T

table component validations 118

- column totals 126
- creating rules 121
- defining 119
- rules example 122
- tokens 126

table components

- using tokens in 59

telephone data mask 110

text fields

- configuring 103
- currency 104

- custom format 105
 - format masks 104
 - numeric 104
 - percentage 105
 - telephone 105
- token
 - evaluation example 100
- Token Builder
 - using 65
- token builder
 - about 50
- token evaluation 49
- tokens
 - building 56, 65
 - default format 54
 - environment tokens 63
 - explicit entity format 55
 - field groups 203
 - for use with commands 162
 - formats 51
 - nesting within tokens 56
 - overview 48
 - parameter format 58
 - request fields 59
 - sub-entity format 62
 - user data format 57
 - where to use 48
- dynamic list 78
- file chooser 114
- file chooser (static environment
 - override) 115
- file chooser (token-based environment
 - override) 115
- list of all 73
- overview 69
- seeded 73
- special characters and 72
- SQL 79
- SQL tips 80
- system 73
- table component 118
- text area 1800 117

U

- user-defined special commands 24

V

- validations 74
 - accessing through packages and requests 71
 - command 81
 - command with delimited output 94
 - command with fixed width output 96
 - configuring for static lists 76
 - creating 74
 - date format 117
 - defined 68
 - directory chooser 113

