

# HP Project and Portfolio Management Center

Software Version: 9.30

## Commands, Tokens, and Validations Guide and Reference

Document Release Date: September 2014  
Software Release Date: September 2014



## Legal Notices

### Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

### Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

### Copyright Notice

© 1997 – 2014 Hewlett-Packard Development Company, L.P.

### Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

## Documentation Updates

The title page of this document 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: <https://softwaresupport.hp.com/>.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

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

The following table indicates changes made to this document since the last released edition.

## Support

Visit the HP Software Support site at: <https://softwaresupport.hp.com>.

This website provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website 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 to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: <https://softwaresupport.hp.com/web/softwaresupport/access-levels>.

**HP Software Solutions Now** accesses the HPSW Solution and Integration Portal website. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this website is <http://h20230.www2.hp.com/sc/solutions/index.jsp>.

# Contents

<b>Chapter 1: Getting Started with Commands, Tokens, and Validations .....</b>	<b>8</b>
Related Document .....	9
<b>Chapter 2: Using Commands .....</b>	<b>10</b>
About Commands .....	10
Object Type Commands and Workflows .....	10
Request Type Commands and Workflows .....	11
Special Commands .....	12
Command Language .....	12
Command Conditions .....	13
About the Commands Tab .....	13
Configuring Commands .....	15
Examples of Command Uses .....	18
<b>Chapter 3: Using Special Commands .....</b>	<b>21</b>
About Special Commands .....	21
Special Command Parameters .....	22
Special Command Language .....	23
Nesting Special Commands .....	23
Special Command Conditions .....	23
Using the PPM Workbench to List Special Commands .....	24
About the Special Command Builder .....	25
Configuring Special Commands .....	25
Configuring Security Options for Special Commands .....	31
Specifying Username and Password for PPM Center Special Commands .....	31
Configuring HTTPS Authentication .....	33
Using the Special Command Builder .....	34
Using the Special Command Details Report to List Special Commands .....	36
Special Commands for Processing of Requests .....	36
ksc_copy_request .....	37
Enabling User Data Tab for Validations .....	39

Creating Multiple Field Pairs .....	42
ksc_move_request_workflow .....	45
Special Commands for Staffing Profile .....	46
ksc_clear_staffingprofile_forecast_assignment .....	47
Examples of Using Special Commands .....	47
<b>Chapter 4: Using Tokens .....</b>	<b>49</b>
About Tokens .....	49
Where to Use Tokens .....	49
Token Evaluation .....	50
About the Token Builder .....	51
Token Formats .....	53
Default Format .....	55
Explicit Entity Format .....	56
Nesting Explicit Entity Tokens within Other Tokens .....	57
User Data Format .....	58
Parameter Format .....	59
Request Field Tokens .....	60
Request Field Token Prefixes .....	60
Tokens in Request Table Components .....	60
Sub-Entity Format .....	62
Environment and Environment Application Tokens .....	63
Using the Token Builder .....	65
<b>Chapter 5: Using Validations .....</b>	<b>68</b>
About Validations .....	68
Validations and Special Characters .....	68
Validation Component Types .....	68
Accessing Validations Through Packages and Requests .....	72
Viewing System Validations .....	74
Configuring Validations .....	75
Configuring Text Field Validations .....	79
Sample Telephone Data Mask Formats .....	83
Sample Custom Data Mask .....	83

Configuring Static List Validations .....	83
Configuring Dynamic List Validations .....	87
SQL Validation Tips .....	87
Auto-Complete Matching Tips .....	88
Auto-Complete Values .....	89
Configuring SQL Validations .....	90
SQL Validated Drop Down Lists .....	90
SQL Validated Auto-Complete Lists .....	90
Adding Search Fields to Auto-Complete Validations .....	94
Configuring the Filter Field Layout .....	97
Configuring Validations by Commands .....	99
Configuring Text Area Validations .....	104
Configuring 1800 Character Text Areas .....	105
Configuring Date Field Validations .....	106
Configuring File and Directory Chooser Validations .....	108
Configuring the Table Component .....	109
Configuring Rules .....	113
Configuring User-Defined Multi-Select Auto-Complete Fields .....	116
Example of Token Evaluation and Validation by Command with Delimited Output .....	118
Example of Using a Table Component on an Order Form .....	122
Example Calculating Column Totals .....	125
<b>Chapter 6: Tokens .....</b>	<b>128</b>
Application Server Tokens .....	128
Benefit Tokens .....	128
Benefit > Financial Benefit Line Tokens .....	129
Contact Tokens .....	129
Distribution Tokens .....	130
Document Management Tokens .....	131
Environment Tokens .....	132
Environment > Dest Env Tokens .....	132
Environment > Dest Env > App Tokens .....	134
Environment > Dest Env > Env Tokens .....	136

Environment > Env Tokens .....	139
Environment > Env > App Tokens .....	142
Environment > Env > Env Tokens .....	143
Environment > Source Env Tokens .....	146
Environment > Source Env > App Tokens .....	149
Environment > Source Env > Env Tokens .....	151
Execution Tokens .....	153
Forecast Actuals Tokens .....	154
Notification Tokens .....	155
Organization Unit Tokens .....	156
Package Tokens .....	157
Package > Package Line Tokens .....	159
Package > Pending Reference Tokens .....	160
Package Line Tokens .....	161
Program Tokens .....	162
Project Tokens .....	162
Project Detail Tokens .....	166
Release Tokens .....	166
Release > Distribution Tokens .....	167
Report Submission Tokens .....	168
Request Tokens .....	168
Request > Pending Reference Tokens .....	171
Request > Field Tokens .....	173
Request Detail Tokens .....	173
Request Detail > Field Tokens .....	173
Resource Pool Tokens .....	173
Security Group Tokens .....	174
Skill Tokens .....	175
Staffing Profile Tokens .....	175
Step TXN (Transaction) Tokens .....	176
System Tokens .....	177
Task Tokens .....	178

Tasks > Pending Tokens .....	180
Time Management Notification Tokens .....	181
User Tokens .....	182
Validation Tokens .....	184
Validation > Value Tokens .....	184
Workflow Tokens .....	185
Workflow > Workflow Step Tokens .....	186
Workflow Step Tokens .....	188
Request > Field Tokens .....	190
CMDB Application Tokens .....	190
Demand Management SLA Tokens .....	191
Demand Management Scheduling Tokens .....	191
MAM Impact Analysis Tokens .....	191
Portfolio Management Asset Tokens .....	192
Portfolio Management Project Tokens .....	192
Portfolio Management Proposal Tokens .....	193
Program Issue Tokens .....	194
Program Reference Tokens .....	195
Project Issue Tokens .....	195
Project Reference Tokens .....	195
Project Risk Tokens .....	195
Project Scope Change Tokens .....	196
Quality Center Defect Information Tokens .....	196
Quality Center Information Tokens .....	196
Resource Management Work Item Tokens .....	197
 Send Documentation Feedback .....	 199

# Chapter 1: Getting Started with Commands, Tokens, and Validations

Commands, tokens, and validations are used throughout 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. ["Using Commands" on page 10](#) 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. ["Using Special Commands" on page 21](#) 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 the following types of tokens:

- Standard
- Custom

["Using Tokens" on page 49](#) 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. ["Using Validations" on page 68](#) provides detailed information on how to use tokens.

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



## Related Document

*Demand Management Configuration Guide* also includes information related to using commands, tokens, and validations.

## Chapter 2: Using Commands

### 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 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:
  - Make the workflow step an execution type step.
  - Set 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 13](#).

## Request Type Commands and Workflows

Like object type commands, request type commands define the execution layer within 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:
  - Make the workflow step 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 the next page](#)

## 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.

**Note:** For more information about special commands, see ["Using Special Commands" on page 21](#).

## 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).

**Note:** For more information about special commands, see ["Using Special Commands" on page 21](#).

## 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. Example conditions" below](#) lists some example conditions. The condition can include tokens. For more information, see ["Using Tokens" on page 49](#).

**Table 2-1. Example conditions**

Condition	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.

**Note:** 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. Commands tab" below shows the **Commands** tab in the Object Type window.

**Figure 2-1. Commands tab**

Object Type : PPM Report Type Migrator

Object Type Name: PPM Report Type Migrator

Description: PPM Report Type Migrator

Extension:  Object Name Column: PARAMETER2

Object Category: Standard Objects Object Revision Column:

Meta Layer View: MPKGL\_ PPM\_REPORT\_TYPE\_MIGR

Enabled: ☒ Yes ☐ No

Fields | Layout | **Commands** | Ownership

Command	Con
Report debugging info	
Default filename	
Export using given password	[P.MI]
Transfer content bundle	[P.MI]
Import using given password	[P.MI]

Command Steps

Command
# ksc_comment Source password is [P.SOURCE_PASSWD]
ksc_mig_default_filename
ksc_mig_extract PASSWD="[P.SOURCE_PASSWD]"
ksc_copy_server_server FILE_TYPE="BIN"
ksc_mig_import PASSWD="[P.DEST_PASSWD]"

Buttons: + All, - All, New Cmd, Edit Cmd, Copy Cmd, Remove, Up, Down

Buttons: OK, Save, Cancel

Ready

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.

**Note:** 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.
- 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 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:

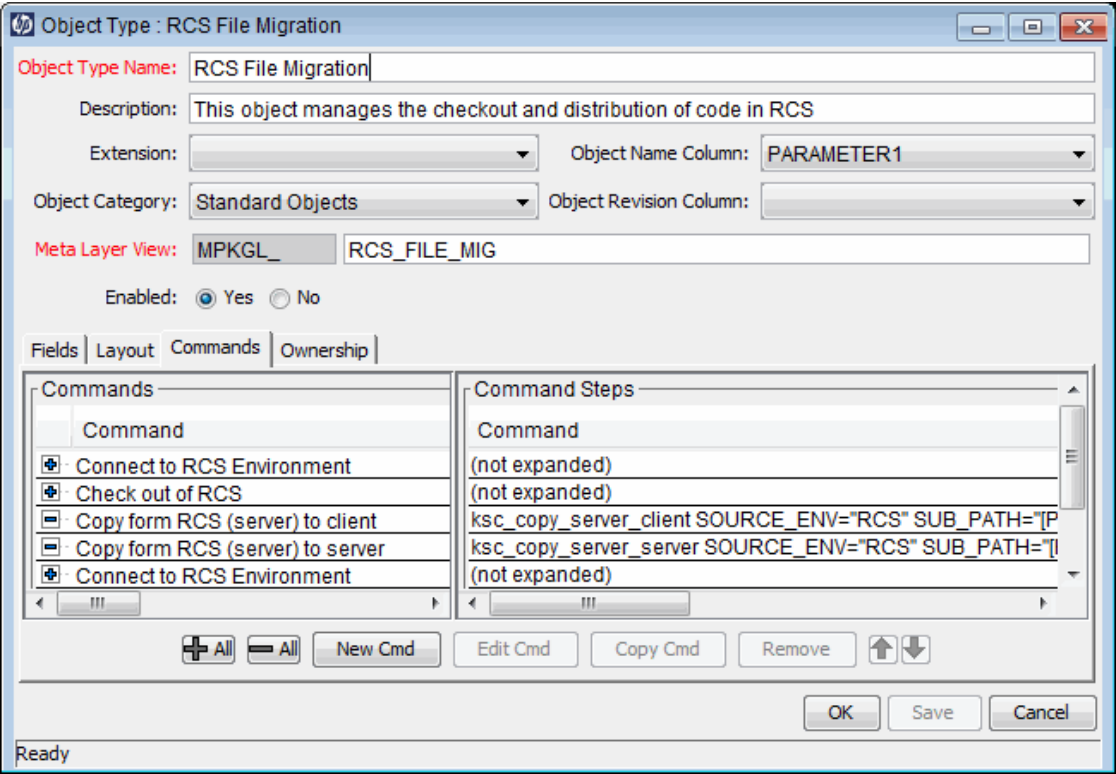
1. Log on to PPM Center.
2. From the menu bar, select **Open > Administration > Open Workbench**.

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
<b>Command</b>	Command name.
<b>Condition</b>	Specific conditions under which the command steps are to be executed. This step is optional. For more information, see <a href="#">"Command Conditions" on page 13</a> .
<b>Description</b>	Command description. This step is optional. For more information, see <a href="#">"Command Conditions" on page 13</a> .
<b>Timeout(s)</b>	Length of time (in minutes) to run the command before stopping. This setting is useful if a command hangs or takes too long to execute.
<b>Enabled</b>	Use the <b>Yes</b> and <b>No</b> option buttons to enable and disable the command.
<b>Steps</b>	Specify at least one command step.

- Click **Tokens** to open the Token Builder window and find a token to add to the command step. For information about tokens, see ["Using Tokens" on page 49](#).

- 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 ["Using Special Commands" on page 21](#).
  - 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

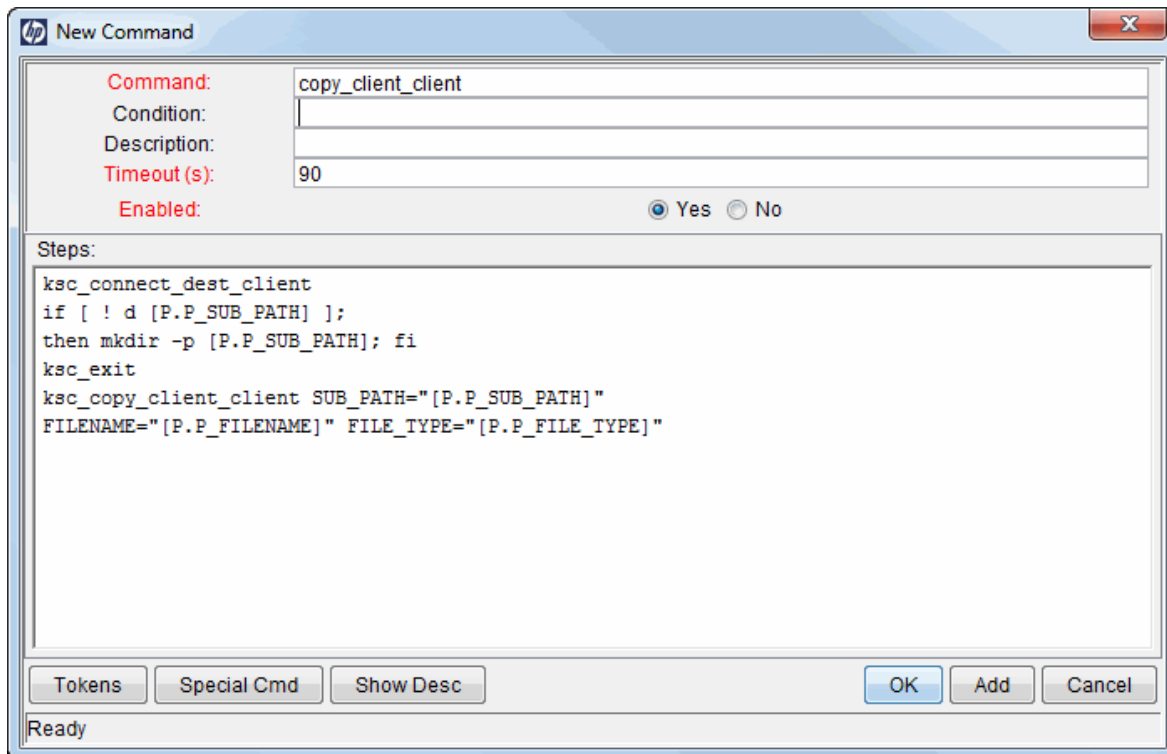
This section provides examples of commands that can be defined using the New Command window.

**To copy a file from one environment to another use the following 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," use the following 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, use the following 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, use the following command:**

ksc\_store

**Command Steps:**

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

**To run a report using UNIX, use the following 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 , use the following 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"
```

# Chapter 3: 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 commands are added to command steps directly in the entity windows (for object types, request types, report types, validations and workflows). For example, ["Figure 3-1. RCS File Migration object type" on the next page](#) shows an example of an object type that was generated using a combination of special commands.

**Figure 3-1. RCS File Migration object type**

Object Type : RCS File Migration

Object Type Name: RCS File Migration

Description: This object manages the checkout and distribution of code in RCS

Extension: Object Name Column: PARAMETER1

Object Category: Standard Objects Object Revision Column:

Meta Layer View: MPKGL\_ RCS\_FILE\_MIG

Enabled: ☒ Yes ☐ No

Fields | Layout | Commands | Ownership

Commands

Command
Connect to RCS Environment
Check out of RCS
Copy form RCS (server) to client
Copy form RCS (server) to server
Connect to RCS Environment

Command Steps

Command
(not expanded)
(not expanded)
ksc_copy_server_client SOURCE_ENV="RCS" SUB_PATH="[P
ksc_copy_server_server SOURCE_ENV="RCS" SUB_PATH="[I
(not expanded)

Buttons: All, All, New Cmd, Edit Cmd, Copy Cmd, Remove, Up, Down

OK Save Cancel

Ready

## 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 25](#)

**Note:** 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.

## 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.

## 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. Example conditions" on the next page](#) 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.

**Note:** 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 ["Using Tokens" on page 49](#) 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 **Open > Administration > Open Workbench**.

The PPM Workbench opens.

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

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.

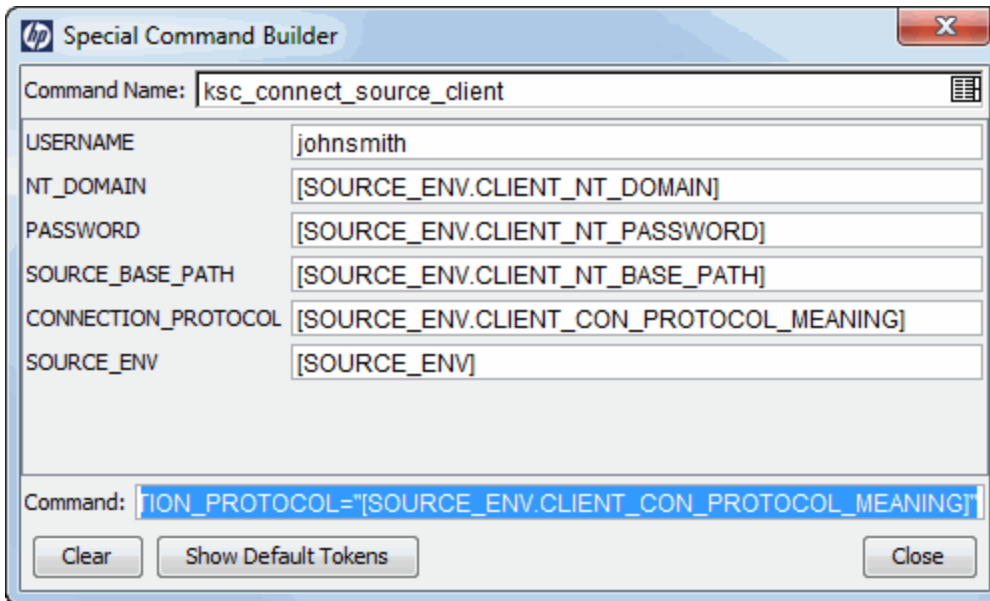
**Tip:** You can also use the Special Command Details report to view a list of special commands on your PPM Center instance. For information on how to access and run this report, see ["Using the Special Command Details Report to List Special Commands" on page 36](#).



## About the Special Command Builder

The Special Command Builder ("Figure 3-2. Special Command Builder" below) helps you format command steps quickly and 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 3-2. Special Command Builder**



Special Command Builder

Command Name: ksc\_connect\_source\_client

USERNAME johnsmith

NT\_DOMAIN [SOURCE\_ENV.CLIENT\_NT\_DOMAIN]

PASSWORD [SOURCE\_ENV.CLIENT\_NT\_PASSWORD]

SOURCE\_BASE\_PATH [SOURCE\_ENV.CLIENT\_NT\_BASE\_PATH]

CONNECTION\_PROTOCOL [SOURCE\_ENV.CLIENT\_CON\_PROTOCOL\_MEANING]

SOURCE\_ENV [SOURCE\_ENV]

Command: TION\_PROTOCOL="[SOURCE\_ENV.CLIENT\_CON\_PROTOCOL\_MEANING]"

Clear Show Default Tokens Close

For information about how to use the Special Command Builder, see ["Using the Special Command Builder" on page 34](#).

## Configuring Special Commands

To configure a new special command:

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

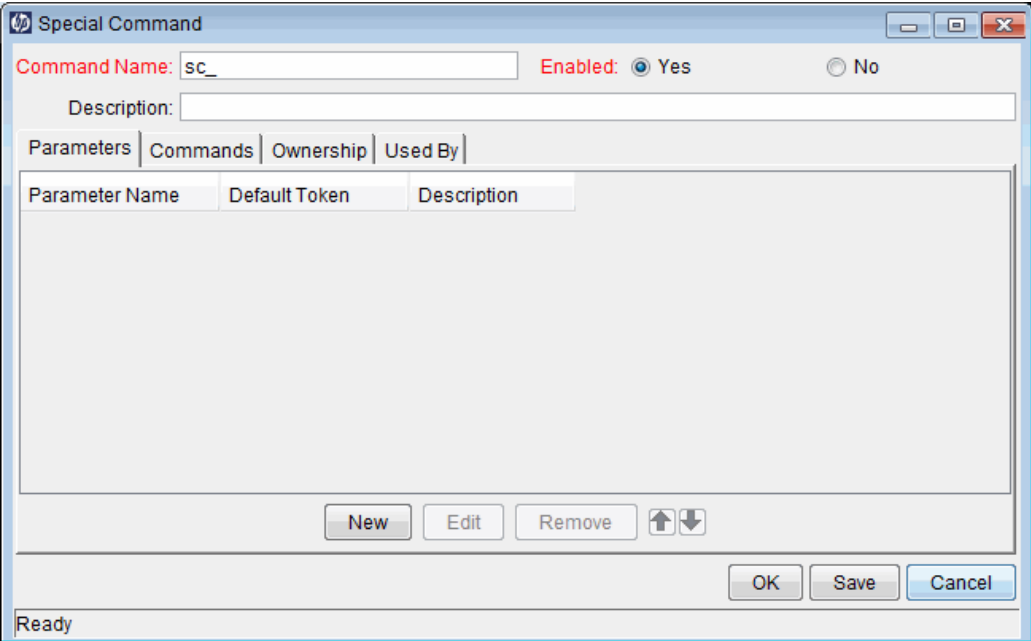
The PPM Workbench opens.

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

The Special Command Workbench opens.

4. Click **New Special Command**.

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.

Special Command

Command Name:  Enabled: ☒ Yes ☐ No

Description:

Parameters | Commands | Ownership | Used By |

Parameter Name	Default Token	Description
FILENAME	P.P_FILENAME	Filename

New Edit Remove ↑ ↓

OK Save Cancel

Ready

- b. Click **New**.

The Parameter: New window opens.

Parameter: New

Name:

Description:

Default Token:

Tokens OK Add Cancel

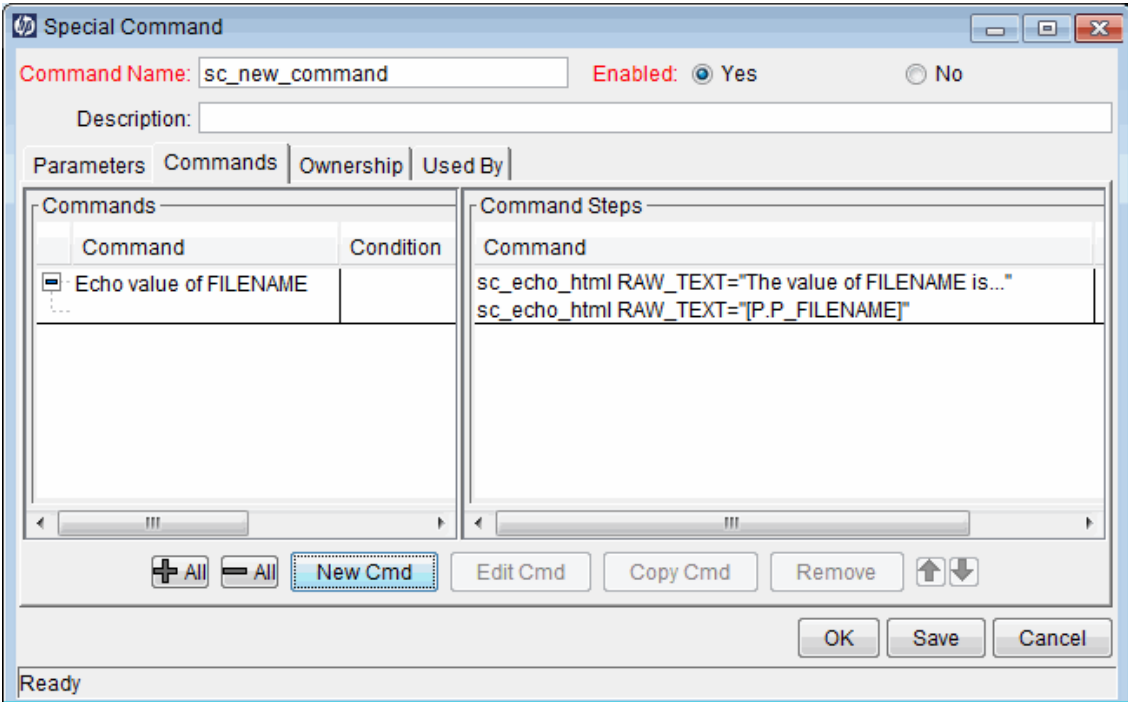
Ready

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

Field Name	Description
<b>Name</b>	The name of the new parameter.
<b>Description</b>	A description of the new parameter.

<b>Default Token</b>	The default token name. Type the token name, or click <b>Tokens</b> 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
<b>Command</b>	Command name.
<b>Condition</b>	Specific conditions under which the command steps are to be executed. This step is optional. For more information, see <a href="#">"Special Command Conditions" on page 23</a> .
<b>Description</b>	Command description. This step is optional. For more information, see .
<b>Timeout(s)</b>	Amount of time (in minutes) to run the command before stopping. This setting is useful if a command hangs or takes too long to execute.
<b>Enabled</b>	<b>Yes</b> and <b>No</b> option buttons enable or disable the command
<b>Steps</b>	Specify at least one command step.

- 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 ["Using Tokens" on page 49](#).

- 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 ["Using Special Commands" on page 21](#).
  - 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.

The screenshot shows the 'Special Command' dialog box with the 'Ownership' tab selected. The 'Command Name' is 'sc\_new\_command' and 'Enabled' is set to 'Yes'. The 'Description' field is empty. The 'Parameters' tab is also visible. Under 'Give ability to edit this Special Command to:', the option 'All users with the Edit Special Commands Access Grant' is selected. Below this, there is a table with two columns: 'Security Group' and 'Description'. The table is currently empty. At the bottom of the table are 'Add' and 'Remove' buttons. At the bottom of the dialog are 'OK', 'Save', and 'Cancel' buttons. The status bar at the bottom left says 'Ready'.

Security Group	Description
----------------	-------------

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**.

## Configuring Security Options for Special Commands

Security options for the `ksc_export_XXX` commands can be configured and used with PPM Center Web services.

## Specifying Username and Password for PPM Center Special Commands

To specify the username, password, and Axis framework used to communicate with the PPM Server for the `ksc_export_XXX` commands, complete the following steps:

1. On the PPM Server, open the `<PPM_InstallPath>/conf/webservices.conf` file for edit.
2. In the `remoteServer` tag and specify the following values:

```
<remoteServer baseUrl="http://<Hostname>:<Port>/<Path>"
username="<Username>"
password="<Password>"
encUsername="ppmservice"
axisVersion="<AxisVersion>"
/>
```

where

<code>&lt;Hostname&gt;</code>	represents the host name of the PPM Server.
<code>&lt;Port&gt;</code>	represents the port number for Web access.

<code>&lt;Path&gt;</code>	represents the path where PPM Center special commands are accessed (via the Web).
<code>&lt;Username&gt;</code>	represents the username for the PPM Server administrative account.
<code>&lt;Password&gt;</code>	<p>represents the password for the PPM Server administrative account.</p> <p>PPM Center supports the WS-Security UsernameToken profile. The UsernameToken profile (as specified by WS-Security) sends the username and password through SOAP headers. The password has two format options:</p> <ul style="list-style-type: none"> <li>◦ <b>Clear-text password.</b> Sends the password to the Web service application and the application is responsible for performing the authentication.</li> <li>◦ <b>Digest password.</b> Asks the Web service application for user password and then validates the returned "digest."</li> </ul>
<code>&lt;AxisVersion&gt;</code>	<p>represents the Axis framework version.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>◦ AXIS1</li> <li>◦ AXIS2</li> </ul> <p>The default value is AXIS2 and is used for version 7.1, 7.5, and 8.00 Web services. Specify AXIS1 for version 6.0 and 7.0 Web services.</p>

For example:

```
<remoteServer baseUrl="http://localhost:8080/itg/
ppmservices/DemandService"
username="admin"
password="admin"
encUsername="ppmservice"
axisVersion="AXIS2"
/>
```

3. (Optional) Repeat [Step 2](#) if you want to enable special commands for additional application modules or remote systems.
4. Save and close the `webservices.conf` file.



## Configuring HTTPS Authentication

The PPM Center `ksc_export_XXX` special commands support HTTPS through JSSE. The standard trust store retrieval path for JSSE is followed for HTTPS connections.

To enable HTTPS on the PPM Server, see the *Installation and Administration Guide*.

To send `ksc_export_XXX` special command requests in SSL mode, complete the following steps:

1. On the PPM Server, open the `<PPM_InstallPath>/server.conf` file for edit.
2. Add (or modify) the following three parameters:

```
com.kintana.core.server.WEB_SERVICES_SSL_KEYSTORE=<Keystore>
```

```
com.kintana.core.server.WEB_SERVICES_SSL_KEYSTORE_PASSWORD=<Password>
```

```
com.kintana.core.server.WEB_SERVICES_SSL_TRUSTSTORE=<Truststore>
```

where

<code>&lt;Keystore&gt;</code>	represents the keystore file.
<code>&lt;Password&gt;</code>	represents the password.
<code>&lt;Truststore&gt;</code>	represents the trust store.  If no separate trust store is specified, the <code>WEB_SERVICES_SSL_KEYSTORE</code> can be used as a trust store.

3. Save and close the `server.conf` file.

### Creating the Keystore Using Keytool

There are several methods for creating a keystore, using `keytool` is only one of them.

If PPM Center is used to call Web services on a remote server using HTTPS, import the certificate that was used to sign the remote server's SSL certificate into the JRE's trusted keystore.

This is required if the certification authority (CA) is not one of the known certificate authorities that ship with the Java Runtime Environment (such as Verisign).

If you use another certification authority, such as an authority internal to your organization:

Import the certificate into the keystore, using the following command:

```
keytool -import -trustcacerts -alias systemca  
-file <CA_certificate>  
-keystore <JRE_home>/lib/security/jssecacerts  
-storepass <password>
```

where

<CA_certificate>	represents the name of the file containing the certificate (from the certification authority) used to sign the remote server's SSL certificate
<JRE_home>	represents the location of the Java Runtime Environment installation used for the local PPM Center instance
<password>	represents the password used for the trusted certificate keystore

**Note:** When using HTTPS, the base URL specified in the `webservices.conf` and `server.conf` files should use the `https://` prefix instead of `http://`.

## 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 **Open > Administration > Open Workbench**.

The PPM Workbench opens.

3. From the shortcut bar, select **Deployment 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.

**Note:** 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. Provide 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.

**Note:** 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.

## 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 **Create > 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.

**Tip:** 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 24](#).

## Special Commands for Processing of Requests

To facilitate automated processing of Demand Management requests, you can use the ["ksc\\_copy\\_request" on the next page](#) and ["ksc\\_move\\_request\\_workflow" on page 45](#) special commands.

## ksc\_copy\_request

This special command makes a copy of a specified request automatically, with no user intervention. As distinct from the "create request" workflow command which raises the standard request form in the user interface for the user to manually fill in and submit.

In addition, when the `ksc_copy_request` special command is run, the resulting ID of the new request must be made available so that it can be captured and used in subsequent commands. This is similar to the `ksc_set` command, in which you can specify a "temporary token" and assign a value to it. In this case, a temporary token named "[COPIED\_REQUEST\_ID]" holds the ID of the new request, and is available within the same run context. If it is necessary to persist this new request ID on the existing request driving the commands, then it should be stored to a request field using the `ksc_store` special command.

**Table 3-2. ksc\_copy\_request parameters**

Parameter	Description
FROM_REQUEST_ID	The ID of the request from which to make the copy. This is a required parameter, and should default to the token for the ID of the current request ("[REQ.REQUEST_ID]").
REQUEST_TYPE_ID	The ID of the request type to use when creating the new request. This is an optional parameter, and should be specified only if it is desired that the new request be of a request type different than that of the request being copied (specified by the FROM_REQUEST_ID parameter). If this parameter is left blank, then the new request created by this special command will be of the same request type as the one specified by the FROM_REQUEST_ID parameter.
WORKFLOW_ID	The ID of the workflow to use when creating the new request. This is an optional parameter, and should be NULL by default.
COPY_FIELDS	Valid values are "Y" or "N." If "Y" then copy the values of every field whose token maps to a token in the new request. Default value is "Y."  <b>NOTE:</b> Resources are not copied from the original request because, in most cases, the new request will be worked on by different resources.
COPY_NOTES	Valid values are "Y" or "N." If "Y" then copy all the notes from the original request to the new request. The default value is "N" since it is reasoned that the notes from the original request are typically not relevant for the new request, and thus will not be copied in most cases.
CREATE_REFERENCE	Valid values are "Y" or "N." If "Y" then create a reference between the

**Table 3-2. ksc\_copy\_request parameters, continued**

Parameter	Description
	new request and the original request. The default value is "Y."
REF_RELATIONSHIP_ID	<p>Specifies the relationship of the reference, if CREATE_REFERENCE=Y, otherwise this parameter is ignored. Valid values are:</p> <ul style="list-style-type: none"> <li>• 12 (Duplicate Request)</li> <li>• 13 (Original of Duplicate Requests)</li> <li>• 14 (Parent of this Request)</li> <li>• 15 (Child of this Request, this is the default value)</li> <li>• 16 (Related to this Request)</li> <li>• 422 (Successor)</li> <li>• 423 (Predecessor)</li> </ul>
SUBMIT	Valid values are "Y" or "N." If "Y," then submit the new request. The default value is "Y."
STATUS_NAME	The status name to set for the new request. The default value is NULL and typical configurations will want to use the initial status name specified by the new request's type configurations. Note that if a status name is provided, its value should be set after the request is submitted (if SUBMIT=Y).
PROCESS_RULE	Valid values are "Y" or "N." If "Y," then process rules for the "Apply on creation" rule event. If "N," then rules are not processed for the "Apply on creation" rule event. The default value is "Y."
USER_ID	Specifies creator user ID of the new request.
VALIDATION_NAME	<p>Specifies a validation to store copied fields with different tokens.</p> <p>The value of the destination token will be replaced by the value of the source token.</p> <p>The source token, source visible token, token type, and destination token are stored in the user data of validation values. You can configure multiple validation values to store multiple pairs of source and target fields.</p> <p>For information about configuring user data for validation values to store field token mappings between source and target requests, see <a href="#">"Enabling User Data Tab for Validations" on the next page.</a></p>

## Enabling User Data Tab for Validations

To use `ksc_copy_request` for the purpose of copying request, you need to create a validation of Auto Complete List component type and validated by List, then enable the User Data tab for the validation value by defining a user data for it.

### Create a Validation

To configure a validation,

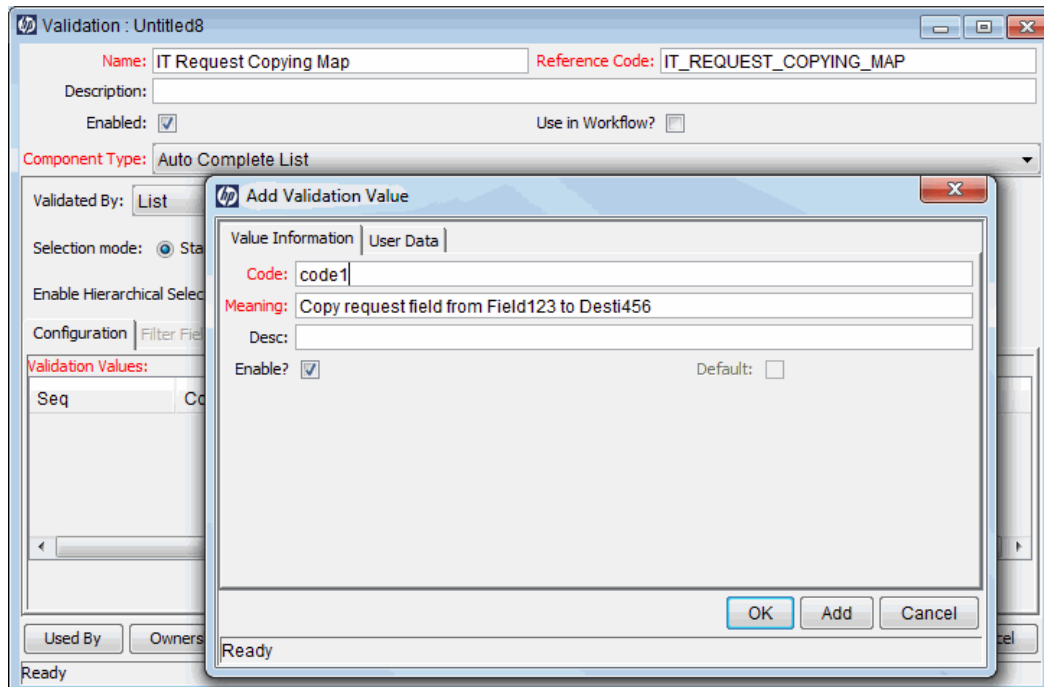
1. Create a new validation with the following information:
  - **Name:** Name of the new validation. For example, IT Request Copying Map.
  - **Reference Code:** Accept the default value.
  - **Enabled:** Select the checkbox.
  - **Component Type:** Auto Complete List.
  - **Validated By:** List

For more information about configuring validations, see ["Configuring Validations" on page 75](#).

2. In the Validation Values section, click **New** to add a validation value.

The Add Validation Value window opens.

3. Provide the information for the validation value.



4. Click **OK**.

### Enable User Data Tab for Validations

To define a user data and enable the User Data tab for a validation,

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

The User Data Workbench opens.

2. Click **New User Data Context**.

The User Data Context window opens to the Fields tab.

3. Click **User Data Type** auto-complete.

The **User Data Type** field displays the value **Validation Value User Data**, the **Context Field** field displays the value **Validation Name**, and the **Scope** field displays **Context**.

4. Click the **Context Value** auto-complete, and select the validation you just created in "[Create a Validation](#)" on the previous page. In this example, **IT Request Copying Map**.



5. On the Fields tab, click **New**.
6. Create four fields by strictly following the pairing order below:

Field Prompt	Token	User Data Col.
source token	SOURCE_TOKEN	USER_DATA1
source visible token	SOURCE_VISIBLE_TOKEN	USER_DATA2
token type	TOKEN_TYPE	USER_DATA3
destination token	DESITINATION_TOKEN	USER_DATA4

User Data Context: Validation Value User Data

User Data Type: Validation Value User Data

Context Field: Validation Name Context Value: IT Request Copying Map

Enabled: ☒ Yes ☐ No Scope: Context

Meta Layer View:

Fields | Layout

Prompt	Token	User Data Col.	Display...	Component T...	Validation	Requir...	Display Only
destination token	DESTINATION_TOKEN	USER_DATA4	Y	Text Field	Text Field - 40	N	N
token type	TOKEN_TYPE	USER_DATA3	Y	Text Field	Text Field - 40	N	N
source visible token	SOURCE_VISIBLE_TOKEN	USER_DATA2	Y	Text Field	Text Field - 40	N	N
source token	SOURCE_TOKEN	USER_DATA1	Y	Text Field	Text Field - 40	N	N

New Edit Remove

OK Save Cancel

Ready

**Note:**

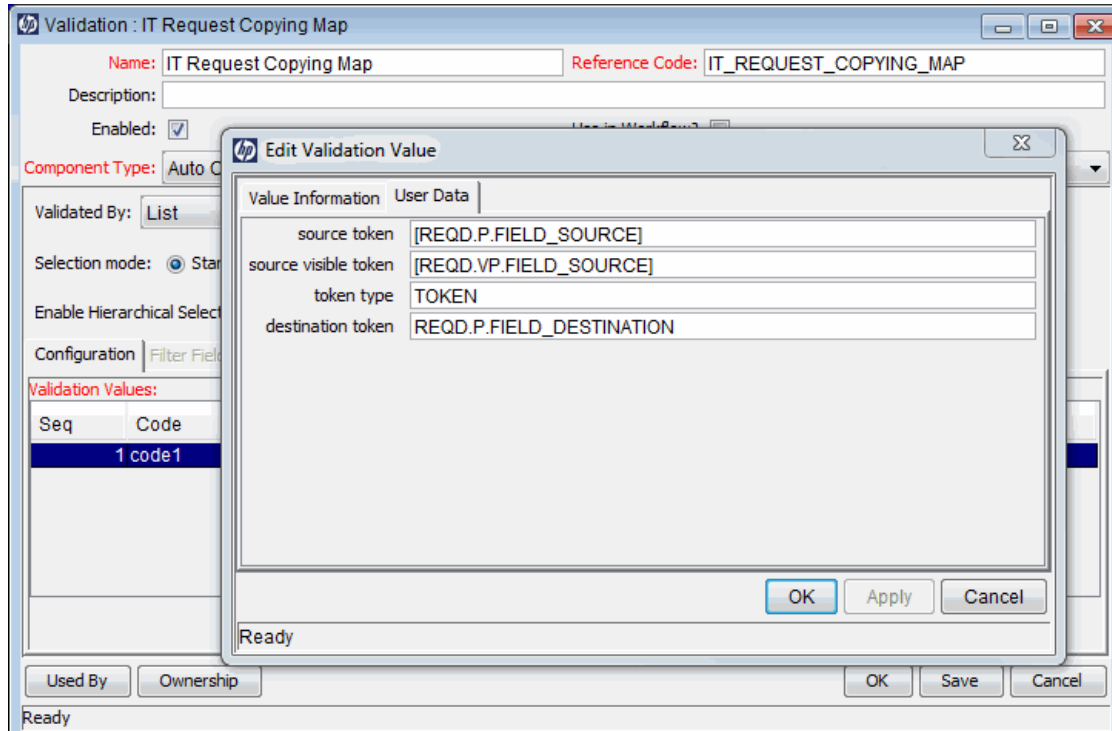
- You can use different names for field prompts, but be sure that USER\_DATA1 is paired to source token, USER\_DATA2 to source visible token, USER\_DATA3 to token type, and USER\_DATA4 to destination token.
- For token type, if you want to use both TOKEN and SQL, you may use drop down list as the component type and include TOKEN and SQL options in its validation.

7. Click **OK**.

The user data is defined for validation **IT Request Copying Map**.

8. (Optional) Go to Validation Workbench and open the validation **IT Request Copying Map**.

Open the validation value, the User Data tab is now enabled in the Validation Value window.



## Creating Multiple Field Pairs

The user data for one validation value stores one pair of source and target fields. If you need to copy multiple request fields, you can add and configure more validations values to specify and store multiple source and target field tokens.

Taking validation **IT Request Copying Map** as an example, to create multiple field pairs,

1. In the Validation Workbench, open the validation **IT Request Copying Map**.

The Validation: IT Request Copying Map window opens.

2. In the Validation Values section, click **New**.

The Add Validation Value window opens to Value Information tab.

3. Provide the information for the validation value on the Value Information tab.
4. Click **User Data** tab, provide token values of source and target request fields for the following fields:

Field Prompt	Description
<b>source token</b>	<p>Invisible token of the source field to be copied from.</p> <p>Value example:</p> <ul style="list-style-type: none"> <li>◦ When token type is TOKEN: [REQD.P.FIELD_SOURCE];</li> <li>◦ When token type is SQL: select '[REQD.P.FIELD_SOURCE]' from dual</li> </ul>
<b>source visible token</b>	<p>Visible token of the source field to be copied from.</p> <p>Value example:</p> <ul style="list-style-type: none"> <li>◦ When token type is TOKEN: [REQD.VP.FIELD_SOURCE];</li> <li>◦ When token type is SQL: select '[REQD.VP.FIELD_SOURCE]' from dual</li> </ul>
<b>token type</b>	<p>Specifies supported token type. Valid value: TOKEN or SQL.</p> <p><b>Note:</b> Only TOKEN and SQL are supported.</p>
<b>destination token</b>	<p>Token of the destination field to be copied to.</p> <p>Value example:</p> <ul style="list-style-type: none"> <li>◦ When token type is TOKEN: REQD.P.FIELD_DESTINATION;</li> <li>◦ When token type is SQL: REQD.P.FIELD_DESTINATION.</li> </ul> <p><b>Note:</b> No square brackets for the destination token value.</p>

For example, if you want to copy the value from field **Field123** to field **Dest456**,

- When token type is TOKEN,

The screenshot shows a dialog box titled "Add Validation Value" with an HP logo. It has two tabs: "Value Information" and "User Data". The "Value Information" tab is active. It contains four input fields: "source token" with the value "[REQD.P.FIELD123]", "source visible token" with the value "[REQD.VP.FIELD123]", "token type" with the value "TOKEN", and "destination token" with the value "REQD.P.DESTI456". At the bottom right are three buttons: "OK", "Add", and "Cancel". At the bottom left, the status "Ready" is displayed.

Field	Value
source token	[REQD.P.FIELD123]
source visible token	[REQD.VP.FIELD123]
token type	TOKEN
destination token	REQD.P.DESTI456

- When token type is SQL,

The screenshot shows the same "Add Validation Value" dialog box, but with the "token type" field set to "SQL". The "source token" field now contains the SQL query "select '[REQD.P.FIELD123]' from dual", and the "source visible token" field contains "select '[REQD.VP.FIELD123]' from dual". The "destination token" field remains "REQD.P.DESTI456". The "OK", "Add", and "Cancel" buttons are at the bottom right, and the status "Ready" is at the bottom left.

Field	Value
source token	select '[REQD.P.FIELD123]' from dual
source visible token	select '[REQD.VP.FIELD123]' from dual
token type	SQL
destination token	REQD.P.DESTI456

5. Click **Add**.
6. Repeat [Step 2"Creating Multiple Field Pairs" on page 42](#)through [Step 5](#) to add more validation values.

## ksc\_move\_request\_workflow

This special command causes a workflow event to occur on the specified request.

**Table 3-3. ksc\_move\_request\_workflow parameters**

Parameter	Description
REQUEST_ID	The ID of the request to take action on. This is a required parameter, and should default to the token for the ID of the current request (" [REQ.REQUEST_ID]").
FROM_WORKFLOW_STEP_SEQ	The sequence number of the step to take action on. This is a required parameter, and should default to the token for the active step of the current request (" [WFS.STEP_NO]").
EVENT_NAME	<p>The type of workflow event to start.This is a required parameter. The following events are supported (same as the workflow transition database open interface):</p> <ul style="list-style-type: none"> <li>• <b>INSTANCE_SET_CREATE.</b> For request submission.</li> <li>• <b>APPROVAL_VOTE.</b> For decision step. If this event is specified, then the RESULT_VISIBLE_VALUE parameter should also be specified to indicate which of the step's outcomes to choose.</li> <li>• <b>APPROVAL_DELEGATE.</b> For decision step being delegated. If this event is specified, then the DELEGATE_TO_USERNAME parameter should also be provided.</li> <li>• <b>EXECUTION_EXECUTE.</b> For execution step.</li> <li>• <b>EXECUTION_SCHEDULE.</b> For execution step being scheduled. If this event is specified, then the SCHEDULE_DATE parameter must be provided.</li> <li>• <b>BYPASS_EXECUTION.</b> For execution step being bypassed. If this event is specified, then the RESULT_VISIBLE_VALUE parameter should also be specified to indicate which result value to override the execution step with.</li> </ul>

**Table 3-3. ksc\_move\_request\_workflow parameters, continued**

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>RESULT_OVERRIDE.</b> For an active step being overridden. If this event is specified, then the RESULT_VISIBLE_VALUE parameter must be specified to indicate which of the step's outcomes to choose.</li> <li>• <b>INSTANCE_SET_CANCEL.</b> For cancelling the request.</li> <li>• <b>FORCE_TRANSITION.</b> For forcing a transition from the step specified in FROM_WORKFLOW_STEP_SEQ to an arbitrary destination step, which may not have a valid transition in the workflow configuration. If this event is specified, then you must specify the RESULT_VISIBLE_VALUE and TO_WORKFLOW_STEP_SEQ parameters, to indicate the result value shown for the source step, and which step should be the destination of the forced transition.</li> </ul>
RESULT_VISIBLE_VALUE	The visible value of the desired result value for the workflow action, if applicable. This is the value the user would choose in the user interface while taking action on a workflow, and should default to NULL. Most of the events require the result visible value to be specified (see the EVENT_NAME for specifics).
SCHEDULE_DATE	The date to schedule an execution event. This should default to NULL. Only used with EXECUTION_SCHEDULE events, otherwise it is ignored.
DELEGATE_TO_USER_ID	The ID of the user to whom the current user's approval vote should be delegated. This should default to NULL. Only used with APPROVAL_DELEGATE events, otherwise it is ignored.
TO_WORKFLOW_STEP_SEQ	The sequence number of the desired step to transition to. This should default to NULL. Only used with FORCE_TRANSITION events, otherwise it is ignored.

## Special Commands for Staffing Profile

- ["ksc\\_clear\\_staffingprofile\\_forecast\\_assignment" on the next page](#)

## ksc\_clear\_staffingprofile\_forecast\_assignment

This special command clears the forecast and assignments of a staffing profile starting from a specified date.

**Note:** The staffing profile must be Completed if you want to clear its forecast and assignments.

Parameter	Description
STAFFING_PROFILE_ID	ID of the staffing profile whose forecast and assignments you want to clear
CLEAR_FROM_DATE	The start date of the forecast and assignments you want to clear. It must be later than the start date of the staffing profile.

For use example of this special command, see ["Examples of Using Special Commands" below](#).

## Examples of Using Special Commands

This section provides examples of special commands.

**To copy a file from one server to another server, use the following command:**

```
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, use the following command:**

```
ksc_mig_import
```

**Special Command Example:**

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

**To change the status of a project, use the following command:**

```
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, use the following command:**

```
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
```

**To copy specified requests with the original creator ID and the request fields, use the following command:**

```
ksc_copy_request
```

**Special Command Example:**

```
ksc_copy_request FROM_REQUEST_ID="[REQ.REQUEST_ID]"
USER_ID="[USER_ID]" VALIDATION_NAME="[VALIDATION_NAME]"
```

**To clear forecast and assignments of a staffing profile:**

```
ksc_clear_staffingprofile_forecast_assignment
```

**Special Command Example:**

```
ksc_clear_staffingprofile_forecast_assignment STAFFING_PROFILE_ID="<staffing_
profile_id>" CLEAR_FROM_DATE="<clear_from_date>"
```



## Chapter 4: Using Tokens

### 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 the following types of 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

Many standard tokens are available to provide 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**

Validation : RSC - Parent Resource Pool

Name: RSC - Parent Resource Pool Reference Code: \_RSC\_PARENT\_RESOURCE\_POOL

Description: RSC - Parent Resource Pool

Enabled: ☒ Use in Workflow? ☐

Component Type: Auto Complete List

Validated By: SQL - Custom Expected list length: ☐ Short ☒ Long

Selection mode: ☒ Starts With ☐ Contains Number of results per page: 50

Enable Hierarchical Selection? ☐

Configuration | Filter Fields | Filter Layout | Hierarchical Display

Column Headers:

Seq	Column Header	Display...	Co
1	Resource Pool ID	N	
2	Resource Pool Name	Y	

SQL:

```
SELECT RP.resource_pool_id, RP.resource_pool_name
FROM rsc_resource_pools RP
WHERE upper(RP.resource_pool_name) like
UPPER('???')
AND RP.enabled_flag = 'Y'
and (RP.resource_pool_name like
upper(substr('?',1,1)) || '%'
or RP.resource_pool_name like lower(substr('?',1,1)) || '%')
```

Tokens Use Bind Variables? ☒

Used By Ownership OK Save Cancel

Ready (Read-Only, Seed Data)

## 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 are ignored. For example, if the token name is misspelled or no context is available.

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, 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:

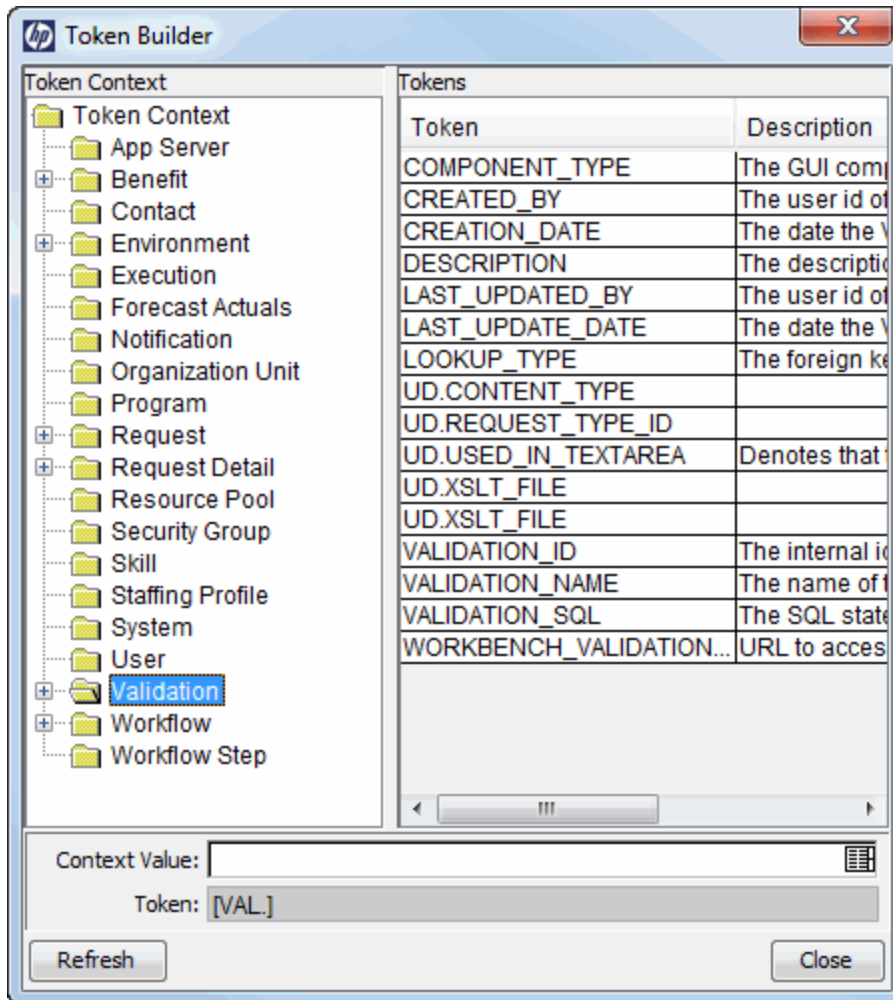
```
[USR="[USR="[REQ.CREATED_BY_NAME]".VUD.MANAGER]".EMAIL_ADDRESS]
```

The token evaluation engine first evaluates the innermost token ([REQ.CREATED\_BY\_NAME]), then the next token ([USR="<name>".VUD.MANAGER]), and finally 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 49](#), you can open the Token Builder window (["Figure 4-2. Token Builder window" on the next page](#)) 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.

For information on using the token builder, see ["Using the Token Builder" on page 65](#).

# 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" on page 55](#)
- ["Explicit Entity Format" on page 56](#)
- ["User Data Format" on page 58](#)
- ["Parameter Format" on page 59](#)
- ["Sub-Entity Format" on page 62](#)
- ["Environment and Environment Application Tokens" on page 63](#)

["Table 4-1. Entities" below](#) lists the entities and the formats that each entity supports.

**Table 4-1. Entities**

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	Y	N

**Table 4-1. Entities, continued**

<b>Prefix (Entity)</b>	<b>Entity and Description</b>	<b>User Data Format</b>	<b>Parameter Format</b>
	use app code values, even if they are null.		
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
PRJ	Project	Y	N
PRJD	Project details	N	Y
REL	Release	N	N
REL.DIST	Release (distribution)	Y	N
REQ	Request	Y	Y
REQ.FIELDS	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

**Table 4-1. Entities, continued**

<b>Prefix (Entity)</b>	<b>Entity and Description</b>	<b>User Data Format</b>	<b>Parameter Format</b>
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
WF	Workflow	Y	N
WF.WFS	Workflow (step). Use this format to specify a specific workflow.	N	Y
WFS	Workflow step	Y	N
WST	Step txn (transaction)	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 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 50](#).

## 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 do one of the following:

- Type in the field to reduce the list
- 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, to reference the email address for jsmith, the token would be:

```
[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 51](#).

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. Because 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.

**Note:** For a list of all explicit entity format tokens, see ["Tokens" on page 128](#).

## 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 is built using the following steps:

1. Build the Description token for the workflow.
2. Copy and paste the Description token into another field.
3. Build the Workflow Name token for the package.
4. Copy and paste the Workflow Name token within the Description token.

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

**Caution:** PPM Center does not support nesting explicit entity tokens within other tokens in the following cases:

- field level security for request header types and request types
- workflow security
- special commands for workflows

## User Data Format

User data fields use the following tokens format:

[<Prefix>.<DataType>.<Token>]

where

<Prefix>	Represents the name of the entity that has user data. For values, see <a href="#">"Table 4-1. Entities" on page 53.</a>
<DataType>	Represents the type of user data being referenced.  Specify: <ul style="list-style-type: none"><li>• UD for user data</li><li>• VUD for visible user data</li></ul>
<Token>	Represents 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]

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.

**Note:** 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 following parameter format for tokens:

[<Prefix>.<DataType>.<Token>]

where

<i>Prefix</i>	Represents the name of the entity that uses a custom field. For values, see <a href="#">"Table 4-1. Entities" on page 53</a> .
<DataType>	Represents the type of user data being referenced. Specify: <ul style="list-style-type: none"><li>• P for an entity that is referenced and hidden</li><li>• VP for an entity that is visible</li></ul>
<Token>	Represents the name of the token for the specific parameter field.

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

[PKGL.P.GAP\_NUMBER]

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.

**Caution:** Do not use words "APPLICATION" and "APPLICATION\_CODE" as token for custom fields in any request type. PPM Center reserves these two words for internal use. Therefore, you may experience issues if you use either of the words as token for any custom field.

**Note:** 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. Table component formats" on the next page](#) 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**

Hardware Information					
<div> <div>View</div> <div> <div>+</div> Add Row </div> <div> <div>✕</div> Delete Rows </div> <div> <div>Copy</div> <div>Paste</div> </div> <div> <div>↑</div> Move Up </div> <div> <div>↓</div> Move Down </div> </div>					
Seq	Products	Quantity	Price	Total	
1	PC	3	1200	3600	
2	PC	2	1200	2400	
<div> <div>Export to Excel</div> <div> <div>⏪</div> <div>⏩</div> <div>Page 1 of 1</div> <div>⏪</div> <div>⏩</div> </div> <div>Show 5 ▾ Each Page</div> </div>					

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

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 Entries".

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. In the rare case that you need to, you can reference specific sub-entities using the explicit entity syntax

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>.<Token>]
```

where

<Prefix>	Represents the name of the entity. For values, see <a href="#">"Table 4-1. Entities" on page 53</a> .
<Sub-entity>	Represents the name of the sub-entity. For values, see <a href="#">"Table 4-1. Entities" on page 53</a> .
<Token>	Represents the name of the token for the specific field.

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]
```

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.

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 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 following example shows token prefixes SOURCE\_ENV and DEST\_ENV used to reference the current source and destination:

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

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

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. Sample environment and application attributes" below](#) shows other attributes of the environments and applications.

**Table 4-2. Sample environment and application attributes**

Environment	App Code	Server Base Paths
QA		/qa



**Table 4-2. Sample environment and application attributes, continued**

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

"Table 4-3. Sample environment tokens" below 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

**Note:** 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 Extensions.

## 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. For more information, see ["About the Token Builder" on page 51](#). 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 **Open > 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.

# Chapter 5: Using Validations

## 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 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 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 requirements, you can use the Validation Workbench to create your own validation. For details, see ["Configuring Validations" on page 75](#).

**Caution:** PPM Center does not support deprecated validations. The user-defined fields and workflow steps that are created by using these validations may not work.



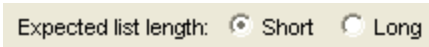

## Validations and Special Characters

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

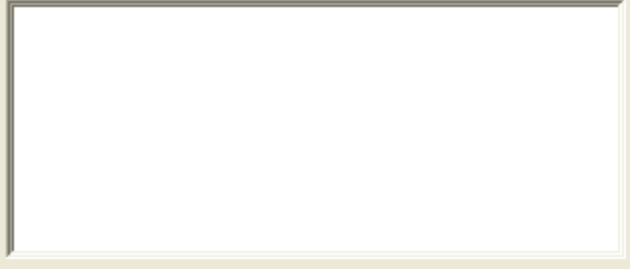




## Validation Component Types

You can only use certain component types in a workflow step source validation. ["Table 5-1. Component types" on the next page](#) summarizes the field component types available.

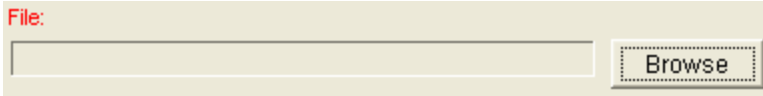

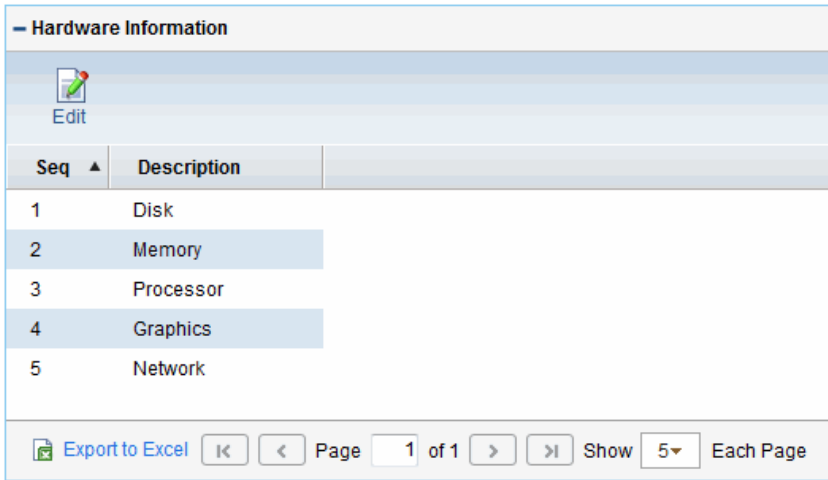
**Table 5-1. Component types**

Component Type	Use In Workflow?	Description, Example, and Configuration instructions
Text field	Yes	<p>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.</p>  <p><a href="#">"Configuring Text Field Validations" on page 79.</a></p>
Drop-down list	Yes	<p>Field that displays a list of values.</p>  <p><a href="#">"Configuring Static List Validations" on page 83.</a></p> <p><a href="#">"Configuring Dynamic List Validations" on page 87.</a></p>
Radio buttons (Yes/No)	No	<p>Field that accepts binary input.</p>  <p>There is nothing to configure for this component type.</p>
Auto-complete list	Yes	<p>Field that lets you open a dialog box that lists choices.</p> <p><b>NOTE:</b> By default, an auto-complete dialog is opened without any filter and all possible results are displayed. If the set of results is large, it may take some time for the auto-complete dialog to open. You can enable the <code>AUTO_COMPLETE_LONG_TYPE_CULLTEXT_REQUIRED</code> parameter (by setting the parameter to true) to open the auto-complete dialog without any results. This allows the user to enter a filter to limit the results displayed. If no filter is desired, the user may select <b>Find</b> to list all results.</p>  <p><a href="#">"Configuring Static List Validations" on page 83.</a></p> <p><a href="#">"Configuring Dynamic List Validations" on page 87.</a></p>
Text area	No	Text entry field that can include multiple lines.

**Table 5-1. Component types, continued**

Component Type	Use In Workflow?	Description, Example, and Configuration instructions
		<p>Initial Version Comment:</p>  <p><a href="#">"Configuring Text Area Validations" on page 104.</a></p>
Date field	No	<p>Field that lets you specify date format.</p>  <p><a href="#">"Configuring Date Field Validations" on page 106.</a></p>
Web address (URL)	No	<p>Text entry field for specifying a URL. Clicking <b>U</b> opens a browser window to the specified Web address.</p>  <p><a href="#">"Accessing Validations Through Packages and Requests" on page 72.</a></p>
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.</p>  <p><a href="#">"Accessing Validations Through Packages and Requests" on page 72.</a></p> <p><a href="#">"Configuring File and Directory Chooser Validations" on page 108</a></p>
Directory chooser	No	<p>Used only in object types. Requires a parameter field defined with the token P_FILE_LOCATION.</p>  <p><a href="#">"Configuring File and Directory Chooser Validations" on page 108.</a></p>
Attachment	No	Field used to locate and attach files.

**Table 5-1. Component types, continued**

Component Type	Use In Workflow?	Description, Example, and Configuration instructions
		<p><b>File:</b></p>  <p><a href="#">"Accessing Validations Through Packages and Requests" on the next page</a></p>
Password field	No	<p>Field used to capture passwords.</p>  <p>There is nothing to configure for this component type.</p>
Table component	No	<p>Used to specify multiple records in 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.</p> <p>You can only add fields of this component type to the Request Details page.</p> <p><b>Limitation:</b> The maximum entry number allowed for fields of this component type is 500.</p>  <p><a href="#">"Configuring the Table Component" on page 109</a></p>
Staffing profile, financial data table,	No	<p>Field that you can add to the request type to enable access to view, edit or create staffing profiles or financial data tables associated with a request, project, or work plan.</p>

**Table 5-1. Component types, continued**

Component Type	Use In Workflow?	Description, Example, and Configuration instructions
link		<p><b>Financial Data Table:</b>      <a href="#">Upgrade Servers</a></p> <p><a href="#">"Accessing Validations Through Packages and Requests" below</a></p>

**Special Note for Link and Web Address (URL) Component Types**

When you use `<!--HTML-->` for **Link** and **Web Address (URL)** component types, make sure not to use `<a></a>` tags as they are not allowed. **Link** and **Web Address (URL)** components are `<a></a>` tags in Web UI, you cannot embed `<a></a>` tags within `<a></a>` tags.

`<!--HTML-->` is designed for advanced users and you should use tags that affects font and color only. For example, the script below is valid:

```
<!--HTML--><font color='red'>PPM</font>
```

```
<!--HTML--><b>PPM</b>
```

The script below is not allowed for Link and URL:

```
<!--HTML--><a href='http://hp.com'>HP</a>
```

Note that link description is also not allowed.

## Accessing Validations Through Packages and Requests

You can access the package and request group validations directly. 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.

**Note:** 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 package and request groups validation window from the Package Workbench:

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



The PPM Workbench opens.

- From the shortcut bar, select **Deployment Mgmt > Packages**.
- From the menu, select **Package > New Package Group**.

**Validation : PPM - Package and Request Groups**

Name: PPM - Package and Request Groups    Reference Code: ACKAGE\_AND\_REQUEST\_GROUPS

Description: groupings for packages and requests

Enabled: ☒    Use in Workflow? ☐

Component Type: Drop Down List

Validated By: List

**Validation Values:**

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

New   Edit   Delete   Copy From   ↑ ↓

Used By   Ownership   OK   Save   Cancel

Ready (Read-Only, Seed Data)

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

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

The PPM Workbench opens.

- From the shortcut bar, select **Demand Mgmt > Request Types**.
- From the menu bar, select **Request Type > Request Type Category Setup**.

The Validation window opens and lists the existing request type categories.

**Validation : CRT - Request Type Category**

Name: CRT - Request Type Category      Reference Code: \_CRT\_REQUEST\_TYPE\_CATEGORY

Description: This validation contains a list of categories used for organizing Request Types

Enabled: ☒      Use in Workflow? ☐

Component Type: Drop Down List

Validated By: List

**Validation Values:**

Seq	Code	Meaning	Description	Enab...	Default
1	MISCELLANEOUS	Miscellaneous		Y	N

New   Edit   Delete   Copy From   ↑ ↓

Used By   Ownership   OK   Save   Cancel

Ready (Read-Only, Seed Data)

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

## Viewing System Validations

PPM Center comes with several pre-configured validations.

**Note:** Some of these validations may have been altered to 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 **Create > 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*.

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.

- 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:

- workflow step sources
- entity fields.

Although a disabled validation continues to function in existing workflow steps and fields, you cannot use it to define a new step source or field.

**Note:** 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 validation:

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

The PPM Workbench opens.

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

The Validations Workbench opens.

4. Click **New Validation** or open an existing validation.

**Note:** If you are opening an existing validation, your PPM Center instance supports multiple languages, and the validation is defined in a language other than your session language, you cannot edit the validation. For more information, see the *Multilingual User Interface Guide*.

The Validation window opens.

**Validation : CRT - Request Header Types - Enabled**

Name: CRT - Request Header Types - Enabled Reference Code: CRT\_REQUEST\_HEADER\_TYPES\_ENABLED

Description: CRT - Request Header Types - Enabled

Enabled: ☒ Use in Workflow? ☐

Component Type: Auto Complete List

Validated By: SQL - Custom Expected list length: ☒ Short ☐ Long

Selection mode: ☒ Starts With ☐ Contains Number of results per page: 50

Enable Hierarchical Selection? ☐

Configuration | Filter Fields | Filter Layout | Hierarchical Display

Seq	Column Header	Display...
1	Request Header Type ID	N
2	Request Header Type	Y
3	Header Set Context ID	N

SQL: select REQUEST\_HEADER\_TYPE\_ID, REQUEST\_HEADER\_TYPE\_NAME, parameter\_set\_context\_id from KCRT\_REQUEST\_HEADER\_TYPES\_V where ENABLED\_FLAG = 'Y' and UPPER(REQUEST\_HEADER\_TYPE\_NAME) like UPPER('?%') and (REQUEST\_HEADER\_TYPE\_NAME like upper(substr('?', 1, 1)) || '%' or REQUEST\_HEADER\_TYPE\_NAME like lower(substr('?', 1, 1)) || '%')

Buttons: New, Edit, Delete, Tokens, Use Bind Variables? ☐

Buttons: Used By, Ownership, OK, Save, Cancel

Ready (Read-Only, Seed Data)

5. Provide the information described in the following table.

Field Name	Description
<b>Name</b>	Name of the new validation.
<b>Reference Code</b>	<p>Unique name to identify the validation across all languages being used in your PPM Center implementation, regardless of whether its name is changed or translated.</p> <p>Either accept the default value or type a new value.</p> <p>The reference code value must be unique across all languages, use capital letters and ASCII characters, not start with an underscore ( _ ), and not use any of the following special characters:</p> <p>~!@#\$\$%^&amp;*() + } { " : ? &gt; &lt; ` - = ] [ ' ; / , ' ,</p> <p>System data reference codes start with an underscore ( _ ) and should not be modified.</p>

<b>Description</b>	Brief description of the validation.
<b>Enabled</b>	Select this checkbox to enable the validation.
<b>Use in Workflow</b>	Select this checkbox to use the validation in a workflow step source. For a list of components that can be used in workflow steps, see <a href="#">"Table 5-1. Component types" on page 69.</a>
<b>Component Type</b>	<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>For a list of components types that can be used, see <a href="#">"Table 5-1. Component types" on page 69.</a></p>

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

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

**Note:** If a validation value contains a lookup code, once you have specified and saved the code, you can no longer edit the code. If you need to make a change to the code, you must delete the existing code and add a new code, recreating any information from the deleted code.

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

- a. From the shortcut bar, select **Sys Admin > Security Groups**.
- b. Select a user.
- c. Click the **Ownership** tab.

The Ownership window opens.

- d. Select **Only groups listing below that have the Edit Validations Access Grant**.
- e. Click **Add**.

The Add Security Group window opens.

- f. Add security groups.

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

**Note:** If you are configuring a new validation and your PPM Center instance supports multiple languages, any validation you configure is defined in the language you selected at logon (your session language). After the validation is configured, it can be modified only in its definition language. For more information, see the *Multilingual User Interface Guide*.

## Configuring Text Field Validations

Text fields are 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. Open a new or existing validation; see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **Text Field**.
3. In the **Data Mask** field, select one of the following data masks:
  - Alphanumeric
  - Alphanumeric Uppercase
  - Numeric
  - Currency
  - Percentage
  - Telephone
  - Custom

**Figure 5-1. Validation window for the currency data mask**

4. Optionally, configure the data mask. Depending on your data mask type, the field you see varies. Use the following table to help you determine how to configure your data mask:

Data Mask Type	Field Name	Description
Alphanumeric and Alphanumeric Uppercase	<b>Max Length</b>	Specify the maximum field length for fields using this validation.
Currency	<b>Region</b>	<p>Determines the currency symbol displayed in the field and the position of the text in the field.</p> <p>The default settings used depends on the locale setting for the machine. You can set the currency display settings by using the Regional Settings tab in the Edit My Profile dialog:</p> <ul style="list-style-type: none"> <li>select <b>Open &gt; Administration &gt; Edit My Profile</b> and select the <b>Regional Settings</b> tab</li> </ul>
Currency, Numeric, and Percentage	<b>Maximum Value</b>	Largest value allowed for this field. You can specify a positive or negative number.



<b>Data Mask Type</b>	<b>Field Name</b>	<b>Description</b>
Currency, Numeric, and Percentage	<b>Minimum Value</b>	Smallest accepted value for the field. You can specify positive or negative number.
Currency, Numeric, and Percentage	<b>If Data not Entered, then display a zero</b>	Determines whether a field with no data displays a zero.
Currency, Numeric, and Percentage	<b>Use Group Separator</b>	<p>Determines if the field uses a group separator (such as a comma) to divide characters within large numbers. For example, display 1000000 as 1,000,000.</p> <p>The default character used as the separator depends on the regional settings defined by each user.</p>
Currency, Numeric, and Percentage	<b>Negative Number looks like</b>	<p>Determines the text used to display negative numbers. Options are:</p> <ul style="list-style-type: none"> <li>◦ (1000)—parentheses and black text</li> <li>◦ (1000)—parentheses and red text</li> <li>◦ -1000—minus character (-) and black text</li> <li>◦ -1000—minus character (-) and red text</li> </ul>
Currency, Numeric, and Percentage	<b>Number of Decimal Places</b>	The maximum number of decimal places displayed.
Telephone	<b>Format</b>	<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"> <li>◦ Parentheses ( )</li> <li>◦ Period (.)</li> <li>◦ Dash (-)</li> <li>◦ Space</li> <li>◦ Plus character (+)</li> </ul> <p>For telephone format examples, see <a href="#">"Sample Telephone"</a></p>

Data Mask Type	Field Name	Description
		<a href="#">Data Mask Formats" on page 83.</a>
Telephone	<b>Maximum # of Digits</b>	The maximum number of digits that the field accepts.
Telephone	<b>Minimum # of Digits</b>	The minimum number of digits that the field accepts.
Custom	<b>Format</b>	<p>Type a combination of the following symbols.</p> <ul style="list-style-type: none"> <li>◦ Use D to specify that the user must provide a numeric value between 0 and 9.</li> <li>◦ Use L to specify that the user must provide an alphabetic character between A and Z.</li> <li>◦ Use A to specify that the user must type a character or space.</li> <li>◦ Use a \ (backslash) to specify that the next character is to be displayed as the literal character. For example: "\A" is displayed as "A".</li> </ul> <p>For custom format examples, see <a href="#">"Sample Custom Data Mask" on the next page.</a></p>

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

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

The Formatted Output window displays the results.

6. Click **OK**.

**Note:** If your PPM Center instance supports multiple languages, any validation you configure is defined in the language you selected at logon (your session language). After the validation is configured, it can be modified only in its definition language. For more information, see the *Multilingual User Interface Guide*.

## Sample Telephone Data Mask Formats

Use the telephone data masks to specify telephone number display.

**Table 5-2. 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

If you define a format that lets users specify a range of number of characters, any extra characters specified are always grouped with the first set of characters.

For example, if you configure the telephone data mask with a minimum of 10 characters and a maximum of 15 characters, the results are as follows:

**Table 5-3. Sample extra character telephone data mask formats**

Format Rule	User Input	Output
DDD-DDD-DDDD	1234567890	123-456-7890
DDD-DDD-DDDD	12345678901	1234-567-8901

## Sample Custom Data Mask

You can customize the field to accept numeric values, alphabetic characters, spaces, and custom delimiters.

**Table 5-4. Sample custom data mask formats**

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

## 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.

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. HP recommends validating using an SQL query or PL/SQL function to obtain the values from a database table.

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.

To create a static list validation:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. 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.

3. In the **Validated By** field, select **List**.
4. For **Selection mode**, select one of the following:

Selection	Description
<b>Starts with</b>	Type characters and then type the <b>Tab</b> key. The selection window opens and lists entries that begin with the specified characters.
<b>Contains</b>	Type characters, and then type the <b>Tab</b> 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.

5. Click **New**.


The Add Validation Window opens.

The screenshot shows a Windows-style dialog box titled "Add Validation Value" with an HP logo. It has two tabs: "Value Information" and "User Data". The "Value Information" tab is active, showing four text input fields labeled "Code:", "Meaning:", "Desc:", and "Enable?". The "Enable?" checkbox is checked. There is also a "Default:" checkbox which is unchecked. At the bottom right are "OK", "Add", and "Cancel" buttons. A status bar at the bottom left says "Ready".

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


Field Name	Description
<b>Code</b>	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. For example, 123.
<b>Meaning</b>	Displayed meaning for the validation value in the drop-down list or auto-complete. For example, western region.
<b>Desc</b>	More complete description of the value specified in the Meaning field.
<b>Enable?</b>	Select to ensure that the value is displayed in the list of values.
<b>Default</b>	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.

7. To add more values and keep the Add Validation Value window open, click **Add**.
8. To save your changes and close the window, click **OK**.

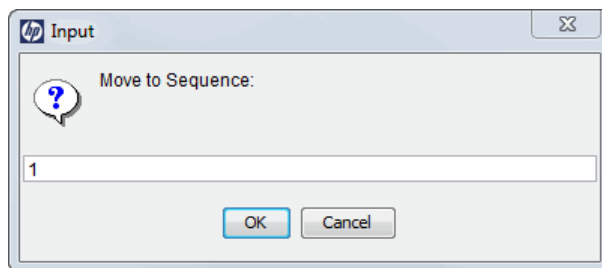
9. To change the sequence of the validation values in the **Validation Values** section,
  - To change the order in which validation values are listed, use the up and down arrow buttons.
  - To sort the validation values alphabetically according to the **Code** column, click the  button.

**Note:** This alphabetical sorting is case-insensitive.

All validation codes are sorted by letter, so "value3" is listed after "value12". To avoid this, HP recommends that you prefix numbers with some zeros. For example, use "value003" and "value012" instead.

- To specify the sequence of a specific validation value, select the desired value, and then click the  button. In the input box that pops up, enter the sequence number of the row that you want to move the selected value to.

For example, if you enter 1 for the selected value and click **OK**, the selected value is moved from its original place to the first.



The sequence of the validation values determines the order that the values are displayed in the list.

10. To copy existing values defined in other validations, 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.
11. To delete list validations, select the row you want to delete and click **Delete**.
12. Click **Save**.

## Configuring Dynamic List Validations

You can create validations that provide a dynamic list to the user. HP recommends using dynamic list validations. 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 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.

## SQL Validation Tips

The following information can be helpful when 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 constrain the list, 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.

## Auto-Complete Matching Tips

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.

Consider the following tips as you configure the "starts with" or "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 user 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. Typing "Financial," to find potential separate request type used for each type of



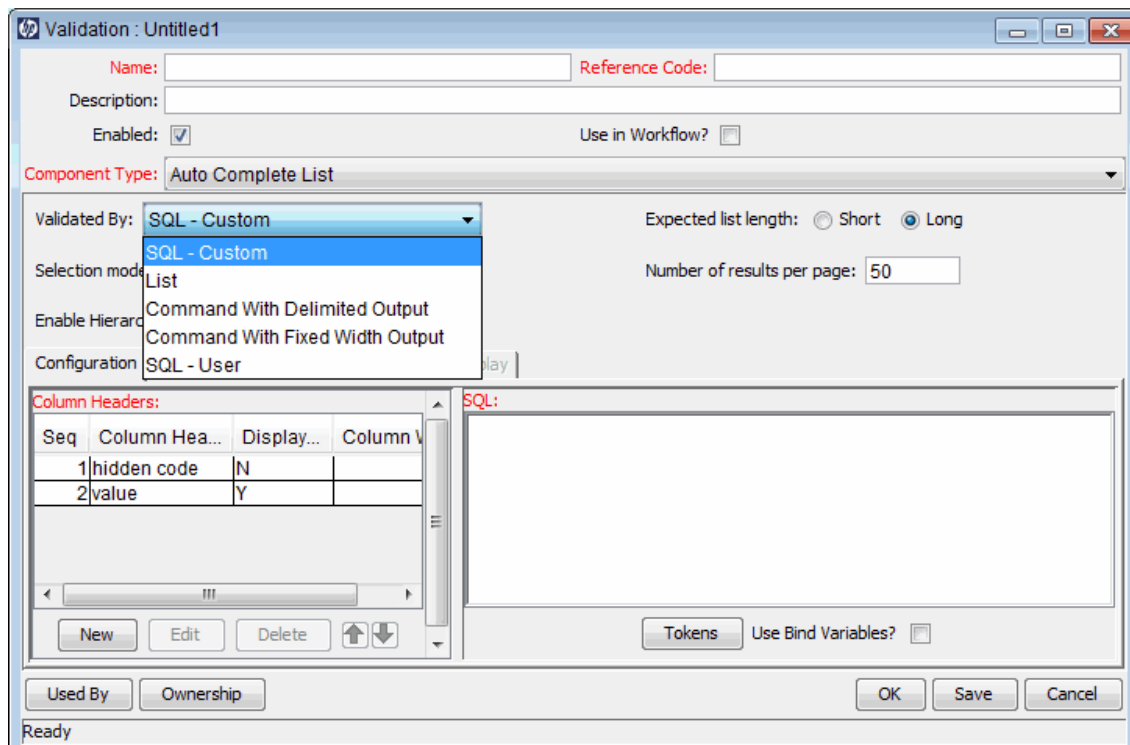
supported application would have returned the entire list. And, the user would be forced to try another "starts with" phrase or read a long list.

## Auto-Complete Values

The values in an auto-complete **Validate By** field, are as follows:

- **List.** Used to provide 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 5-2. Auto-complete list**



## 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.

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 74](#).

### SQL Validated Drop Down Lists

To define a dynamic list for a drop down list:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **Drop Down List**.

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

3. In the **Validated By** field, select **SQL**.
4. In the **SQL** field, provide the Select statement that queries the necessary database information. An ending semicolon is not necessary.
5. To add token to your SQL statement, click **Tokens**, select the token text that you want, copy the text, close the Tokens window, and paste the text into the SQL statement where it is needed.

### SQL Validated Auto-Complete Lists

Custom auto-completes or validations (Validated by: SQL-Custom) can be configured for short or long list formats.

User auto-completes or validations (Validated by: SQL-User) have the following 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.

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **Auto Complete List**.

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

3. In the **Validated By** field, select **SQL - Custom** or **SQL - User**.
4. For SQL - Custom, **Selection mode**, select one of the following:

Selection	Description
<b>Starts with</b>	Type characters and if the list is not automatically filtered, then type the <b>Tab</b> key. The selection window opens and lists entries that begin with the specified characters.
<b>Contains</b>	Type characters, and if the list is not automatically filtered, then type the <b>Tab</b> 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.

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.

5. For SQL - Custom, in the **Expected list length** field, select one of the following:

Selection	Description
<b>Short</b>	Auto-completes configured as short lists load all values when the window is opened. This can adversely affect performance.
<b>Long</b>	Auto-completes configured as long lists load a limited set of values when the window is opened. 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.

6. If you selected a long list, in the **Number of results per page**, indicate the number of results you want displayed on each page.

- On the **Configuration** tab, in the **SQL** field, provide the Select statement that queries the necessary database information. An ending semicolon is not necessary.

Validation: CRT - Request Header Types - Enabled

Name: CRT - Request Header Types - Enabled Reference Code: CRT\_REQUEST\_HEADER\_TYPES\_ENABLED

Description: CRT - Request Header Types - Enabled

Enabled: ☒ Use in Workflow? ☐

Component Type: Auto Complete List

Validated By: SQL - Custom Expected list length: ☒ Short ☐ Long

Selection mode: ☒ Starts With ☐ Contains Number of results per page: 50

Enable Hierarchical Selection? ☐

Configuration | Filter Fields | Filter Layout | Hierarchical Display

Column Headers:

Seq	Column Header	Display...
1	Request Header Type ID	N
2	Request Header Type	Y
3	Header Set Context ID	N

SQL:

```
select REQUEST_HEADER_TYPE_ID,
REQUEST_HEADER_TYPE_NAME, parameter_set_context_id from
KCRT_REQUEST_HEADER_TYPES_V where ENABLED_FLAG = 'Y' and
UPPER(REQUEST_HEADER_TYPE_NAME) like UPPER('?%') and
(REQUEST_HEADER_TYPE_NAME like upper(substr('?',1,1)) || '%' or
REQUEST_HEADER_TYPE_NAME like lower(substr('?',1,1)) || '%')
```

Tokens ☒ Use Bind Variables? ☐

Used By Ownership OK Save Cancel

Ready (Read-Only, Seed Data)

For example, XYZ Corporation creates an auto-complete field that lists all users in the Engineering department and is validated 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. 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.

**Note:** If you use the same token in both the WHERE clause of the SQL statement and the SQL statement of the validation's filter field, PPM Center may return no value for the ACL field when you change the value of the field associated with the token in PPM Center Web pages.

This is because the value of the token used in the WHERE clause of the ACL validation's SQL statement is retrieved from the Web pages, while the value of the token used in the SQL statement of the validation's filter field is retrieved from PPM Center database.

To avoid this issue, HP recommends that you not use the same token in the WHERE clause of an ACL validation's SQL statement and the SQL statement of the validation's filter field.

8. 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)) || '%')
```

9. 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)) || '%')
```

10. Under the **Column Headers** table, click **New**.

11. Specify values for the following:

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

12. For short lists, you are done, click **Save**.
13. For long lists and for SQL - User, you can continue with ["Adding Search Fields to Auto-Complete Validations" on the next pages](#).

## Adding Search Fields to Auto-Complete Validations

Auto-completes 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 provide values in the filter fields, and then click **Find** to display only the values that match the search criteria. The following shows the Select window with additional filter fields:

**Figure 5-3. Filter fields in the auto-complete select window**

The screenshot shows a window titled "Click a value to select" with a close button (X) in the top right corner. Below the title bar, there are four filter fields: "Assigned To:" (text input), "Department:" (dropdown menu), "First Name:" (text input), and "Last Name:" (text input). A "Find" button is located to the right of the "Last Name:" field. Below the filter fields is a table with four columns: "Full Name", "Username", "Department", and "Email". The table contains 15 rows of user data. At the bottom of the window, there is a pagination bar showing "Page: 1 2 3 4 5" and "Showing 1-50 of 378".

Full Name	Username	Department	Email
APM Administrator	aadministrator		
APM Analyst	aanalyst		ppm@ppmdemo.com
APM User	auser		ppm@ppmdemo.com
Abe Vernon	avernon	Corporate	ppm@ppmdemo.com
Admin Backup	admin2		ppm@ppmdemo.com
Admin User	admin		ppm@ppmdemo.com
Alex Jones	Alex		
Alex Richter	arichter	IT Finance	ppm@ppmdemo.com
Alex Smith	alex_qc	IT Quality Assurance	ppm@ppmdemo.com
Alexander Schneider	aschneider	IT Software Development	ppm@ppmdemo.com
Alice Jones	alice_qc	IT Quality Assurance	ppm@ppmdemo.com
Allen Hughes	ahughes	IT Software Development	ppm@ppmdemo.com

**Note:** Filter fields offer a powerful way to efficiently locate specific values. 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.

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

1. Click the **Filter Fields** tab.
2. Click **New**.

The Field: New window opens.

3. Provide the required and any optional information for the following:

Field Name	Description
<b>Field Prompt</b>	Name displayed for the field in the auto-complete Select window.
<b>Product</b>	PPM Center product that uses the field.

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



Field Name	Description
View Full Query	Opens a window that displays the full query.

4. Click **OK**.

You are returned to the Validation window.

5. Continue adding filter fields.
6. When you are done, click **Save**.
7. Optionally continue with ["Configuring the Filter Field Layout" below](#).

## Configuring the Filter Field Layout

To modify the filter field layout:

1. Follow the instructions in ["Configuring SQL Validations" on page 90](#) and ["Adding Search Fields to Auto-Complete Validations" on page 94](#).
2. 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 selected value.

Validation: CRT - Assigned To - Enabled

Name: CRT - Assigned To - Enabled Reference Code: \_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

Enable Hierarchical Selection? ☐

Configuration | Filter Fields | Filter Layout | Hierarchical Display

<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	<input type="checkbox"/> Company

Field Width: 1 Component Lines: 50 Move Field:     ☐ Swap Mode

Preview

Used By Ownership OK Save Cancel

Ready (Read-Only, Seed Data)

3. Select the dimensions of the field.

Fields can have a width of 1, 2, or 3. The field width must correspond to the column location. For example, a field located in Column 2 cannot have a width set to 3. For fields of the Text Area component type, you can determine the number of lines the Text Area will display. Select the field and change the value in the Component Lines field. If the selected field is not of type Text Area, this attribute is blank and non-updateable.

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

To select more than one field, type the Shift key while selecting a range to select a continuous set of fields. Type the Ctrl key to select a non-contiguous set of fields.

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

**Note:** A field or a set of fields cannot be moved to an area where other fields already exist.

The other fields 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.

**Note:** 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 Validations by Commands

Validating by commands with delimited or fixed-width output can be used to get data from an alternate source, and that data can be used 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 with these alternate data sources, in the context of a validation, to provide a list of values.

**Figure 5-4. Validation by command with delimited output**

To configure a validation by command with delimited or fixed-width output:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **Auto Complete List**.

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

3. In the **Validated By** field, select **Command With Delimited Output** or **Command With Fixed-Width Output**.
4. For **Selection mode**, select one of the following:

Selection	Description
<b>Starts with</b>	Type characters and if the list is not automatically filtered, then type the <b>Tab</b> key. The selection window opens and lists entries that begin with the specified characters.
<b>Contains</b>	Type characters, and if the list is not automatically filtered, then type the <b>Tab</b>

Selection	Description
	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.

5. In the **Expected list length** field, select one of the following:

Selection	Description
<b>Short</b>	Auto-completes configured as short lists load all values when the window is opened. This can adversely affect performance.
<b>Long</b>	Auto-completes configured as long lists only load a limited set of values when the window is opened. 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.

6. If you selected a long list, in the **Number of results per page**, indicate the number of results you want displayed on each page.
7. Click **New Command**.

The New Command window opens.

8. Complete the fields described in the following table.

Field Name	Description
<b>Command</b>	Command name.
<b>Condition</b>	Specific conditions under which the command steps are to be executed. This step is optional. For more information, see <a href="#">"Command Conditions" on page 13</a> .
<b>Description</b>	Command description. This step is optional. For more information, see <a href="#">"Command Conditions" on page 13</a> .
<b>Timeout(s)</b>	Length of time (in minutes) to run the command before stopping. This setting is useful if a command hangs or takes too long to execute.
<b>Enabled</b>	Use the <b>Yes</b> and <b>No</b> option buttons to enable and disable the command.

Field Name	Description
<b>Steps</b>	<p>Provide at least one command step.</p> <p>These can include PPM Center special commands. Include the special command <code>ksc_capture_output</code>, which captures and parses the delimited command output. If you place the <code>ksc_capture_output</code> special command between the <code>ksc_connect</code> and <code>ksc_disconnect</code> commands, the command is run on the remote system. Otherwise, the command is run locally on the PPM Server (like <code>ksc_local_exec</code>).</p>

- Click **Tokens** to open the Token Builder window and find a token to add to the command step. For information about tokens, see ["Using Tokens" on page 49](#).
- Click **Special Cmd** to open the Special Command Builder and find a special command to add to a command step. For information about special commands, see ["Using Special Commands" on page 21](#).
- To show or hide a **Descriptions** field in the **Steps** field, click **Show Desc** or **Hide Desc**.

9. Click **OK**, **Add** or **Cancel**.

10. For delimited data, in the **Data Delimiter** field, indicate 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. If you do not define a column header for each column in a command, a default header is used.

11. Under the **Column Headers** table, click **New**.

12. Specify values for the following:

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

13. Click **Save**.

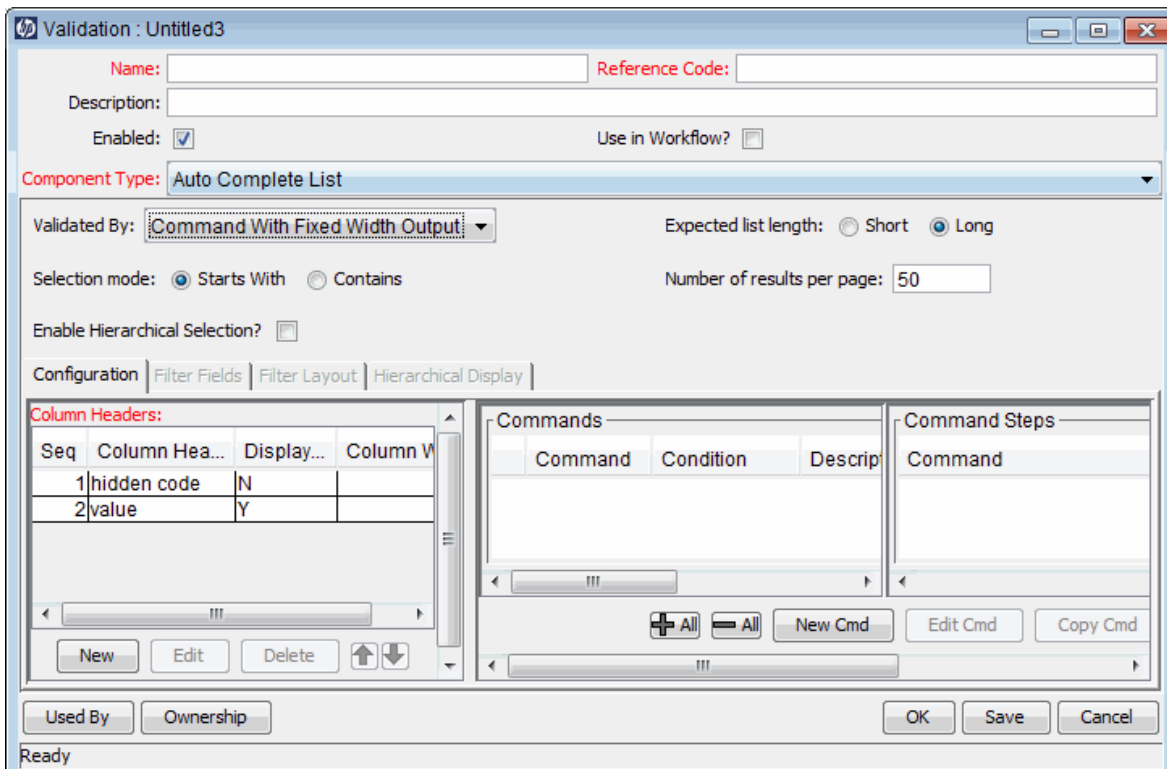
The following example uses a comma as the delimiter and includes the validation values red, blue, and green. The script places the validations into the `newfile.txt` file, and then uses the special command `ksc_capture_output` to process the text in the file.

```
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
```

The following example includes the validations red, blue, and green. The column width of the red, green, and blue columns 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 5-5. Validation by command with fixed-width output**



# Configuring Text Area Validations

Text areas vary in size. You can create a text field or area of length 40, 200, 1800, or 4000. Text fields can be configured to display the data according to a certain format.

To create a text area validation:

- 1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
- 2. In the **Component Type** field, select **Text Area**.
- 3. In the **Data Mask** field, select and optionally configure one of the following data masks:

Data Mask	Description
Alphanumeric	Field allows all alphanumeric characters.
Alphanumeric Uppercase	Field allows alphanumeric characters and formats all characters as uppercase text.
Numeric	Field allows numeric values.



4. For the Numeric data mask, configure the following;

Field Name	Description
<b>Maximum Value</b>	Largest value allowed for this field. You can specify a positive or negative number.
<b>Minimum Value</b>	Smallest accepted value for the field. You can specify positive or negative number.
<b>If Data not Entered, then display a zero</b>	Determines whether a field with no data displays a zero.
<b>Use Group Separator</b>	<p>Determines if the field uses a group separator (such as a comma) to divide characters within large numbers. For example, display 1000000 as 1,000,000.</p> <p>The default character used as the separator depends on the locale setting for the machine. You can set the delimiter by using the Regional Settings tab in the Edit My Profile dialog:</p> <ul style="list-style-type: none"> <li>◦ select <b>Open &gt; Administration &gt; Edit My Profile</b> and select the <b>Regional Settings</b> tab</li> </ul>
<b>Negative Number looks like</b>	<p>Determines the text used to display negative numbers. Options are:</p> <ul style="list-style-type: none"> <li>◦ (1000)—parentheses and black text</li> <li>◦ (1000)—parentheses and red text</li> <li>◦ -1000—minus character (-) and black text</li> <li>◦ -1000—minus character (-) and red text</li> </ul>
<b>Number of Decimal Places</b>	The maximum number of decimal places displayed.

5. Click **Save**.

## 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. On the **Results** tab, select **Text Area - 1800**.
4. Click **Copy**.
5. Rename the validation.
6. Edit the validation if necessary.
7. Click **Save**.

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

## 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.

To create a date field validation:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **Date Field**.

## 3. Configure the date and time formats as necessary, using the following:

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

4. Click **Save**.

## Configuring File and Directory Chooser Validations

A **File Chooser** field and **Directory Chooser** field can be used by object types to select a valid file or directory from an environment. Deployment Management connects to the first source environment on a workflow and provides the ability to do one of the following:

- View all files within a specific directory and select one from the list.
- Navigate through the directory structure and the selection of a directory from the list.

File Chooser requires the following:

- A field defining the file location for the directory chooser, described ["Configuring File and Directory Chooser Validations" above](#).
- 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.

Directory Chooser has the following requirements:

- You can only use the **Directory Chooser** field on an object type.
- On every object type where 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.

To create a file or directory chooser validation:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. In the **Component Type** field, select **File Chooser or Directory Chooser**.
3. For file chooser validations, use the **Base File Name Only** checkbox to define whether the base file name only (without its suffix) or the complete name is displayed.

4. In the **Environment Override Behavior** field, select one of the following:

Selection	Description
<b>Default Behavior</b>	Used to select files from the default environment.
<b>Static Environment Override</b>	Lets you override one environment at a time.
<b>Token-based Environment Override</b>	Provides the ability to select a token that is to resolve to the overriding environment.

Depending on your selection the fields you see vary.

5. For static and token-based selections you need to specify values for the following fields as appropriate:

Field Name	For	Description
<b>Environment Token</b>	token-based	Select the token that is to resolve to the overriding environment.
<b>Overriding Environment</b>	static	Select the environment to be overridden.
<b>Overriding Client Basepath</b>	both	Specify a basepath to override the client basepath of the environment.
<b>Overriding Server Basepath</b>	both	Specify a basepath to override the server basepath of the environment.

6. Click **Save**.

## Configuring the Table Component

The table component is used to provide multiple records in 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 the following elements:

- Products
- Quantity
- Price
- Total

To collect this information, XYZ creates a table component field labeled **Hardware 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 5-6. Hardware information window**

Hardware Information					
View		Add Row		Copy	
		Delete Rows		Paste	
				Move Up	
				Move Down	
Seq	Products	Quantity	Price	Total	
1	PC	3	1200	3600	
2	PC	2	1200	2400	
<a href="#">Export to Excel</a> <span>⏪</span> <span>⏴</span> Page <input type="text" value="1"/> of 1 <span>⏵</span> <span>⏩</span> Show <input type="text" value="5"/> Each Page					

You can only add fields of this component to request types.

To create a table component field for the Request Details page:

1. Open a new or existing validation, see ["Configuring Validations" on page 75](#).
2. From the **Component Type** list, select **Table Component**.

The screenshot shows the 'Validation : Untitled1' dialog box. The 'Name' field is empty, and the 'Reference Code' field is also empty. The 'Description' field is empty. The 'Enabled' checkbox is checked, and the 'Use in Workflow?' checkbox is unchecked. The 'Component Type' dropdown menu is open, showing 'Table Component' selected. The 'User Instructions' field is empty. The 'Meta Layer View' dropdown is set to 'MREQ\_'. The 'Table Columns' tab is active, showing a table with columns: Column S..., Column Hea..., Column To..., Parameter Col., Enab..., Component T..., Validation, Editable, Requir..., and D. The 'New' button is highlighted.

3. Type a validation name and description.
4. Provide any user instructions to display at the top of the table entry page.
5. To create the table columns, on the **Table Columns** tab, click **New**.

The Field window opens.

Define the type of information to store in that column.

This may require that you create a validation for the column.

You cannot use file attachments in a table component column.

The screenshot shows the 'Field: New' dialog box. It contains the following elements:

- Column Header:** Text input field.
- Column Token:** Text input field.
- Description:** Text input field.
- Enabled:** Radio buttons for 'Yes' (selected) and 'No'.
- Validation:** Text input field with 'New' and 'Open' buttons.
- Component Type:** Dropdown menu set to 'None'.
- Multi-Select Enabled:** Radio buttons for 'Yes' and 'No' (selected).
- Editable:** Radio buttons for 'Yes' (selected) and 'No'.
- Display Total:** Radio buttons for 'Yes' and 'No'.
- Required:** Dropdown menu set to 'Never'.
- Buttons:** 'Copy From...', 'OK', 'Add', and 'Cancel'.
- Status Bar:** 'Ready'.

Element	Description
Column Header	Provide a name for the table column.
Column Token	<p>Provide the token syntax for the table column. For a list of all explicit entity format tokens, see <a href="#">"Tokens" on page 128</a>.</p> <p>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 <a href="#">"Using Tokens" on page 49</a>.</p>
Description	Provide a description of the data held in the column.
Enabled	Select Yes or No.



Element	Description
<b>Validation</b>	Click the auto-complete button to retrieve a list of validations.
<b>New</b>	Click to obtain the Validation window so you can define a new validation.
<b>Component Type</b>	Displays the component type associated with the validation.
<b>Multi-Select Enabled</b>	Button availability depends on validation and component type. Select Yes to allow user to multi-select values for the field.
<b>Attributes tab</b>	<p>Editable—Use to indicate if the field is editable or read-only.</p> <p>Display Total—Use to indicate if the total for the column is displayed at the bottom of the table or not.</p> <p>Required—Use to indicate if the field is required or optional.</p>
<b>Default tab</b>	<p>Default Type—Choose Constant, to provide a default value for the field.</p> <p>Visible Value—Specify the default value for the field.</p>
<b>Storage tab</b>	<p>Max Length—Select the maximum length of the column.</p> <p>Parameter Col—Select a parameter from the list of values.</p>
<b>Copy From</b>	Use to query for and copy an existing column into this table.
<b>OK</b>	Use to close the Field window and save your changes after you finish adding columns.
<b>Add</b>	Use to save the column information and add another column.
<b>Cancel</b>	Use to close the window without saving changes.

6. When you are done, click **Save**.

**Note:** Starting from PPM Center version 9.12, a new table component is implemented to improve usability. You can switch back to the original table component layout by setting the server configuration parameter `TABLE_COMP_USE_LEGACY` to `true`. The default value is `false`.

## Configuring Rules

To set up rules for advanced defaulting behavior or calculating column totals, configure any required table logic, as follows:

You can configure table rules in the same way you configure advanced request type rules. That is, 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 have enormous flexibility for the data that populates the table cells.

1. Follow the instructions in ["Configuring the Table Component" on page 109](#).
2. Click the **Rules** tab.
3. Click **New**.

**Rules Window**

Rule Name: Set Unit Price

Description:

Enabled? ☒ Yes ☐ No

Rule Event: Apply on field change

Process subsequent rules? ☒ Yes ☐ No

*(If an event triggers multiple rules, they are processed in sequential order. Check 'No' above if you do not want subsequent rules to be processed after this rule completes.)*

**Dependencies**

Column Header	Condition
Products	contains any value

New Edit Remove

**Results**

Result Fields:

Column Header	Column	Token
Price	1	TE.P.PRICE
	2	TE.VP.PRICE

New Remove

Logic: SQL Default ?

```
SELECT
DECODE([TE.P.PRODUCTS].PC,
1200,
'Mouse', 50,
'Monitor', 560,
'Keyboard', 110, 0)
FROM sys.dual
```

OK Add Cancel

Rules Only Apply within the same Entry.

## 4. Create a rule.

Field Name	Description
<b>Rule Name</b>	Provide a name for the rule.
<b>Description</b>	Provide a description of the rule.
<b>Enabled?</b>	Indicate whether this rule is active or not.
<b>Rule Event</b>	Select a value from the list.
<b>Process subsequent rules?</b>	This rule could trigger other rules, indicate if the triggered rules should be followed or ignored.
<b>Dependencies</b>	<p>Use the New, Edit, and Remove buttons to indicate column dependencies. For example:</p> <p>Column = Price [All Values = Yes]</p> <p>Column = Quantity [All Values = Yes]</p>
<b>Results</b>	<p>Use to indicate results columns.</p> <p>Column Header = Total</p>
<b>Logic</b>	<p>Define the logic associated with the rule. Can be SQL Default or UI Rules.</p> <p>For example, the SQL could be:</p> <pre>SELECT [TE.P.PRICE] * [TE.P.QUANTITY], [TE.P.PRICE] * [TE.P.QUANTITY]  FROM sys.dual</pre> <p>For information on how to work with UI Rules, see the <i>Demand Management User's Guide</i>.</p>

For detailed examples, see ["Example of Using a Table Component on an Order Form" on page 122](#).

5. Click **OK** or **Add**.

## 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.

Field Type	Supported
User data fields	Yes
Report type fields	Yes
Request type fields	Yes
Project template fields	Yes
Request header types	No
Object types	No

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 similar.

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.

The screenshot shows the 'Field: New' dialog box. The 'Attributes' tab is selected. The 'Validation' field is set to 'Auto-Complete List'. The 'Multi-Select Enabled' option is set to 'Yes'. The 'Section Name' is 'Request Type Fields'. The 'Display Only' option is set to 'No'. The 'Transaction History' option is set to 'No'. The 'Notes History' option is set to 'No'. The 'Display on Search and Filter' option is set to 'Yes'. The 'Display' option is set to 'Yes'. The 'Search Validation' field is empty. The 'OK' button is highlighted.

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 specified in another field.

To use field dependent token evaluation, you must configure a validation with an object type, request type, report type, project template, or user data definition.

Sample set up of an object type using field-dependent tokens:

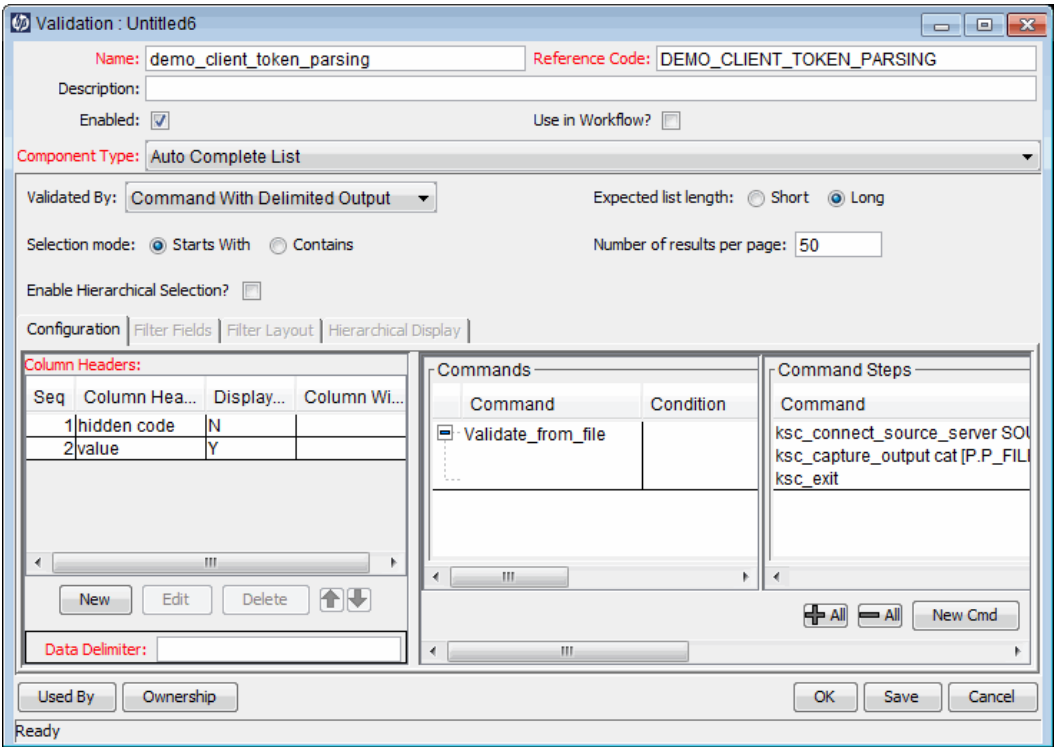
1. Generate a validation and set the following parameters:

- 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
```



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.

Object Type : Untitled21

Object Type Name: token\_parsing\_demo

Description:

Extension: Object Name Column: PARAMETER1

Object Category: Custom Objects Object Revision Column:

Meta Layer View: MPKGL\_ TOKEN\_PARSING\_DEMO

Enabled: ☒ Yes ☐ No

Fields | Layout | Commands | Ownership

Prompt	Token	Parameter Col.	Display...	Component Type	Validation	Requir...	Display Only
AutoCo...	P_AUTOC...	PARAMETER1	Y	Auto Complete List	demo_client_token_parsing	N	N
Filename	P_FILENA...	PARAMETER2	Y	Text Field	Text Field - 40	N	N

New Edit Remove

OK Save Cancel

Ready

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 (defined in "Example of Token Evaluation and Validation by Command with Delimited Output " on page 118)

3. To return values in the auto-complete, generate a file named parse\_test1.txt in the directory specified in the Base Path in the Environment Detail of "Your Env" environment that contains the following:

```
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 **Deployment Mgmt > Packages**.

The Packages Workbench opens.

2. Open a new package.
3. In the Package window, select a workflow, and click **New Line**.

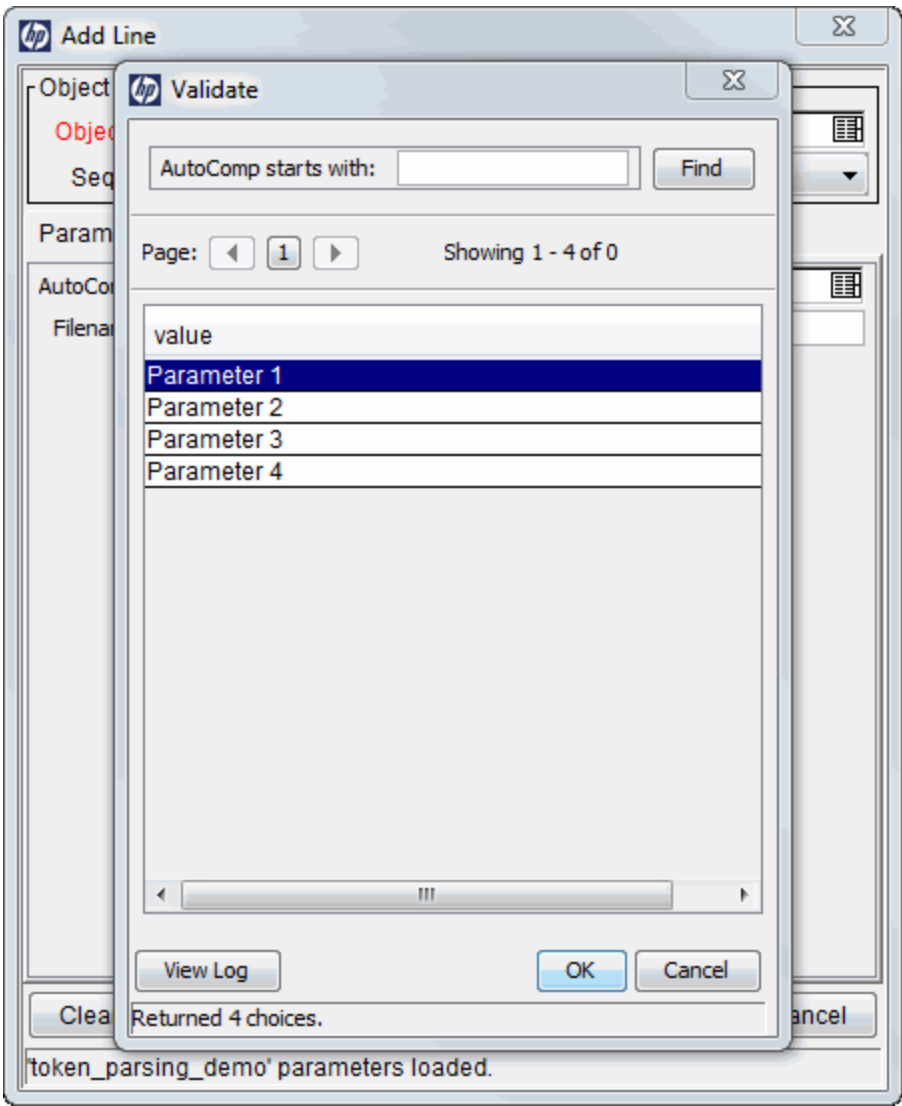
The Add Line window opens.

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**.



# Example of Using a Table Component on an Order Form

The following example shows the table component and 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 must first configure a new validation with the following specifications:

**Table 5-6. 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 5-7. Validations window**

**Validation : Product Order Information**

Name:  Reference Code:

Description:

Enabled: ☒ Use in Workflow? ☐

Component Type:

User Instructions:

Meta Layer View:

Table Columns | Form Layout | Rules

Column S...	Column He...	Column To...	Parameter Col.	Enab...	Component Type	Validation
1	Products	PRODUCTS	PARAMETER1	Y	Auto Complete List	Customized Auto Complete Lis
2	Quantity	QUANTITY	PARAMETER2	Y	Text Field	Numeric Text Field
3	Price	PRICE	PARAMETER3	Y	Text Field	Numeric Text Field
4	Total	TOTAL	PARAMETER4	Y	Text Field	Numeric Text Field

Used By:  Ownership:

OK Save Cancel

Ready

Next XYZ Corporation uses the **Rules** tab to set the default unit price based on the product selected.

**Table 5-7. 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,</pre>

**Table 5-7. Example, Set Unit Price rule settings, continued**

Setting	Value / Description
	'Monitor', 560, 'Keyboard', 110, 0)  FROM sys.dual

Then XYZ Corporation adds another rule to calculate and display the total line price in the **Total** column based on the values in the **Products** and **Quantity** fields.

**Table 5-8. 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	SELECT [TE.P.PRICE] * [TE.P.QUANTITY], [TE.P.PRICE] * [TE.P.QUANTITY]  from sys.dual

## Example Calculating Column Totals

The following example shows how to configure a column to calculate and display the column total.

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

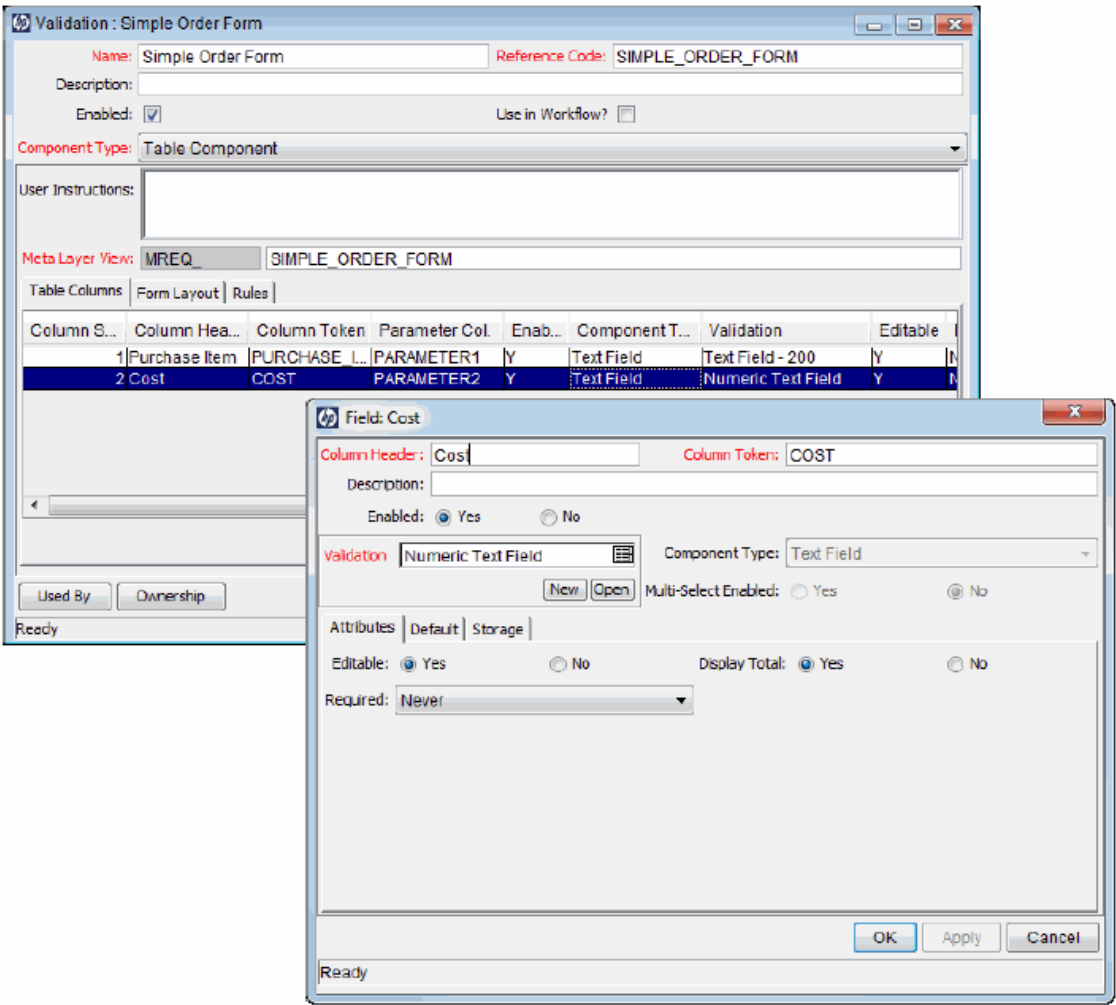
XYZ creates a validation with the following settings:

- **Component Type** = Table Component
- Column 1 = Purchase Item of **Validation** text field

- Column 2 = Cost of **Validation** number
- In the Field window for the Cost column, **Display Total** = Yes

The **Display Total** field is only enabled if the field's validation is a number.

**Figure 5-8. Sample validation for a Simple Order table component.**



XYZ Corporation 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 5-9. Sample table component displaying a column total

Simple Order Form

View

Add Row

Delete Rows

Copy

Paste

Move Up

Move Down

Seq	Purchase Item	Cost	
1	Flatscreen Monitor	1800	
2	Cable	40	
3	Monitor Switch	80	
Total		1920	

Export to Excel

Page

1

of 1

Show

5

Each Page

## Chapter 6: 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

The prefix for application server tokens is AS.

**Table A-1. Application server (AS) tokens**

Tokens	Description
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 *Installation and Administration Guide*.

### Benefit Tokens

The prefix for these tokens is FBEN.

**Table A-2. Benefit (FBEN) tokens**

Tokens	Description
ACTIVE_FLAG	Active flag of the financial benefit.
BENEFIT_ID	ID of the financial benefit.
BENEFIT_IS_FOR_ENTITY_NAME	Entity name to which the financial benefit is linked.
BENEFIT_IS_FOR_ID	ID of the asset, project, or proposal to which the financial benefit is linked.
BENEFIT_IS_FOR_NAME	Name of the asset, project, or proposal to which the financial benefit is



**Table A-2. Benefit (FBEN) tokens, continued**

Tokens	Description
	linked.
BENEFIT_NAME	Name of the financial benefit.
BENEFIT_URL	URL to view the financial benefit.
CREATED_BY	Username of the user who created the financial benefit.
CREATION_DATE	Date when the financial benefit was created.
DESCRIPTION	Description of the financial benefit.
END_PERIOD	End period of the financial benefit.
INITIATION_REQ	Initiation request ID of the financial benefit.
PERIOD_SIZE	Period size of the financial benefit.
REGION	Region associated with the financial benefit.
START_PERIOD	Start period of the financial benefit.
STATUS_CODE	Status code of the financial benefit.
STATUS_NAME	Status name of the financial benefit.

## Benefit > Financial Benefit Line Tokens

Within the token builder, Benefit > Financial Benefit Line is an empty folder.

## Contact Tokens

The prefix for contact tokens is CON.

**Table A-3. Contact (CON) tokens**

Tokens	Description
COMPANY	Company ID for which the contact works.
COMPANY_NAME	Name of the company for which the contact works.
CONTACT_ID	Contact ID (defined in the table KCRT_CONTACTS).
CREATED_BY	ID of the user who created the contact.

**Table A-3. Contact (CON) tokens, continued**

<b>Tokens</b>	<b>Description</b>
CREATION_DATE	Date the contact was created.
EMAIL_ADDRESS	Email address of the contact.
FIRST_NAME	First name of the contact.
FULL_NAME	Full name of the contact.
LAST_NAME	Last name of the contact.
LAST_UPDATED_BY	ID of the user who last updated the contact.
LAST_UPDATE_DATE	Date the contact was last updated.
PHONE_NUMBER	Phone number of the contact.
USERNAME	Contact username (if applicable). This can be the username for an external system, and not PPM Center.
USER_ID	UserID of the contact, if the contact is a PPM Center user.

## Distribution Tokens

The prefix for distribution tokens is DIST.

**Table A-4. Distribution (DIST) tokens**

<b>Tokens</b>	<b>Description</b>
CREATED_BY	ID of the user who created the distribution.
CREATED_BY_USERNAME	PPM Center username for the user who created the distribution.
DESCRIPTION	Release description.
DISTRIBUTION_ID	Distribution ID (defined in table KREL_DISTRIBUTION).
DISTRIBUTION_NAME	Distribution name.
DISTRIBUTION_STATUS	Distribution workflow status.
FEEDBACK_FLAG	Determines whether the distribution has fed back a specified value to the package lines being distributed.

**Table A-4. Distribution (DIST) tokens, continued**

<b>Tokens</b>	<b>Description</b>
FEEDBACK_VALUE	Value to be returned to the original package lines.
LAST_UPDATED_BY	ID of the user who last updated the distribution.
LAST_UPDATED_BY_USERNAME	PPM Center username for the user who last updated the distribution.
LAST_UPDATE_DATE	Date the distribution was last updated.
RELEASE_ID	ID of the release that created this distribution.
RELEASE_NAME	Name of the release that created this distribution.
WORKFLOW	Workflow used to process the distribution.

## Document Management Tokens

The prefix for document management tokens is DMS.

**Table A-5. Document Management (DMS) tokens**

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

The prefix for destination environment tokens is DEST\_ENV.

**Table A-6. Environment > Dest Env (DEST\_ENV) tokens**

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

**Table A-6. Environment > Dest Env (DEST\_ENV) tokens, continued**

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

**Table A-6. Environment > Dest Env (DEST\_ENV) tokens, continued**

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

## Environment > Dest Env > App Tokens

The prefix for destination environment application tokens is DEST\_ENV.APP.

**Table A-7. Environment > Dest Env > App (DEST\_ENV.APP) tokens**

<b>Tokens</b>	<b>Description</b>
APP_CODE	Short name (code) for the application.
APP_NAME	Descriptive name for the application.
CLIENT_BASE_PATH	Application-specific base (root) path of the client.
CLIENT_CON_PROTOCOL	Application-specific protocol used to connect to this client.
CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connection protocol.
CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
CLIENT_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this client.
CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
CLIENT_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the client.
CLIENT_USERNAME	Application-specific username PPM Center uses to log on to or access the client.
CREATED_BY	ID of the user who created the application.
CREATION_DATE	Date the application was created.
DB_LINK	For Oracle database type, the application-specific database link from the PPM Center schema to the database schema for the environment.
DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.
DB_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the database.
DB_USERNAME	Application-specific username or schema name that PPM Center uses to log on to or access the database.
DESCRIPTION	Application description.
ENABLED_FLAG	Flag that indicates whether the application is enabled and available for selection in package lines.
ENVIRONMENT_APP_ID	ID of the application in the table KENV_ENVIRONMENT_APPS.

**Table A-7. Environment > Dest Env > App (DEST\_ENV.APP) tokens, continued**

Tokens	Description
ENVIRONMENT_ID	ID of the environment with which the application is associated.
ENVIRONMENT_NAME	Name of the environment with which the application is associated.
LAST_UPDATED_BY	ID of the user who last updated the application.
LAST_UPDATE_DATE	Date the application was last updated.
SERVER_BASE_PATH	Application-specific base (root) path of the server.
SERVER_CON_PROTOCOL	Application-specific protocol used to connect to this server.
SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
SERVER_PASSWORD	Encrypted, application-specific password PPM Center uses to log on to or access the server.
SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
SERVER_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_USERNAME	Application-specific username PPM Center uses to log on to or access the server.
WORKBENCH_ENVIRONMENT_URL	URL of the environment window in the PPM Workbench.

## Environment > Dest Env > Env Tokens

The prefix for these tokens is DEST\_ENV.ENV.

**Table A-8. Environment > Dest Env > Env (DEST\_ENV.ENV) tokens**

Tokens	Description
CLIENT_BASE_PATH	Base (root) path of the client.
CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.



**Table A-8. Environment > Dest Env > Env (DEST\_ENV.ENV) tokens, continued**

<b>Tokens</b>	<b>Description</b>
CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
CLIENT_NAME	DNS name or IP address of the client computer.
CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
CLIENT_PASSWORD	Encrypted Password that PPM Center uses to log on to or access the client.
CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
CLIENT_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this client.
CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
CLIENT_TYPE_CODE	Validation value code of the client machine type.
CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
CREATED_BY	ID of the user who created the environment.
CREATION_DATE	Date the environment was created.
DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DATABASE_TYPE	Validation value code of the database type.
DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
DB_NAME	DNS name or IP address of the database server.
DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
DB_PASSWORD	Encrypted password that PPM Center uses to log on to or access

**Table A-8. Environment > Dest Env > Env (DEST\_ENV.ENV) tokens, continued**

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

**Table A-8. Environment > Dest Env > Env (DEST\_ENV.ENV) tokens, continued**

Tokens	Description
SERVER_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this server.
SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_TYPE_CODE	Validation value code of the server machine type.
SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

## Environment > Env Tokens

The prefix for these tokens is ENV.

**Table A-9. Environment > Env (ENV) tokens**

Tokens	Description
CLIENT_BASE_PATH	Base (root) path of the client.
CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
CLIENT_NAME	DNS name or IP address of the client computer.
CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
CLIENT_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this client.

**Table A-9. Environment > Env (ENV) tokens, continued**

<b>Tokens</b>	<b>Description</b>
CLIENT_TYPE_CODE	Validation value code of the client machine type.
CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
CREATED_BY	ID of the user who created the environment.
CREATION_DATE	Date the environment was created.
DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DATABASE_TYPE	Validation value code of the database type.
DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.
DB_NAME	DNS name or IP address of the database server.
DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
DB_PASSWORD	Encrypted password that PPM Center uses to log on to or access the database.
DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
DB_VERSION	Database version (such as 8.1.7).
DESCRIPTION	Environment description.
ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.

**Table A-9. Environment > Env (ENV) tokens, continued**

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

## Environment > Env > App Tokens

The prefix for these tokens is ENV.APP.

**Table A-10. Environment > Env > App (ENV.APP) tokens**

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

**Table A-10. Environment > Env > App (ENV.APP) tokens, continued**

Tokens	Description
DB_USERNAME	Application-specific username or schema name that PPM Center uses to log on to or access the database.
DESCRIPTION	Application description.
ENABLED_FLAG	Flag that indicates whether the application is enabled and available for selection in package lines.
ENVIRONMENT_APP_ID	ID of the application in the table KENV_ENVIRONMENT_APPS.
ENVIRONMENT_ID	ID of the environment with which the application is associated.
ENVIRONMENT_NAME	Name of the environment with which the application is associated.
LAST_UPDATED_BY	ID of the user who last updated the application.
LAST_UPDATE_DATE	Date the application was last updated.
SERVER_BASE_PATH	Application-specific base (root) path of the server.
SERVER_CON_PROTOCOL	Application-specific protocol used to connect to this server.
SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
SERVER_PASSWORD	Application-specific password PPM Center uses to log on to or access the server. This value is encrypted.
SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
SERVER_TRANSFER_PROTOCOL	Application-specific protocol used to transfer files to and from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_USERNAME	Application-specific username that PPM Center uses to log on to or access the server.
WORKBENCH_ENVIRONMENT_URL	URL of the environment window in the PPM Workbench.

## Environment > Env > Env Tokens

The prefix for these tokens is ENV.ENV.

**Table A-11. Environment > Env > Env (ENV.ENV) tokens**

<b>Tokens</b>	<b>Description</b>
CLIENT_BASE_PATH	Base (root) path of the client.
CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
CLIENT_NAME	DNS name or IP address of the client computer.
CLIENT_NT_DOMAIN	Domain name for the client, if the client machine is running Windows.
CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.
CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
CLIENT_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this client.
CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
CLIENT_TYPE_CODE	Validation value code of the client machine type.
CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
CREATED_BY	ID of the user who created the environment.
CREATION_DATE	Date the environment was created.
DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DATABASE_TYPE	Validation value code of the database type.
DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DB_LINK	For Oracle database type, the database link from the PPM Center schema to the environment's database schema.



**Table A-11. Environment > Env > Env (ENV.ENV) tokens, continued**

<b>Tokens</b>	<b>Description</b>
DB_NAME	DNS name or IP address of the database server.
DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
DB_PASSWORD	Password PPM Center uses to log on to or access the database. This value is encrypted.
DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
DB_VERSION	Database version (such as 8.1.7).
DESCRIPTION	Environment description.
ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
ENVIRONMENT_NAME	Environment name.
LAST_UPDATED_BY	ID of the user who last updated the environment.
LAST_UPDATE_DATE	Date the environment was last updated.
LOCATION	Environment location.
MSSQL_DB_NAME	For a Microsoft SQL Server database type, database name used to access the database from the command line.
SERVER_BASE_PATH	Base (root) path of the server.
SERVER_CON_PROTOCOL	Protocol used to connect to this server.
SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
SERVER_NAME	DNS name or IP address of the server computer.
SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.

**Table A-11. Environment > Env > Env (ENV.ENV) tokens, continued**

Tokens	Description
SERVER_PASSWORD	Password PPM Center uses to log on to or access the server. This value is encrypted.
SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
SERVER_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this server.
SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_TYPE_CODE	Validation value code of the server machine type.
SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

## Environment > Source Env Tokens

The prefix for these tokens is SOURCE\_ENV.

**Table A-12. Environment > Source Env (SOURCE\_ENV) tokens**

Tokens	Description
CLIENT_BASE_PATH	Base (root) path of the client.
CLIENT_CON_PROTOCOL	Protocol used to connect to this client.
CLIENT_CON_PROTOCOL_MEANING	Visible value of the client connect protocol.
CLIENT_NAME	DNS name or IP address of the client computer.
CLIENT_NT_DOMAIN	Domain name for a client running Windows.
CLIENT_ENABLED_FLAG	Flag that indicates whether the client portion of the environment is enabled.
CLIENT_PASSWORD	Password PPM Center uses to log on to or access the client. This value is encrypted.

**Table A-12. Environment > Source Env (SOURCE\_ENV) tokens, continued**

<b>Tokens</b>	<b>Description</b>
CLIENT_SQL_COMMAND	Default command line SQL*Plus command name.
CLIENT_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this client.
CLIENT_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this client.
CLIENT_TRANSFER_PROTOCOL_MEANING	Visible value of the client transfer protocol.
CLIENT_TYPE_CODE	Validation value code of the client machine type.
CLIENT_USERNAME	Username PPM Center uses to log on to or access the client.
CREATED_BY	ID of the user who created the environment.
CREATION_DATE	Date the environment was created.
DATABASE_ENABLED_FLAG	Flag that indicates whether the database portion of the environment is enabled.
DATABASE_TYPE	Validation value code of the database type.
DB_CONNECT_STRING	For Oracle database type, connect string used to access the database from the command line.
DB_JDBC_URL	JDBC URL used in Oracle 9i RAC configuration.
DB_LINK	For Oracle database type, database link from the PPM Center schema to the environment's database schema.
DB_NAME	DNS name or IP address of the database server.
DB_ORACLE_SID	For Oracle database type, SID of the database (often the same as the DB_CONNECT_STRING).
DB_PASSWORD	Password PPM Center uses to log on to or access the database. This value is encrypted.
DB_PORT_NUMBER	For Oracle database type, port number on which SQL*Net is listening for remote SQL connections on the database server.
DB_USERNAME	Username or schema name PPM Center uses to log on to or access the database.
DB_VERSION	Database version (such as 8.1.7).

**Table A-12. Environment > Source Env (SOURCE\_ENV) tokens, continued**

<b>Tokens</b>	<b>Description</b>
DESCRIPTION	Environment description.
ENABLED_FLAG	Flag that Indicates whether the environment is enabled and available for use in workflows.
ENVIRONMENT_ID	ID of the environment in the table KENV_ENVIRONMENTS.
ENVIRONMENT_NAME	Environment name.
LAST_UPDATED_BY	ID of the user who last updated the environment.
LAST_UPDATE_DATE	Date the environment was last updated.
LOCATION	Environment location.
MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
SERVER_BASE_PATH	Base (root) path of the server.
SERVER_CON_PROTOCOL	Protocol used to connect to this server.
SERVER_CON_PROTOCOL_MEANING	Visible value of the server connection protocol.
SERVER_ENABLED_FLAG	Flag that indicates whether the server portion of the environment is enabled.
SERVER_NAME	DNS name or IP address of the server computer.
SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
SERVER_PASSWORD	Encrypted password PPM Center uses to log on to or access the server.
SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
SERVER_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this server.
SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_TYPE_CODE	Validation value code of the server machine type.
SERVER_USERNAME	Username PPM Center uses to log on to or access the server.

**Table A-12. Environment > Source Env (SOURCE\_ENV) tokens, continued**

Tokens	Description
WORKBENCH_ ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

## Environment > Source Env > App Tokens

The prefix for these tokens is SOURCE\_ENV.APP.

**Table A-13. Environment > Source Env > App (SOURCE\_ENV.APP) tokens**

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

**Table A-13. Environment > Source Env > App (SOURCE\_ENV.APP) tokens, continued**

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

## Environment > Source Env > Env Tokens

The prefix for these tokens is SOURCE\_ENV.ENV.

**Table A-14. Environment Source Env > Env (SOURCE\_ENV.ENV) tokens**

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

**Table A-14. Environment Source Env > Env (SOURCE\_ENV.ENV) tokens, continued**

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



**Table A-14. Environment Source Env > Env (SOURCE\_ENV.ENV) tokens, continued**

Tokens	Description
SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
SERVER_NAME	DNS name or IP address of the server computer.
SERVER_NT_DOMAIN	Domain name for the server, if the server machine type is Windows.
SERVER_PASSWORD	Password that PPM Center uses to log on to or access the server. This value is encrypted.
SERVER_SQL_COMMAND	Default command line SQL*Plus command name.
SERVER_STREAM_ENCODING	Encoding used to send messages and commands to or decode messages from this server.
SERVER_TRANSFER_PROTOCOL	Protocol used to transfer files to or from this server.
SERVER_TRANSFER_PROTOCOL_MEANING	Visible value of the server transfer protocol.
SERVER_TYPE_CODE	Validation value code of the server machine type.
SERVER_USERNAME	Username PPM Center uses to log on to or access the server.
WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the PPM Workbench.

## Execution Tokens

The prefix for these tokens is EXEC.

**Table A-15. Execution (EXEC) 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]'
```

## Forecast Actuals Tokens

The prefix for these tokens is BGT.

**Table A-16. Forecast Actuals (BGT) tokens**

Tokens	Description
ACTIVE_FLAG	Active flag for the budget.
BUDGET_ID	ID of the budget (defined in the table KCST_BUDGETS).
BUDGET_IS_FOR_ENTITY_NAME	Entity name (work plan, program, or org unit) to which the budget is linked.
BUDGET_IS_FOR_ID	ID of the work plan/program/org unit to which the budget is linked.
BUDGET_IS_FOR_NAME	Name of the work plan/program/org unit to which the budget is linked.
BUDGET_NAME	Name of the budget.
BUDGET_ROLLS_UP_TO_ID	ID of the budget into which this budget rolls up.
BUDGET_ROLLS_UP_TO_NAME	Name of the budget into which this budget rolls up.
BUDGET_URL	URL used to view this budget.
CREATED_BY	Username of the user who created the budget.
CREATION_DATE	Date the budget was created.

**Table A-16. Forecast Actuals (BGT) tokens, continued**

Tokens	Description
DESCRIPTION	Budget description.
END_PERIOD	Budget end period.
INITIATION_REQ	Budget initiation request ID.
PERIOD_SIZE	Budget period size.
REGION	Region associated with the budget.
START_PERIOD	Budget start period.
STATUS_CODE	Budget status code.
STATUS_NAME	Budget status name.

## Notification Tokens

The prefix for these tokens is NOTIF.

**Table A-17. Notification (NOTIF) tokens**

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

# Organization Unit Tokens

The prefix for these tokens is ORG.

**Table A-18. Organization Unit (ORG) tokens**

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

## Package Tokens

The prefix for these tokens is (PKG).

**Table A-19. Package (PKG) tokens**

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

**Table A-19. Package (PKG) tokens, continued**

<b>Tokens</b>	<b>Description</b>
NOTES	All notes for the package.
NUMBER	Package name/number.
PACKAGE_GROUP_CODE	Package group code.
PACKAGE_GROUP_NAME	Package group name.
PACKAGE_ID	ID of the package in the table KDLV_PACKAGES.
PACKAGE_TYPE	Validation value meaning of the package type.
PACKAGE_TYPE_CODE	Validation value code for the package type.
PACKAGE_URL	URL of the package in the standard interface.
PARENT_REQUEST_ID	ID of the request that created this package (if applicable).
PERCENT_COMPLETE	Percent complete of the package.
PRIORITY	Package priority.
PRIORITY_CODE	Validation value code for the package priority.
PRIORITY_NAME	Validation value meaning of the package priority.
PRIORITY_SEQ	Package priority sequence.
PROJECT_CODE	Validation value code of the work plan to which the package belongs.
PROJECT_NAME	Validation value meaning of the work plan to which the package belongs.
REQUESTED_BY_EMAIL	Email address of the user who requested the package.
REQUESTED_BY_USERNAME	PPM Center username of the user who requested the package.
REQUESTED_BY_USER_ID	ID of the user who requested the package.
RUN_GROUP	Package run group.
STATUS	Validation value meaning for the package status.
STATUS_CODE	Validation value code for the package status.
SUBMIT_DATE	Date on which the package was submitted.

**Table A-19. Package (PKG) tokens, continued**

Tokens	Description
WORKBENCH_PACKAGE_NO_LINK	Package URL in the PPM Workbench.
WORKBENCH_PACKAGE_URL	Package screen URL in the PPM Workbench.
WORKFLOW_ID	ID of the workflow that the package uses.
WORKFLOW_NAME	Name of the workflow that the package uses.

## Package > Package Line Tokens

The prefix for these tokens is PKG. PKGL.

**Table A-20. Package > Package Line (PKG. PKGL) tokens**

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

## Package > Pending Reference Tokens

The prefix for these tokens is PKG.PEND.

**Table A-21. Package > Pending Reference (PKG.PEND) tokens**

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



**Table A-21. Package > Pending Reference (PKG.PEND) tokens, continued**

Tokens	Description
PERCENT_COMPLETE	Current percent complete value associated with the entity that is blocked by the package.
ENTITY_TYPE_ID	ID of the type of entity that is blocked by the package.
ENTITY_TYPE_NAME	Name of the type of entity blocked by the package.

## Package Line Tokens

The prefix for these tokens is PKGL.

**Table A-22. Package Line (PKGL) tokens**

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

## Program Tokens

The prefix for these tokens is PRG.

**Table A-23. Program (PRG) tokens**

Tokens	Description
CREATED_BY	ID of the user who created the program.
CREATED_BY_USERNAME	Name of the user who created the program.
LAST_UPDATED_BY	ID of the user who last updated the program.
LAST_UPDATED_BY_USERNAME	Name of the user who last updated the program.
MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PROGRAM_MANAGER	IDs of the users assigned to manage the program.

## Project Tokens

The prefix for these tokens is PRJ.

**Table A-24. Project (PRJ) tokens**

Tokens	Description
ACTUAL_DURATION	Actual duration of the work plan.
ACTUAL_EFFORT	Actual effort associated with the work plan.
ACTUAL_FINISH_DATE	Actual finish date of the work plan.
ACTUAL_START_DATE	Actual start date of the work plan.
BUDGET_ID	ID of the budget linked to the work plan.

**Table A-24. Project (PRJ) tokens, continued**

<b>Tokens</b>	<b>Description</b>
BUDGET_NAME	Name of the budget linked to the work plan.
CONFIDENCE_CODE	Code of the confidence value specified by the user.
CONFIDENCE_NAME	Name of the confidence value specified by the user.
CREATED_BY	User who created the work plan.
CREATED_BY_EMAIL	Email address of the user who created the work plan.
CREATED_BY_USERNAME	Username of the person who created the work plan.
CREATION_DATE	Creation date of the work plan.
DEPARTMENT_CODE	Code of the department value specified by the user.
DEPARTMENT_NAME	Name of the department value specified by the user.
DESCRIPTION	Description of the work plan.
ESTIMATED_REMAINING_DURATION	Estimated time remaining for the work plan.
ESTIMATED_REMAINING_EFFORT	Estimated remaining effort involved in the work plan.
ESTIMATED_FINISH_DATE	Estimated finish date of the work plan.
LAST_UPDATE_DATE	Date on which the work plan was last updated.
LAST_UPDATED_BY	Last person to update the work plan.
LAST_UPDATED_BY_EMAIL	Email address of the last person to update the project plan.
LAST_UPDATED_BY_USERNAME	Username of the last person to update the work plan.
MASTER_PROJECT_ID	ID of the master project.
MASTER_PROJECT_NAME	Name of the master project.
MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent note.
MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.

**Table A-24. Project (PRJ) tokens, continued**

<b>Tokens</b>	<b>Description</b>
MOST_RECENT_NOTE_TEXT	Text of the most recent note.
MOST_RECENT_NOTE_TYPE	Type of the most recent note (USER or FIELD CHANGE).
MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent user note.
MOST_RECENT_USER_NOTE_AUTHOR_USERNAME	Username of the author of the most recent user note.
MOST_RECENT_USER_NOTE_AUTHORED_DATE	Date of the most recent user note.
MOST_RECENT_USER_NOTE_TEXT	Text of the most recent user note.
PARENT_PROJECT_ID	ID of the parent work plan.
PARENT_PROJECT_NAME	Name of the parent work plan.
PERCENT_COMPLETE	Percent of the work plan completed.
PRIORITY	Priority of the work plan.
PROGRAM_ID	Delimited list of program ids of all programs associated with the project.
PROGRAM_MANAGER	Delimited list of manager ids of all programs associated with the project.
PROGRAM_MANAGER_USERNAME	Delimited list of manager usernames of all programs associated with the project.
PROGRAM_NAME	Delimited list of program names of all programs associated with the project.
PROJECT_ID	Number that uniquely identifies the work plan (same as PROJECT_NUMBER) in the table KDRV_PROJECTS.
PROJECT_MANAGER	Manager of the work plan. Use this token to get the project manager information from the project container.
PROJECT_MANAGER_EMAIL	Email address of the project manager.
PROJECT_MANAGER_USERNAME	Username of the project manager.

**Table A-24. Project (PRJ) tokens, continued**

<b>Tokens</b>	<b>Description</b>
PROJECT_NAME	Work plan name.
PROJECT_NAME_LINK	Standard hyperlink to the work plan in HTML-formatted notifications.
PROJECT_NUMBER	Number that uniquely identifies the work plan (same as PROJECT_ID).
PROJECT_PATH	Work plan path. This is a hierarchy of parent work plans that contain this work plan.
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.
PROJECT_RESOURCES	Resources for this project.
PROJECT_STAKEHOLDER	Delimited list of user ids of the project stakeholders.
PROJECT_STAKEHOLDER_EMAIL	Delimited list of emails of the project stakeholders.
PROJECT_STAKEHOLDER_USERNAME	Delimited list of usernames of the project stakeholders.
PROJECT_STATE	Work plan state.
PROJECT_SUMMARY_TASK_OWNERS	Summary task owners for the work plan.
PROJECT_TEMPLATE	Name of the project template used to create the project plan.
PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
PROJECT_URL	URL for the Project Overview page of the work plan.
REGIONAL_CALENDAR	Name of the regional calendar for the work plan
SCHEDULED_EFFORT	Scheduled effort defined in the work plan.
SCHEDULED_DURATION	Scheduled duration for the work plan.
SCHEDULED_FINISH_DATE	Finish date scheduled for the work plan.
SCHEDULED_START_DATE	Start date scheduled for the work plan.
SUMMARY_CONDITION	Summary condition of the work plan.
WORKBENCH_PROJECT_URL	URL used to access this work plan in the PPM Workbench.

## Project Detail Tokens

The prefix for these tokens is PRJD.

**Table A-25. Project Detail (PRJD) tokens**

Tokens	Description
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

The prefix for these tokens is REL.

**Table A-26. Release (REL) tokens**

Tokens	Description
RELEASE_ID	ID of the release in the KREL_RELEASES table.
RELEASE_NAME	Release name.
RELEASE_STATUS	Release status.
CREATED_BY	ID of the user who created the release.
CREATED_BY_USERNAME	PPM Center username of the user who created the release.
LAST_UPDATED_BY	ID of the user who last updated the release.
LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the release.
LAST_UPDATE_DATE	Date on which the release was last updated.
MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
MOST_RECENT_NOTE_TEXT	Text of the most recent note.

**Table A-26. Release (REL) tokens, continued**

Tokens	Description
RELEASE_MANAGER	PPM Center user designated as the release manager.
RELEASE_TEAM	Group of PPM Center users associated with the release.
RELEASE_GROUP	High-level categorization of the release.
DESCRIPTION	Release description.
NOTES	Notes contained within the release.

## Release > Distribution Tokens

The prefix for these tokens is REL.DIST.

**Table A-27. Release > Distribution (REL.DIST) tokens**

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

## Report Submission Tokens

The prefix for these tokens is RP.

**Table A-28. Report submission (RP) tokens**

Tokens	Description
CREATED_BY	ID of the user who submitted the report.
CREATED_BY_USERNAME	Name of the user who created the report.
CREATION_DATE	The date the report was submitted.
FILENAME	The filename of the report.
LAST_UPDATE_DATE	The date the report was last updated.
LAST_UPDATED_BY	ID of the user who last updated the report.
REPORT_LOG_URL	URL for the log file of the report.
REPORT_SUBMISSION_ID	Submission ID of the report.
REPORT_TYPE_ID	ID of the report type.
REPORT_TYPE_NAME	The name of the report type.
REPORT_URL	URL of the report.
STATUS	The status of the report.
STATUS_CODE	The status code of the report.
SUBMISSION_LANGUAGE	The language of the submitted report.
WORKBENCH_REPORT_TYPE_URL	URL to access the report type windows for this report type in the PPM Workbench.

## Request Tokens

The prefix for these tokens is REQ.



**Table A-29. Request (REQ) tokens**

<b>Tokens</b>	<b>Description</b>
APPLICATION_CODE	Validation value code for the application to which the request is assigned.
APPLICATION_NAME	Validation value meaning of the application to which the request is assigned.
ASSIGNED_TO_EMAIL	Email address of the user to whom the request is assigned.
ASSIGNED_TO_GROUP_ID	ID of the security group to which the request is assigned.
ASSIGNED_TO_GROUP_NAME	Name of the security group to which the request is assigned.
ASSIGNED_TO_USERNAME	PPM Center username of the user to whom the request is assigned.
ASSIGNED_TO_NAME	Full name of the assigned user.
ASSIGNED_TO_USER_ID	ID of the user to whom the request is assigned.
COMPANY	Company employing the user who created the request.
COMPANY_NAME	Name of the company employing the user who created the request.
CONTACT_EMAIL	Email address of the contact for the request.
CONTACT_NAME	Full name of the contact for the request.
CONTACT_PHONE_NUMBER	Phone number of the contact for the request.
CREATED_BY	ID of the user who created the request.
CREATED_BY_EMAIL	Email address of the user who created the request.
CREATED_BY_NAME	Full name of the created by user.
CREATED_BY_USERNAME	PPM Center username of the user who last updated the request.
CREATION_DATE	Date on which the request was created.
DEPARTMENT_CODE	Validation value code of the department for the request.
DEPARTMENT_NAME	Validation value meaning of the department for the request.
DESCRIPTION	Request description.
LAST_UPDATED_BY	ID of the user who last updated the request.

**Table A-29. Request (REQ) tokens, continued**

<b>Tokens</b>	<b>Description</b>
LAST_UPDATED_BY_EMAIL	Email address of the user who last updated the request.
LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the request.
LAST_UPDATE_DATE	Date on which the request was last updated.
MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
MOST_RECENT_NOTE_TEXT	Text of the most recent note.
MOST_RECENT_NOTE_TYPE	Type of the most recent note (USER or FIELD CHANGE).
MOST_RECENT_NOTE_CONTEXT	In the case of requests, this is the request status; blank in all other cases.
MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent user note.
MOST_RECENT_USER_NOTE_AUTHOR_USERNAME	Username of the author of the most recent user note.
MOST_RECENT_USER_NOTE_AUTHORED_DATE	Date of the most recent user note.
MOST_RECENT_USER_NOTE_TEXT	Text of the most recent user note.
MOST_RECENT_USER_NOTE_CONTEXT	Request status.
NOTES	All notes for the request.
PERCENT_COMPLETE	Percent of the request that is completed.
PRIORITY_CODE	Validation value code of the request priority.
PRIORITY_NAME	Validation value meaning of the request priority.
PROJECT_CODE	Validation value code of the work plan to which the request belongs.

**Table A-29. Request (REQ) tokens, continued**

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

## Request > Pending Reference Tokens

The prefix for these tokens is REQ.PEND.

**Table A-30. Request > Pending Reference (REQ.PEND) tokens**

Tokens	Description
ID	ID of the entity that the request is blocking.

**Table A-30. Request > Pending Reference (REQ.PEND) tokens, continued**

<b>Tokens</b>	<b>Description</b>
NAME	Name of the entity that the request is blocking.
DETAIL	Detail information for the entity that the request is blocking.
DESCRIPTION	Description of the entity that the request is blocking.
STATUS_ID	ID of the state or code of the status of the entity that the request is blocking.
STATUS_NAME	Name of the status (or state) of the entity that the request is blocking.
STATE	Name of the state of the entity of the request that is blocked by the request.
ASSIGNED_TO_USERNAME	Name of the assigned user (or resource) of the entity that the request is blocking.
ASSIGNED_TO_USER_ID	Username of the assigned user (or resource) of the entity that the request is blocking.
ASSIGNED_TO_GROUP_NAME	Name of the assigned group (or resource group) of the entity that the request is blocking.
ASSIGNED_TO_GROUP_ID	ID of the assigned group (or resource group) of the entity that the request is blocking.
RESOURCE_USERNAME	Name of the resource associated with the entity that the request is blocking.
RESOURCE_ID	Username of the assigned user (or resource) associated with the entity that the request is blocking.
RESOURCE_GROUP_NAME	Name of the assigned group (or resource group) associated with the entity that is blocked by the request.
RESOURCE_GROUP_ID	ID of the assigned group (or resource group) associated with the entity that is being blocked by the request.
PERCENT_COMPLETE	Current percent complete value associated with the entity that the request is blocking.
ENTITY_TYPE_ID	ID of the type of entity that the request is blocking.
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 190](#).

## Request Detail Tokens

The prefix for these tokens is REQD.

**Table A-31. Request Detail (REQD) tokens**

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

The prefix for these tokens is RSCP.

**Table A-32. Resource Pool (RSCP) tokens**

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

## Security Group Tokens

The prefix for these tokens is SG.

**Table A-33. Security Group (SG) tokens**

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

## Skill Tokens

The prefix for these tokens is SKL.

**Table A-34. Skill (SKL) tokens**

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

## Staffing Profile Tokens

The prefix for these tokens is STFP.

**Table A-35. Staffing Profile (STFP) tokens**

<b>Tokens</b>	<b>Description</b>
ACTIVE_FLAG	Active flag of the staffing profile.
CREATED_BY	Username of the user who created the staffing profile.
CREATION_DATE	Date on which the staffing profile was created.
DESCRIPTION	Description of the staffing profile.
END_PERIOD	End period of the staffing profile.
PERIOD_SIZE	Period size of the staffing profile.
STAFFING_PROFILE_URL	URL to view this staffing profile.
STAFF_PROF_ID	ID of the staffing profile in table KRSC_STAFF_PROFS.

**Table A-35. Staffing Profile (STFP) tokens, continued**

<b>Tokens</b>	<b>Description</b>
STAFF_PROF_IS_FOR_ENTITY_NAME	Entity name to which the staffing profile is linked.
STAFF_PROF_IS_FOR_ID	ID of the work plan, program or org unit to which the staffing profile is linked.
STAFF_PROF_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).
STAFF_PROF_NAME	Staffing profile name.
START_PERIOD	Staffing profile start period.
STATUS_CODE	Staffing profile status code.
STATUS_NAME	Staffing profile status name.

## Step TXN (Transaction) Tokens

The prefix for these tokens is WST.

**Table A-36. Step TXN (WST) tokens**

<b>Tokens</b>	<b>Description</b>
CONCURRENT_REQUEST_ID	Identifier for the Oracle Concurrent Request for the Workflow Step Transaction.
CREATED_BY	User ID of the user who created the Workflow Step Transaction.
CREATION_DATE	Date the Workflow Step Transaction was created.
ERROR_MESSAGE	Any system level error message associated with the Workflow Step Transaction.
EXECUTION_BATCH_ID	Execution batch ID for the Workflow Step Transaction.
HIDDEN_STATUS	Hidden status of the Workflow Step Transaction.
LAST_UPDATED_BY	User ID of the user who last updated the Workflow Step Transaction.
LAST_UPDATED_BY_EMAIL	Email address of the user who last updated the Workflow Step Transaction.
LAST_UPDATED_BE_	Username of the user who last updated the Workflow Step



**Table A-36. Step TXN (WST) tokens, continued**

Tokens	Description
USERNAME	Transaction.
LAST_UPDATE_DATE	Date the Workflow Step Transaction was last updated.
STATUS	Status of the Workflow Step Transaction.
STEP_TRANSACTION_ID	Transaction ID for the Workflow Step Transaction.
TIMEOUT_DATE	Date of the last timeout on the Workflow Step Transaction.
USER_COMMENT	Any comments a user added to the Workflow Step Transaction.
WORKFLOW_ID	Workflow ID for the Workflow Step Transaction.
WORKFLOW_STEP_ID	Workflow step ID for the Workflow Step Transaction.
NEW_HIDDEN_STATUS	New hidden status of the Workflow Step Transaction.
OLD_HIDDEN_STATUS	Old hidden status of the Workflow Step Transaction.
NEW_STATUS	New status of the Workflow Step Transaction.
OLD_STATUS	Old status of the Workflow Step Transaction.

## System Tokens

The prefix for these tokens is SYS.

**Table A-37. System (SYS) tokens**

Tokens	Description
DATE	Date on which the token is parsed.
FULL_NAME	Full name of the PPM Center user.
ITG_TIME_STAMP	Date and time stamp when the token is parsed. You can use this token with the <code>ksc_store</code> command.
NEWLINE	New line character.
TIME_STAMP	Date and time stamp. (Deprecated)
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.

**Table A-37. System (SYS) tokens, continued**

<b>Tokens</b>	<b>Description</b>
UNIX_NEWLINE	UNIX new line character.
USERNAME	PPM Center username for the user currently logged on to PPM Center.
USER_ID	ID of the user currently logged on to PPM Center.

## Task Tokens

The prefix for these tokens is TSK.

**Table A-38. Tasks (TSK) tokens**

<b>Tokens</b>	<b>Description</b>
ACTUAL_DURATION	Actual task duration.
ACTUAL_EFFORT	Actual effort associated with the task.
ACTUAL_FINISH_DATE	Actual date the task finished.
ACTUAL_START_DATE	Actual date the task started.
CONFIDENCE_CODE	Confidence code that the user specified.
CONFIDENCE_NAME	Confidence name that the user specified.
CONSTRAINT_DATE	Task constraint date.
CREATED_BY	User who created the task.
CREATED_BY_EMAIL	Email address of the user who created the task.
CREATED_BY_USERNAME	Username of the user who created the task.
CREATION_DATE	Date the task was created.
DEPARTMENT_CODE	Department code value the user specified.
DEPARTMENT_NAME	Department name the user specified.
DESCRIPTION	TASK description.
ESTIMATED_REMAINING_DURATION	Estimated time remaining to complete the task.
ESTIMATED_REMAINING_EFFORT	Estimated remaining effort involved in the task.

**Table A-38. Tasks (TSK) tokens , continued**

<b>Tokens</b>	<b>Description</b>
ESTIMATED_FINISH_DATE	Estimated finish date of the task.
HAS_EXCEPTIONS	Flag to show whether or not the task has exceptions.
LAST_UPDATE_DATE	Date on which the task was last updated.
LAST_UPDATED_BY	Last user to update the task.
LAST_UPDATED_BY_EMAIL	Email address of the last user to update the task.
LAST_UPDATED_BY_USERNAME	Username of the last person to update the task.
MASTER_PROJECT_NAME	Name of the master project.
MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.
MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PARENT_PROJECT_ID	ID of the parent work plan.
PARENT_PROJECT_NAME	Name of the parent work plan.
PARENT_TASK_ID	ID of the parent task.
PERCENT_COMPLETE	Percentage of the task completed.
PRIORITY	Task priority.
PROJECT_PATH	Work plan path. Hierarchy of parent work plans that contain this task.
PROJECT_TEMPLATE	Name of the project template used to create the work plan that contains the task.
PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
RESOURCE_ID	ID of the resource assigned to the task.
RESOURCE_EMAIL	Email address of the resource.
RESOURCE_GROUP_ID	ID of the resource group assigned to the task.

**Table A-38. Tasks (TSK) tokens , continued**

<b>Tokens</b>	<b>Description</b>
RESOURCE_GROUP_NAME	Name of the resource group assigned to the task.
RESOURCE_USERNAME	Username of the resource.
SCHEDULED_EFFORT	Scheduled effort involved in the task.
SCHEDULED_DURATION	Duration scheduled for the task.
SCHEDULED_FINISH_DATE	Finish date scheduled for the task.
SCHEDULED_START_DATE	Start date scheduled for the task.
SCHEDULING_CONSTRAINT	Scheduling constraint for the task.
TASK_ID	The number that uniquely identifies the task (same as TASK_NUMBER). This corresponds to the PROJECT_ID column in table KDRV_PROJECTS.
TASK_NAME	Task name.
TASK_NAME_LINK	Standard hyperlink to the task in HTML-formatted notifications.
TASK_NUMBER	Number that uniquely identifies the task (same as TASK_ID).
TASK_STATE	The task state.
TASK_URL	URL for the task Detail page.
WORKBENCH_TASK_URL	URL to access this task in the PPM Workbench.

## Tasks > Pending Tokens

The prefix for these tokens is TSK.PEND.

**Table A-39. Tasks > Pending (TSK.PEND) tokens**

<b>Tokens</b>	<b>Description</b>
ID	ID of the entity that is blocked by the task.
NAME	Name of the entity that is blocked by the task.
DETAIL	Detail information for the entity that is blocked by the task as shown in the References field.
DESCRIPTION	Description of the entity that is blocked by the task.

**Table A-39. Tasks > Pending (TSK.PEND) tokens , continued**

<b>Tokens</b>	<b>Description</b>
STATUS_ID	ID of the state or the code of the status of the entity that is being blocked by the task.
STATUS_NAME	Name of the status (or state) of the entity that is blocked by the task.
STATE	Name of the state of the entity blocked by the task.
ASSIGNED_TO_USERNAME	Name of the assigned user (or resource) of the entity blocked by the task.
ASSIGNED_TO_USER_ID	Username of the assigned user (or resource) of the entity blocked by the task.
ASSIGNED_TO_GROUP_NAME	Name of the assigned group (or resource group) of the entity blocked by the task.
ASSIGNED_TO_GROUP_ID	ID of the assigned group (or resource group) of the entity that is being blocked by the task.
RESOURCE_USERNAME	Name of the resource associated with the entity that is blocked by the task.
RESOURCE_ID	Username of the resource (or assigned user) associated with the entity blocked by the task.
RESOURCE_GROUP_NAME	Name of the resource group (or assigned user) associated with the entity blocked by the task.
RESOURCE_GROUP_ID	ID of the resource group (or assigned group) associated with the entity blocked by the task.
PERCENT_COMPLETE	Current percent complete value associated with the entity blocked by the task.
ENTITY_TYPE_ID	ID of the type of entity blocked by the task.
ENTITY_TYPE_NAME	Name of the type of entity that is being blocked by the task.

## Time Management Notification Tokens

The prefix for these tokens is TMG.

**Table A-40. Time Management Notification (TMG) tokens**

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

## User Tokens

The prefix for these tokens is USR.

**Table A-41. User (USR) tokens**

Tokens	Description
AUTHENTICATION_MODE_CODE	Authentication mode for the user (such as LDAP).
AUTHENTICATION_MODE_NAME	Authentication mode for the user (such as LDAP).
COMPANY	Company that employs the user.
COMPANY_NAME	Name of the company that employs the user.
CREATED_BY	ID of the user who created the user.
CREATED_BY_FULL_NAME	Full name of the "created by" user.
CREATED_BY_USERNAME	PPM Center username of the user who created the user.
CREATION_DATE	Date on which the user was created.
DEPARTMENT_CODE	Lookup code of the department the user belongs to (lookup type = DEPT).
DEPARTMENT_NAME	Name of the department to which the user belongs.
EMAIL_ADDRESS	Email address of the user.
END_DATE	Date on which the user is made inactive in the application.
FIRST_NAME	First name of the user.
FULL_NAME	Full name of the user.
LAST_NAME	Last name of the user.

**Table A-41. User (USR) tokens , continued**

<b>Tokens</b>	<b>Description</b>
LAST_UPDATED_BY	ID of the user who last updated the user.
LAST_UPDATE_BY_FULL_NAME	Full name of the last updated by user.
LAST_UPDATED_BY_USERNAME	PPM Center username of the user who last updated the user.
LAST_UPDATE_DATE	Date the user was last updated.
LOCATION_CODE	Lookup code of the user's location (lookup type = RSC - Location).
LOCATION_NAME	Name of the user's location.
MANAGER_USERNAME	Username of the user's manager.
MANAGER_USER_ID	ID of the user's manager.
PASSWORD	Password for the user to use to log on to PPM Center. This value is encrypted.
PASSWORD_EXPIRATION_DATE	Date the password needs to be reset for the user.
PASSWORD_EXPIRATION_DAYS	Number of days until the password must be reset for the user.
PHONE_NUMBER	Phone number of the user.
PRIMARY_ROLE_ID	ID of the primary role associated with the user.
PRIMARY_ROLE_NAME	Name of the primary role associated with the user.
REGION	Region associated with the user.
REGIONAL_CALENDAR	Name of the regional calendar for the user.
RESOURCE_CATEGORY_CODE	Lookup code of resource category (lookup type = RSC - Category) to which the user belongs.
RESOURCE_CATEGORY_NAME	Name of the category to which the user belongs.
RESOURCE_TITLE_CODE	Lookup code of the user's resource title (lookup type = RSC - Resource Title).
RESOURCE_TITLE_NAME	Name of the user's resource title.
START_DATE	Date the user is made active in the application.
USERNAME	Username for the user to use to log on to PPM Center.
USER_ID	ID of the user in the table KNTA_USERS.
WORKLOAD_CAPACITY	Workload capacity of the user (% of FTE)

## Validation Tokens

The prefix for these tokens is VAL.

**Table A-42. Validation (VAL) tokens**

Tokens	Description
COMPONENT_TYPE	Component type associated with the validation.
CREATED_BY	ID of the user who created the validation.
CREATION_DATE	Date the validation was created.
DESCRIPTION	Description of the validation.
LAST_UPDATED_BY	ID of the user who last updated the validation.
LAST_UPDATE_DATE	Date the validation was last updated.
LOOKUP_TYPE	Lookup type associated with the validation (if applicable).
VALIDATION_ID	ID of the validation in the table KNTA_VALIDATIONS.
VALIDATION_NAME	Name of the validation.
VALIDATION_SQL	SQL statement associated with the validation (if applicable).
WORKBENCH_VALIDATION_URL	URL for the validation in the PPM Workbench.

## Validation > Value Tokens

The prefix for these tokens is VAL.VALUE.

**Table A-43. Validation > Value (VAL.VALUE) tokens**

Tokens	Description
CREATED_BY	ID of the user who created the value.
CREATION_DATE	Date the value was created.
DEFAULT_FLAG	Flag to indicate whether the value is the default value for the associated lookup type.
DESCRIPTION	Description of the value.



**Table A-43. Validation > Value (VAL.VALUE) tokens , continued**

Tokens	Description
ENABLED_FLAG	Flag that indicates whether the value is available for selection in a list.
LAST_UPDATED_BY	ID of the user who last updated the value.
LAST_UPDATE_DATE	Date the value was last updated.
LOOKUP_CODE	Code associated with the value.
LOOKUP_TYPE	Value lookup type.
MEANING	Meaning associated with the 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

The prefix for these tokens is WF.

**Table A-44. Workflow (WF) tokens**

Tokens	Description
CREATED_BY	ID of the user who created the workflow.
CREATION_DATE	Date on which the workflow was created.
DESCRIPTION	Workflow description.
ENABLED_FLAG	Flag that indicates whether the workflow is enabled for use in packages and/or requests.
FIRST_WORKFLOW_STEP_ID	ID of the first workflow step in the workflow.
FIRST_WORKFLOW_STEP_NAME	Name of the first workflow step in the workflow.
ICON_NAME	Name of the workflow step icon.
LAST_UPDATED_BY	ID of the user who last updated the workflow.
LAST_UPDATE_DATE	Date the workflow was last updated.

**Table A-44. Workflow (WF) tokens , continued**

Tokens	Description
PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow.
REOPEN_WORKFLOW_STEP_ID	ID of the reopened workflow step.
REOPEN_WORKFLOW_STEP_NAME	Name of the reopened workflow step.
SUBWORKFLOW_FLAG	Specifies whether this workflow can be used as a subworkflow.
WORKFLOW_ID	ID of the workflow defined in the table KWFL_WORKFLOWS.
WORKFLOW_NAME	Name of the workflow.
WORKBENCH_WORKFLOW_URL	URL to open the workflow in the PPM Workbench.

## Workflow > Workflow Step Tokens

The prefix for these tokens is WF.WFS.

**Table A-45. Workflow > Workflow Step (WF.WFS) tokens**

Tokens	Description
ACTION_BUTTON_LABEL	Label displayed on the package or request action button for the workflow step.
AVERAGE_LEAD_TIME	Average lead time in days defined for the workflow step.
CREATED_BY	ID of the user who created the workflow step.
CREATION_DATE	Date the workflow step was created.
DESCRIPTION	Workflow step description.
DEST_ENV_GROUP_ID	ID of the destination environment group for the workflow step.
DEST_ENV_GROUP_NAME	Name of the destination environment group for the workflow step.
DEST_ENVIRONMENT_ID	ID of destination environment for the workflow step.
DEST_ENVIRONMENT_NAME	Name of the destination environment for the workflow step.
ENABLED_FLAG	Flag that indicates whether the workflow step is enabled and can be traversed in a package or request.
GL_ARCHIVE_FLAG	For GL object migration, a flag that indicates whether to save the GL

**Table A-45. Workflow > Workflow Step (WF.WFS) tokens , continued**

<b>Tokens</b>	<b>Description</b>
	object being migrated to the GL Migrator archive.
INFORMATION_URL	Workflow step information URL.
JUMP_RECEIVE_LABEL_CODE	Code for a Jump/Receive workflow step.
JUMP_RECEIVE_LABEL_NAME	Name of a Jump/Receive workflow step.
LAST_UPDATED_BY	ID of the user who last updated the workflow step.
LAST_UPDATE_DATE	Date the workflow step was last updated.
OM_ARCHIVE_FLAG	For AOL object migration, a flag that indicates whether to save the AOL object being migrated to the Object*Migrator archive.
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).
PARENT_ASSIGNED_TO_GROUP_NAME	Security group that the current package or request is assigned to (determined by context at time of evaluation).
PARENT_ASSIGNED_TO_USERNAME	Name of the user to whom the current package or request is assigned (determined by context at time of evaluation).
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).
PARENT_STATUS	Validation value code of the status of the request that is using the workflow step.
PARENT_STATUS_NAME	Validation value meaning of the status of the request that is using the workflow step.
PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow containing the workflow step.
RESULT_WORKFLOW_PARAMETER_ID	ID of the workflow parameter to which the result of the workflow step is written.
RESULT_WORKFLOW_PARAMETER_NAME	Name of the workflow parameter to which the result of the workflow step is written.
SORT_ORDER	Display sequence of the workflow step relative to all other steps in the workflow.
SOURCE_ENV_GROUP_ID	ID of the source environment group for the workflow step.

**Table A-45. Workflow > Workflow Step (WF.WFS) tokens , continued**

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

## Workflow Step Tokens

The prefix for these tokens is WFS.

**Table A-46. Workflow Step (WFS) tokens**

Tokens	Description
ACTION_BUTTON_LABEL	Label displayed on the package or request action button for the workflow step.
AVERAGE_LEAD_TIME	Average lead time in days defined for the workflow step.
CREATED_BY	ID of the user who created the workflow step.
CREATION_DATE	Date the workflow step was created.
DESCRIPTION	Description of the workflow step.
DEST_ENV_GROUP_ID	ID of the destination environment group for the workflow step.
DEST_ENV_GROUP_NAME	Name of the destination environment group for the workflow step.
DEST_ENVIRONMENT_ID	ID of destination environment for the workflow step.

**Table A-46. Workflow Step (WFS) tokens , continued**

<b>Tokens</b>	<b>Description</b>
DEST_ENVIRONMENT_NAME	Name of the destination environment for the workflow step.
ENABLED_FLAG	Flag that indicates whether the workflow step is enabled and can be traversed in a package or request.
GL_ARCHIVE_FLAG	For GL object migration, a flag that indicates whether to save the GL object being migrated to the GL Migrator archive.
INFORMATION_URL	Workflow step information URL.
JUMP_RECEIVE_LABEL_CODE	Code for a Jump/Receive workflow step.
JUMP_RECEIVE_LABEL_NAME	Name of a Jump/Receive workflow step.
LAST_UPDATED_BY	ID of the user who last updated the workflow step.
LAST_UPDATE_DATE	Date the workflow step was last updated.
OM_ARCHIVE_FLAG	For AOL object migration, the flag that indicates whether to save the AOL object being migrated to the Object*Migrator archive.
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).
PARENT_ASSIGNED_TO_GROUP_NAME	Security group to which the current package or request is assigned to (determined by context at time of evaluation).
PARENT_ASSIGNED_TO_USERNAME	Name of the user to whom the current package or request is assigned (determined by context at time of evaluation).
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).
PARENT_STATUS	Validation value code of the status of the request that is using the workflow step.
PARENT_STATUS_NAME	Validation value meaning of the status of the request that is using the workflow step.
PRODUCT_SCOPE_CODE	Validation value code for the product scope of the workflow containing the workflow step.
RESULT_WORKFLOW_PARAMETER_ID	ID of the workflow parameter to which the result of the workflow step is written.
RESULT_WORKFLOW_PARAMETER_NAME	Name of the workflow parameter to which the result of the workflow step is written.

**Table A-46. Workflow Step (WFS) tokens , continued**

<b>Tokens</b>	<b>Description</b>
<code>SORT_ORDER</code>	Display sequence of the workflow step relative to all other steps in the workflow.
<code>SOURCE_ENV_GROUP_ID</code>	ID of the source environment group for the workflow step.
<code>SOURCE_ENV_GROUP_NAME</code>	Name of the source environment group for the workflow step.
<code>SOURCE_ENVIRONMENT_ID</code>	ID of the source environment for the workflow step.
<code>SOURCE_ENVIRONMENT_NAME</code>	Name of the source environment for the workflow step.
<code>STEP_NAME</code>	W workflow step name.
<code>STEP_NO</code>	Display sequence of the workflow step relative to all other steps in the workflow.
<code>STEP_SOURCE_NAME</code>	Name of the workflow step source.
<code>STEP_TYPE_NAME</code>	Name of the workflow step source type.
<code>WORKFLOW_ID</code>	ID of the workflow containing the workflow step.
<code>WORKFLOW_NAME</code>	Name of the workflow containing the workflow step.
<code>WORKFLOW_STEP_ID</code>	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

The prefix for these tokens is REQ.P.

**Table A-47. CMDB Application (REQ.P) tokens**

<b>Tokens</b>	<b>Description</b>
<code>KNTA_CMDB_APPLICATION</code>	CMDB application referenced by the request.

## Demand Management SLA Tokens

The prefix for these tokens is REQ.P.

**Table A-48. Demand Management SLA (REQ.P) tokens**

Tokens	Description
KNTA_SLA_LEVEL	SLA level.
KNTA_SLA_VIOLATION_DATE	SLA violation date.
KNTA_SLA_SERV_REQUESTED_ON	Service request date.
KNTA_SLA_SERV_SATISFIED_ON	Service satisfied date.

## Demand Management Scheduling Tokens

The prefix for these tokens is REQ.P.

**Table A-49. Demand Management Scheduling (REQ.P) tokens**

Tokens	Description
KNTA_EST_START_DATE	Estimated start date.
KNTA_EFFORT	Estimated effort.
KNTA_REJECTED_DATE	Reject date.
KNTA_DEMAND_SATISFIED_DATE	Demand satisfied date.

## MAM Impact Analysis Tokens

The prefix for these tokens is REQ.P.

**Table A-50. MAM Impact Analysis (REQ.P) tokens**

Tokens	Description
KNTA_MAM_RFC_ID	MAM RFC ID number.
KNTA_MAM_IMPACT_RESULT	MAM impact results.

## Portfolio Management Asset Tokens

The prefix for these tokens is REQ.P.

**Table A-51. Portfolio Management Asset (REQ.P) tokens**

Tokens	Description
KNTA_ASSET_DEPENDENCIES	Asset Dependencies.
KNTA_BUSINESS_UNIT	Business Unit.
KNTA_PROJECT_NAME	Asset Name.
KNTA_PROJECT_HEALTH	Asset Health.
KNTA_PROJECT_CLASS	Project Class.
KNTA_ASSET_CLASS	Asset Class.
KNTA_BUSINESS_OBJECTIVE	Business Objective.
KNTA_PROJECT_MANAGER	Project Manager.
KNTA_PROJECT_PLAN	Work Plan.
KNTA_BUDGET	Budget.
KNTA_FINANCIAL_BENEFIT	Financial Benefit.
KNTA_STAFFING_PROFILE	Staffing Profile.
KNTA_NPV	Net Present Value.
KNTA_VALUE_RATING	Value Rating.
KNTA_RISK_RATING	Risk Rating.
KNTA_ROI	Return on Investment.
KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
KNTA_TOTAL_SCORE	Total Score.
KNTA_DISCOUNT_RATE	Discount Rate.

## Portfolio Management Project Tokens

The prefix for these tokens is REQ.P.



**Table A-52. Portfolio Management Project (REQ.P) tokens**

<b>Tokens</b>	<b>Description</b>
KNTA_BUSINESS_UNIT	Business Unit.
KNTA_PROJECT_DEPENDENCIES	Project Dependencies.
KNTA_PROJECT_NAME	Project Name.
KNTA_PROJECT_HEALTH	Project Health.
KNTA_PROJECT_CLASS	Project Class.
KNTA_ASSET_CLASS	Asset Class.
KNTA_BUSINESS_OBJECTIVE	Business Objective.
KNTA_PROJECT_PLAN	Work Plan.
KNTA_PROJECT_MANAGER	Project Manager.
KNTA_BUDGET	Budget.
KNTA_FINANCIAL_BENEFIT	Financial Benefit.
KNTA_STAFFING_PROFILE	Staffing Profile.
KNTA_NPV	Net Present Value.
KNTA_VALUE_RATING	Value Rating.
KNTA_RISK_RATING	Risk Rating.
KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
KNTA_ROI	Return on Investment.
KNTA_TOTAL_SCORE	Total Score.
KNTA_DISCOUNT_RATE	Discount Rate.
KNTA_PLAN_START_DATE	Start Date.
KNTA_PLAN_FINISH_DATE	Finish Date.

## Portfolio Management Proposal Tokens

The prefix for these tokens is REQ.P.

**Table A-53. Portfolio Management Proposal (REQ.P) tokens**

<b>Tokens</b>	<b>Description</b>
KNTA_BUSINESS_UNIT	Business Unit.
KNTA_PROJECT_NAME	Project Name.
KNTA_PROJECT_CLASS	Project Class.
KNTA_ASSET_CLASS	Asset Class.
KNTA_BUSINESS_OBJECTIVE	Business Objective.
KNTA_PROJECT_PLAN	Project Plan.
KNTA_PROJECT_DEPENDENCIES	Project Dependencies.
KNTA_PROJECT_MANAGER	Project Manager.
KNTA_PROJECT_TYPE	Project Type.
KNTA_BUDGET	Budget.
KNTA_FINANCIAL_BENEFIT	Expected Benefit.
KNTA_STAFFING_PROFILE	Staffing Profile.
KNTA_NET_PRESENT_VALUE	Net Present Value.
KNTA_VALUE_RATING	Value Rating.
KNTA_RISK_RATING	Risk Rating.
KNTA_RETURN_ON_INVESTMENT	Return on Investment.
KNTA_CUSTOM_FIELD_VALUE	Custom Field Value.
KNTA_TOTAL_SCORE	Total Score.
KNTA_DISCOUNT_RATE	Discount Rate.
KNTA_PLAN_START_DATE	Start Date.
KNTA_PLAN_FINISH_DATE	Finish Date.

## Program Issue Tokens

Within token builder, Program Issue is an empty folder.

## Program Reference Tokens

The prefix for these tokens is REQ.P.

**Table A-54. Program Reference (REQ.P) tokens**

Tokens	Description
KNTA_PROGRAM_REFERENCE	Program reference.

## Project Issue Tokens

The prefix for these tokens is non-app.

**Table A-55. Project Issue (non-app) tokens**

Tokens	Description
KNTA_ESCALATION_LEVEL	Escalation level.

## Project Reference Tokens

The prefix for these tokens is REQ.P.

**Table A-56. Project Issue (REQ.P) tokens**

Tokens	Description
KNTA_MASTER_PROJ_REF	Master project reference.

## Project Risk Tokens

The prefix for these tokens is REQ.P.

**Table A-57. Project Issue (REQ.P) tokens**

Tokens	Description
KNTA_IMPACT_LEVEL	Impact level.
KNTA_PROBABILITY	Probability level.

## Project Scope Change Tokens

The prefix for these tokens is non-app.

**Table A-58. Project Scope Change (non-app) tokens**

Tokens	Description
KNTA_CR_LEVEL	Critical level.
KNTA_IMPACT_LEVEL	Impact level.

## Quality Center Defect Information Tokens

The prefix for these tokens is REQ.P.

**Table A-59. Quality Center Defect Information (REQ.P) tokens**

Tokens	Description
KNTA_QC_DEECT_DOMAIN	Quality Center domain name.
KNTA_QC_DEFECT_PROJECT	Quality Center project.
KNTA_QC_DEFECT_ASSIGNED_TO	Quality Center defect assigned to.
KNTA_QC_DEFECT_NO	Quality Center defect number.
KNTA_QC_DEFECT_STATUS	Quality Center defect status.
KNTA_QC_DEFECT_ATT_URL	Quality Center defect ATT URL.
KNTA_QC_DEFECT_INT_MSG	Quality Center defect instant message.
KNTA_QC_DEFECT_INSTANCE	Quality Center defect instance.

## Quality Center Information Tokens

The prefix for these tokens is REQ.P.

**Table A-60. Quality Center Information (REQ.P) tokens**

Tokens	Description
KNTA_QC_DOMAIN	Quality Center domain name.

**Table A-60. Quality Center Information (REQ.P) tokens , continued**

<b>Tokens</b>	<b>Description</b>
KNTA_QC_ASSIGNED_TO	Quality Center assigned to user.
KNTA_QC_PROJECT	Quality Center project.
KNTA_QC_REQUIREMENT_NO	Quality Center requirement number.
KNTA_QC_REQUIREMENT_STATUS	Quality Center requirement status.
KNTA_QC_REQUIREMENT_ATT_URL	Quality Center requirement ATT URL.
KNTA_QC_REQUIREMENT_INT_MSG	Quality Center requirement instant messaging.
KNTA_QC_INSTANCE	Quality Center instance.
KNTA_QC_DASHBOARD_SUBJECT	Quality Center dashboard subject.
KNTA_QC_REQUIREMENT_COVERAGE	Quality Center requirement coverage.
KNTA_QC_OPEN_DEFECTS	Quality Center open defects.

## Resource Management Work Item Tokens

The prefix for these tokens is REQ.P.

**Table A-61. Resource Management Work Item (REQ.P) tokens**

<b>Tokens</b>	<b>Description</b>
KNTA_USR_SCHED_START_DATE	Scheduled Start Date
KNTA_USR_ACTUAL_START_DATE	Actual Start Date
KNTA_USR_SCHED_FINISH_DATE	Scheduled Finish Date
KNTA_USR_ACTUAL_FINISH_DATE	Actual Finish Date
KNTA_SCHED_DURATION	Scheduled Duration
KNTA_ACTUAL_DURATION	Actual Duration
KNTA_SCHED_EFFORT	Scheduled Effort
KNTA_ACTUAL_EFFORT	Actual Effort
KNTA_WORKLOAD	Workload
KNTA_WORKLOAD_CATEGORY	Workload Category

**Table A-61. Resource Management Work Item (REQ.P) tokens , continued**

<b>Tokens</b>	<b>Description</b>
KNTA_ROLE	Role
KNTA_SCHED_START_DATE	Scheduled Start Date
KNTA_ACTUAL_START_DATE	Actual Start Date
KNTA_SCHED_FINISH_DATE	Scheduled Finish Date
KNTA_ACTUAL_FINISH_DATE	Actual Finish Date
KNTA_SCHED_EFF_OVER_DUR	Scheduled Effort Over Duration

## Send Documentation Feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

**Feedback on Commands, Tokens, and Validations Guide and Reference (Project and Portfolio Management Center 9.30)**

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to [HPSW-BTO-PPM-SHIE@hp.com](mailto:HPSW-BTO-PPM-SHIE@hp.com).

We appreciate your feedback!