

Mercury IT Governance Center™

**Commands, Tokens, and Validations
Guide and Reference**

Version: 7.0



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

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

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

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

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

Mercury
379 North Whisman Road
Mountain View, CA 94043
<http://www.mercury.com>

© 1997–2006 Mercury Interactive Corporation. All rights reserved.

If you have any comments or suggestions regarding this document, please send email to documentation@mercury.com.

Table of Contents

List of Figures	ix
List of Tables	xi
Chapter 1: Getting Started with Commands, Tokens, and Validations.....	15
Introduction to Commands, Tokens, and Validations.....	16
Related Information.....	17
Chapter 2: Using Commands.....	19
About Commands	20
Object Type Commands and Workflows.....	20
Request Type Commands and Workflows	21
Special Commands	22
Command Language.....	22
Command Conditions.....	23
About the Commands Tab	24
Configuring Commands	25
Examples of Command Uses	28
Chapter 3: Using Special Commands	31
About Special Commands	32
Special Command Parameters	32
Special Command Language.....	33
Special Command Conditions.....	34
About the Special Command Builder	35
Configuring Special Commands.....	35
Using Special Commands.....	41

Using the Special Command Builder	42
Nesting Special Commands	43
Listing all of the Special Commands	44
Examples of Using Special Commands	45
Chapter 4: Using Tokens	47
About Tokens	48
Where to Use Tokens	48
Token Evaluation	49
About Token Builder	50
Token Formats	51
Default Format	54
Explicit Entity Format	55
Nesting Explicit Entity Tokens within Other Tokens	56
User Data Format	57
Parameter Format	58
Request Field Tokens	59
Request Token Prefixes	59
Tokens in Request Table Components	59
Sub-Entity Format	62
Environment and Environment Application Tokens	63
Using Token Builder	65
Chapter 5: Using Validations	67
About Validations	68
Validation Component Types	69
Accessing Validations Through Packages and Requests	71
Validations and Special Characters	73
Viewing System Validations	73
Configuring Validations	74
Configuring Static List Validations	77
Configuring Dynamic List Validations	79
Configuring SQL Validations	79
SQL Validation Tips	81
Command Validations	82
Configuring Short List Auto-Complete Field Validations	82
Configuring Long List Auto-Complete Field Validations	84
Configuring Automatic Value Matching and Interactive Select Pages	85
An Overview of Matching for “Starts with” or “Contains”	86
Configuration Tips	88
Adding Search Fields to Long List Auto-Complete Validations	89
Configuring the Filter Field Layout	92
Configuring an Auto-Complete List of Users (Special Case)	94

Configuring the Auto-Complete Values	94
Configuring Validations by Commands With Delimited Output	95
Configuring Validations by Commands with Fixed Width Output	97
Configuring User-Defined Multi-Select Auto-Complete Fields	99
Example of Token Evaluation and Validation by Command with Delimited Output ...	101
Configuring Text Field Validations	104
Text Data Masks for Validations	105
Configuring the Numeric Data Mask	107
Configuring the Currency Data Mask	108
Configuring the Percentage Data Mask	111
Configuring the Telephone Data Mask	112
Configuring a Custom Data Mask	114
Configuring Directory Chooser Validations	115
Configuring File Chooser Validations	116
Configuring Date Field Validations	118
Configuring 1800 Character Text Areas	120
Configuring the Table Validations	120
Configuring Table Components	121
Configuring Table Rules	125
Example of Using a Table Component on an Order Form	125
Example of Setting Unit Prices	127
Example of Calculating Totals	128
Using Table Components	129
Using Tokens in Table Components	129
Calculating Column Totals	129
Appendix A: Tokens	133
Overview of Tokens	135
Application Server Tokens	135
Budget Tokens	135
Contact Tokens	137
Distribution Tokens	138
Document Management Tokens	139
Environment Tokens	140
Environment > Dest Env Tokens	140
Environment > Dest Env > App Tokens	143
Environment > Dest Env > Env Tokens	145
Environment > Env Tokens	148
Environment > Env > App Tokens	151
Environment > Env > Env Tokens	153
Environment > Source Env Tokens	156

Environment > Source Env > App Tokens	159
Environment > Source Env > Env Tokens	161
Command Tokens	164
Financial Benefit Tokens.....	165
Notification Tokens.....	166
Organization Unit Tokens.....	167
Package Tokens	168
Package > Package Line Tokens	170
Package > Pending Reference Tokens.....	171
Package Line Tokens	173
Program Tokens	174
Project Tokens	174
Project Detail Tokens	178
Release Tokens	178
Release > Distribution Tokens.....	179
Request Tokens.....	180
Request > Pending Reference Tokens.....	184
Request > Field Tokens.....	185
Request Detail Tokens.....	185
Request Detail > Field Tokens.....	186
Resource Pool Tokens	186
Security Group Tokens.....	187
Skill Tokens.....	188
Staffing Profile Tokens.....	188
Step TXN (Transaction) Tokens.....	189
System Tokens.....	191
Task Tokens.....	191
Tasks > Pending Tokens	194
Time Management Notification Tokens.....	196
User Tokens.....	196
Validation Tokens	198
Validation > Value Tokens	199
Workflow Tokens	200
Workflow > Workflow Step Tokens.....	201
Workflow Step Tokens.....	203
Request > Field Tokens	206
CMBD Application Tokens.....	206

Demand Management SLA Tokens.....	207
Demand Management Scheduling Tokens.....	207
MAM Impact Analysis Tokens.....	207
Portfolio Management Asset Tokens.....	208
Portfolio Management Project Tokens.....	209
Portfolio Management Proposal Tokens.....	210
Program Issue Tokens.....	211
Program Reference Tokens.....	211
Project Issue Tokens.....	211
Project Reference Tokens.....	211
Project Risk Tokens.....	211
Project Scope Change Tokens.....	212
Quality Center Defect Information Tokens.....	212
Quality Center Information Tokens.....	212
Resource Management Work Item Tokens.....	213
Service Catalog Tokens.....	214
Index	215

List of Figures

Figure 2-1	Commands tab	24
Figure 3-1	Special Command Builder	35
Figure 3-2	RCS File Migration object type	41
Figure 4-1	Example of a token used in a SQL statement	49
Figure 4-2	Token Builder window	51
Figure 4-3	Table component formats.....	60
Figure 5-1	Validation window	72
Figure 5-2	Validation window	72
Figure 5-3	Auto-complete using command validation	82
Figure 5-4	Short list auto-complete	83
Figure 5-5	Long list auto-complete	84
Figure 5-6	Auto-complete field and matching values on the Select page.....	86
Figure 5-7	Filter fields in the auto-complete select window.....	89
Figure 5-8	Auto-Complete List	95
Figure 5-9	Validation by command with delimited output	96
Figure 5-10	Validation by command with fixed width output	98
Figure 5-11	Validation window for the numeric data mask	107
Figure 5-12	Validation window for the currency data mask.....	109
Figure 5-13	Validation window for the percentage data mask	111
Figure 5-14	Validation window for the telephone data mask	113
Figure 5-15	Validation window for the custom data mask.....	114
Figure 5-16	Validation window for static environment override in file chooser.....	116
Figure 5-17	Validation window for token-based environment override in file chooser.	117

List of Figures

Figure 5-18	Hardware information window.....	121
Figure 5-19	Rules window accessed from the Rules tab.....	125
Figure 5-20	Validations window.....	126
Figure 5-21	Rules window.....	128
Figure 5-22	Hardware information window.....	129
Figure 5-23	Sample validation for a Simple Order table component.....	130
Figure 5-24	Sample table component displaying a column total.....	131

List of Tables

Table 2-1	Example Conditions	23
Table 3-1	Example Conditions	34
Table 4-1	Entities.....	52
Table 4-2	Sample environment and application attributes.....	64
Table 4-3	Sample environment tokens.....	64
Table 5-1	Component Types.....	69
Table 5-2	Column Headers	80
Table 5-3	Automatic character matching field behavior.....	86
Table 5-4	Automatic character matching Select page behavior	87
Table 5-5	Fields in the Fields: New window	91
Table 5-6	Validation by command with delimited output.....	97
Table 5-7	Column headers	97
Table 5-8	Validation by command with fixed width output	98
Table 5-9	Column headers	99
Table 5-10	Data Mask Formats	105
Table 5-11	Fields for configuring the numeric data mask for text fields	108
Table 5-12	Fields configuring the currency data mask for text fields	109
Table 5-13	Fields configuring the percentage data mask for text fields.....	112
Table 5-14	Fields configuring the telephone data mask for text fields.....	113
Table 5-15	Sample telephone data mask formats.....	113
Table 5-16	Sample custom data mask formats.....	115
Table 5-17	File chooser field	116

List of Tables

Table 5-18	Static environment override	117
Table 5-19	Token-based environment override.....	118
Table 5-20	Date field.....	119
Table 5-21	Example, table component validation settings	126
Table 5-22	Example - Set Unit Price rule settings.....	127
Table 5-23	Example - Calculate Total rule settings.....	128
Table A-1	Application Server tokens.....	135
Table A-2	Budget tokens.....	135
Table A-3	Contact tokens	137
Table A-4	Distribution tokens.....	138
Table A-5	Document Management tokens	139
Table A-6	Environment > Dest Env tokens.....	140
Table A-7	Environment > Dest Env > App tokens	143
Table A-8	Environment > Dest Env > Env tokens	145
Table A-9	Environment > Env tokens	148
Table A-10	Environment > Env > App tokens.....	151
Table A-11	Environment > Env > Env tokens	153
Table A-12	Environment > Source Env tokens	156
Table A-13	Environment > Source Env > App tokens.....	159
Table A-14	Environment Source Env > Env tokens	161
Table A-15	Command tokens.....	164
Table A-16	Financial Benefit tokens.....	165
Table A-17	Notification tokens.....	166
Table A-18	Organization Unit tokens.....	167
Table A-19	Package tokens.....	168
Table A-20	Package > Package Line tokens.....	170
Table A-21	Package > Pending Reference tokens	171
Table A-22	Package Line tokens	173
Table A-23	Program tokens	174
Table A-24	Project tokens	174
Table A-25	Project Detail tokens	178
Table A-26	Release tokens	178
Table A-27	Release > Distribution tokens	179
Table A-28	Request tokens.....	180

Table A-29	Request > Pending Reference tokens.....	184
Table A-30	Request Detail tokens.....	185
Table A-31	Resource Pool tokens.....	186
Table A-32	Security Group tokens.....	187
Table A-33	Skill tokens.....	188
Table A-34	Staffing Profile tokens.....	188
Table A-35	Step TXN (Transaction) tokens.....	189
Table A-36	System tokens	191
Table A-37	Tasks tokens	191
Table A-38	Tasks > Pending tokens.....	194
Table A-39	Time Management Notification tokens.....	196
Table A-40	User tokens.....	196
Table A-41	Validation tokens	198
Table A-42	Validation > Value tokens.....	199
Table A-43	Workflow tokens.....	200
Table A-44	Workflow > Workflow Step tokens.....	201
Table A-45	Workflow Step tokens.....	203
Table A-46	CMBD Application tokens.....	206
Table A-47	Demand Management SLA tokens	207
Table A-48	Demand Management Scheduling tokens.....	207
Table A-49	MAM Impact Analysis tokens	207
Table A-50	Portfolio Management Asset tokens.....	208
Table A-51	Portfolio Management Project tokens.....	209
Table A-52	Portfolio Management Proposal tokens.....	210
Table A-53	Program Reference tokens	211
Table A-54	Project Issue tokens.....	211
Table A-55	Project Issue tokens.....	211
Table A-56	Project Issue tokens.....	211
Table A-57	Project Scope Change tokens.....	212
Table A-58	Quality Center Defect Information tokens.....	212
Table A-59	Quality Center Information tokens.....	212
Table A-60	Resource Management Work Item tokens.....	213
Table A-61	Service Catalog tokens	214

Chapter

1

Getting Started with Commands, Tokens, and Validations

In This Chapter:

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

Introduction to Commands, Tokens, and Validations

Commands, tokens, and validations are used throughout the Mercury IT Governance Center™ implementation to enable advanced automation and defaulting.

Commands are at the heart of the execution layer within the deployment system. They determine which actions are executed at specific workflow steps. Actions performed at a workflow step can include file migration, script execution, data analysis, or code compilation. [Chapter 2, Using Commands, on page 19](#) provides an overview of commands, and examples of how to use them.

Special commands are commands with variable parameters and are used in object types, request types, report types, workflows, and validation command steps. (Workflows use special commands in their workflow step sources.) These command steps perform a variety of functions, such as copying files between environments and establishing connections to environments for remote command execution. [Chapter 3, Using Special Commands, on page 31](#) contains information about how to create, edit, and use special commands in Mercury IT Governance Center.

Tokens are variables that Mercury IT Governance 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.

Mercury IT Governance Center uses two types of tokens: standard tokens and custom tokens. [Chapter 4, Using Tokens, on page 47](#) shows how to use tokens.

Validations determine the 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 field component type and workflow step results. [Chapter 5, Using Validations, on page 67](#) provides detailed information on how to use tokens.



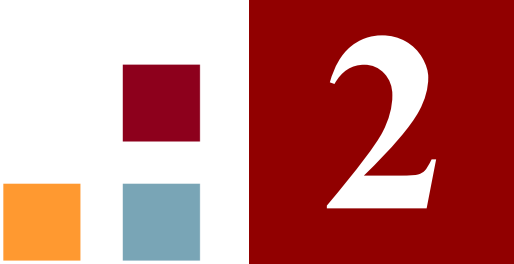
Note

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

Related Information

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

- *Mercury Demand Management Configuration Guide*
- *Mercury Deployment Management Configuration Guide*
- *Configuring the Standard Interface*



Chapter
2
Using Commands

In This Chapter:

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

About Commands

Commands are at the heart of the execution layer within Mercury IT Governance 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 Mercury IT Governance 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 Mercury Deployment Management™ package lines.

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

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

- You can configure the object type to run only certain commands at a given step. To do this, specify command conditions. For information about how to specify command conditions, see [Command Conditions on page 23](#).

Request Type Commands and Workflows

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

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

- To execute request type commands at a given workflow step, configure the workflow step as follows:
 - The workflow step must be an execution type step
 - Set the following parameter values:
 - Workflow Scope = Requests
 - Execution Type = Built-in Workflow Event
 - Workflow Command = `execute_request_commands`
- When the request reaches the workflow step (Workflow Command = `execute_request_commands`), all commands with all conditions satisfied are run in the listed order in which they are listed on the command field for the request type.
- To set up command conditions so that the request type runs only certain commands at a given step, specify command conditions. For information about how to specify command conditions, see [Command Conditions on page 23](#).

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, Mercury IT Governance 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.

Mercury IT Governance Center features the following two types of special commands:

- System special commands are shipped with Mercury IT Governance 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 [Chapter 3, Using Special Commands](#), on page 31.

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 Mercury IT Governance Center.

Mercury IT Governance 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 [Chapter 3, Using Special Commands](#), on page 31.

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 a SQL statement. It allows for enormous flexibility in evaluating scenarios. [Table 2-1 on page 23](#) lists some example conditions. The condition can include tokens. For more information, see [Chapter 4, Using Tokens, on page 47](#).

Table 2-1. Example Conditions

Condition ^a	Evaluates to
BLANK	Command is executed in all situations.
'[P.P_VERSION_LABEL]' IS NOT NULL	Command is executed if the parameter with the token P_VERSION_LABEL in the package line is not null.
'[DEST_ENV.ENVIRONMENT_NAME]' = 'Archive'	Command is executed when the destination environment is named Archive.
'[AS.SERVER_TYPE_CODE]' = 'UNIX'	Command is executed if the application server is installed on a UNIX machine.
a. You must place single quotes around string literals or tokens that are used to evaluate strings.	

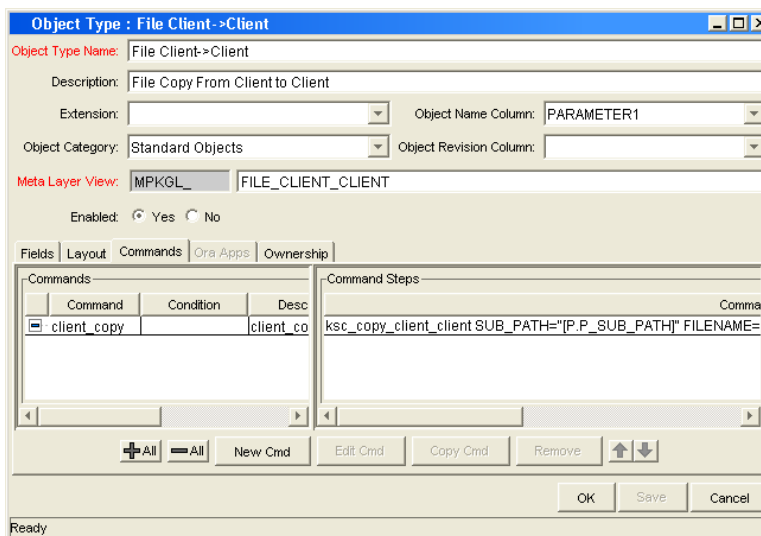
About the Commands Tab

Within Mercury IT Governance Center, commands are configured using the **Commands** tab of various Mercury IT Governance Center entities. These entities include:

- Object Types
- Request Types
- Report Types
- Validations
- Workflow Step Sources
- Special Commands

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

Figure 2-1. Commands tab



Commands tabs are divided into two parts:

- **Commands.** Commands defines the command-line directive or special command to be issued.
- **Command Steps.** Command steps represent the actual directives that Mercury IT Governance Center specifies to execute the commands. A command step can be an actual command-line directive that is sent to the Mercury IT Governance Server or target machine, or it can be one of the many special commands.



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

- On the **Commands** tab, click the command or command step, and then use the up and down pointers to change the order 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. Defining this events requires configuring commands using the **Commands** tab. The following Mercury IT Governance 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, the commands are accessed through the Mercury Deployment Management Object Type window. However, the controls are the same in the other windows that you can use to configure commands.

To configure commands associated with an object type:

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

The Workbench opens.

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

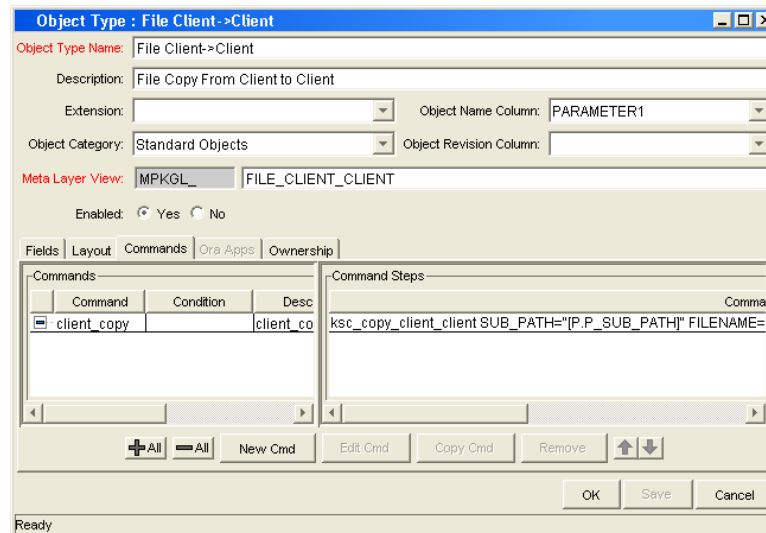
The Object Type Workbench window opens.

4. Open an existing object type or create a new object type.

The Object Type window opens.

5. Click the **Commands** tab.

The Commands tab is opened.



- To remove an existing command, select the command and click **Remove**.

7. Complete the fields of the New Command window as specified in the following table.

Field Name	Description
Command	The command name.
Condition	The specific conditions under which the command steps are to be exclusively executed. This step is optional. For more information, see Command Conditions on page 23 .
Description	The command description. This step is optional. For more information, see Command Conditions on page 23 .
Timeout(s)	The number of minutes to run the command before stopping. The Timeout(s) setting is useful if a command hangs or takes too long to execute.
Steps	Enter at least one command step.
Enable	Yes/No radio buttons. Enables or disables the command. Yes allows the command to be run.

- Click **Tokens** to open the Token Builder window. The Token Builder window allows you to find and add a token to the command step. Tokens are variables used to facilitate the creation of general objects. For more information concerning tokens, see [Chapter 4, Using Tokens, on page 47](#).

- Click **Special Cmds** to open the Special Command Builder. The Special command Builder allows you to find and add a special command to the command step. For more information concerning tokens, see [Chapter 3, Using Special Commands, on page 31](#).
 - Click **Show/Hide Desc** to open (show) or hide (close) a **Descriptions** field in the **Steps** field. When the **Descriptions** field is visible, you can add a description to the command step.
8. Click **OK** to add the command to the **Commands** tab and close the New Command window.
- Click **Add** to add the command to the **Commands** tab and leave open the New Command window.
 - Click **Cancel** to stop all work on the command and close the New Command window.

Examples of Command Uses

This section provides examples of commands.

To copy a file from one environment to another environment:

Command:

```
copy_client_client
```

Command Steps:

```
ksc connect dest client
if [ ! d [P.P_SUB_PATH] ];
then mkdir -p [P.P_SUB_PATH]; fi
ksc exit
ksc_copy_client_client SUB_PATH="[P.P_SUB_PATH]"
FILENAME="[P.P_FILENAME]" FILE_TYPE="[P.P_FILE_TYPE]"
```

To automatically update the staffing profile status to “In Planning:”**Command:**

```
Update Staffing Profile Status
```

Command Steps:

```
ksc set staffing_profile_status USE NAMES FLAG="N"  
STAFF_PROF_IDS="[REQ.P.KNTA_STAFFING_PROFILE]"  
STATUS_NAME="In Planning"
```

To execute Oracle SQL script against an Oracle Database using JDBC:**Command:**

```
Execute SQL
```

Command Steps:

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

To log a program issue using a request type:**Command:**

```
ksc_store
```

Command Steps:

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

To run a report using UNIX:

Command:

Run report.

Command Steps:

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

To run a report using Windows:

Command:

Run report.

Command Steps:

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



Chapter
3

Using Special Commands

In This Chapter:

- *About Special Commands*
 - *Special Command Parameters*
 - *Special Command Language*
 - *Special Command Conditions*
 - *About the Special Command Builder*
 - *Configuring Special Commands*
 - *Using Special Commands*
 - *Using the Special Command Builder*
 - *Nesting Special Commands*
 - *Listing all of the Special Commands*
 - *Examples of Using Special Commands*
-

About Special Commands

Object types, request types, report types, workflows and validations all use commands to access the execution layer. To simplify the use of command executions, Mercury IT Governance 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 a variety of functions, such as copying files between environments and establishing connections to environments for remote command execution. Mercury IT Governance Center features two types of special commands:

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

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

Special Command Parameters

The **Parameters** tab displays the current parameters for the special command. Most special commands have parameters to override standard behavior. Nearly all parameters are optional. When 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 shown 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 35*.



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 Mercury IT Governance Center.

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

Special Command Conditions

Depending on the context in which commands are executed, it may be necessary to run a different set of commands. For example, one command may be needed to update a Web page, while another command may be required to 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 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* provides some example conditions.

Table 3-1. Example Conditions

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



Note

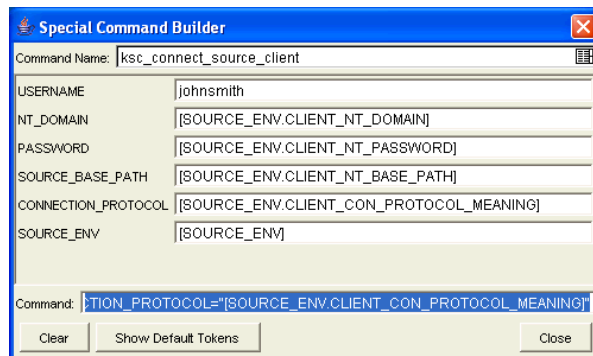
When using conditional commands, you must use single quotes to enclose strings.

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

About the Special Command Builder

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

Figure 3-1. Special Command Builder



Configuring Special Commands

To configure a new special command:

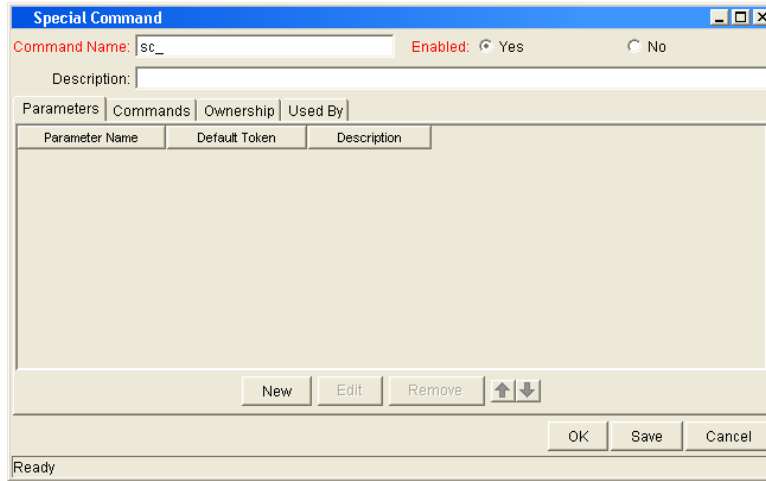
1. Log on to Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
The Workbench opens.
3. From the shortcut bar, select **Configuration > Special Commands**.

The Special Command Workbench opens.

4. From the Special Command Workbench, click **New Special Command**.

- To open an existing special command, select a special command and click **Open**.

The Special Command window opens.

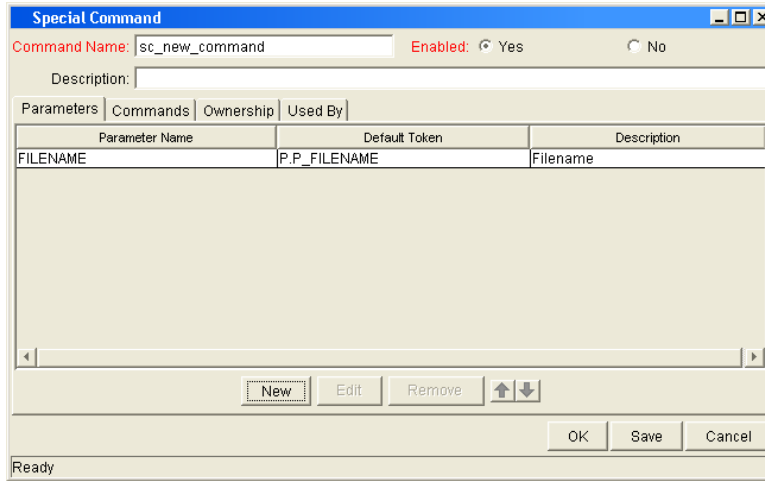


5. Complete the fields in the Special Command window as specified in the following table.

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.

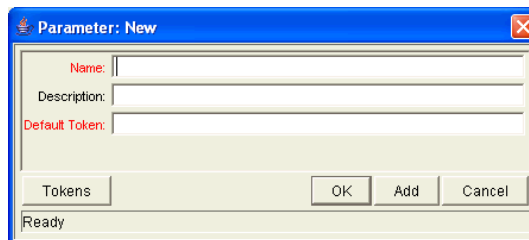
6. Configure the **Parameters** tab.
 - a. Click the **Parameters** tab.

The **Parameters** tab opens.



b. To configure a new parameter, click **New**.

The Parameter window opens.



c. Complete the fields in the Parameter window as specified in the following table.

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

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

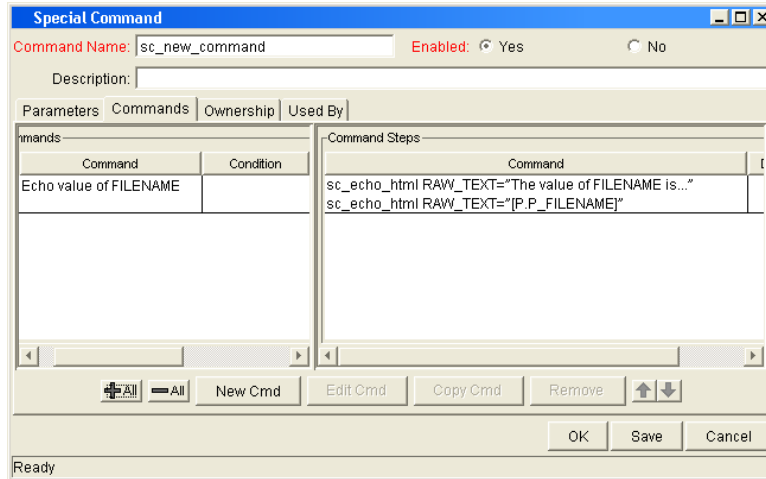
e. Add the field to the **Parameters** tab.

- To add the field and close the Parameters window, click **OK**.
- To add the field and leave open the Parameters window, click **Add**.
- To stop work on the parameter and close the Parameters window, click **Cancel**.

7. Configure the **Commands** tab.

- a. Click the **Commands** tab.

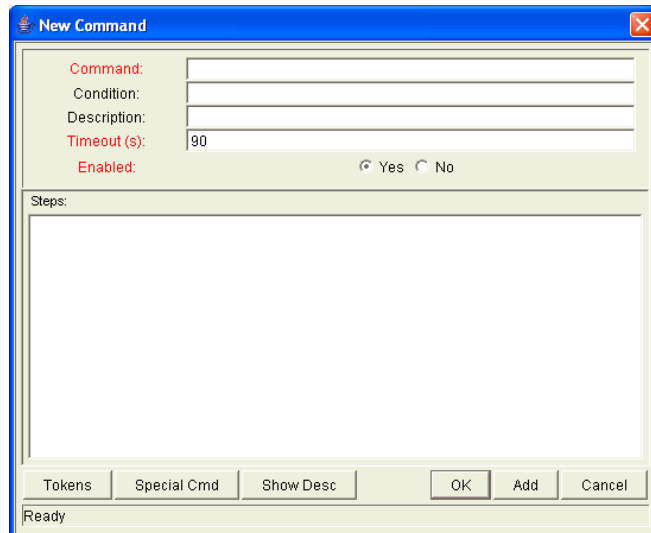
The **Commands** tab opens.



- b. On the **Commands** tab, click **New Cmd**.

- To open an existing command, select the command and click **Edit Cmd**.

The New Command window opens.



- To remove an existing command, select the command and click **Remove**.

- c. Complete the fields of the New Command window as specified in the following table.

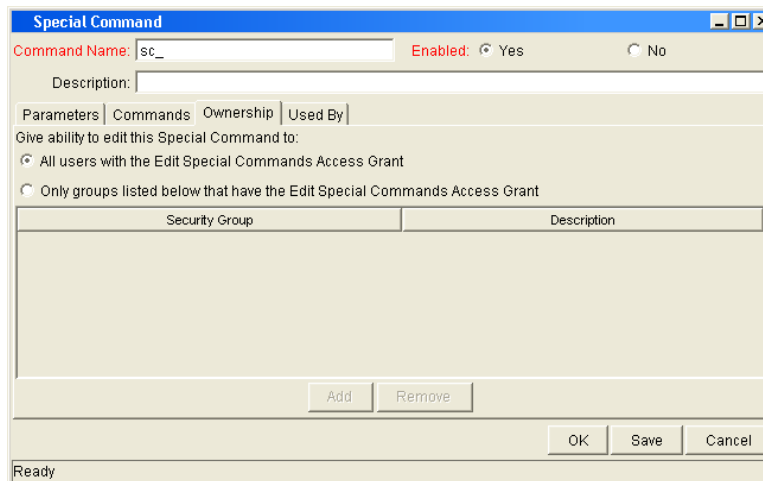
Field Name	Description
Command	The command name.
Condition	The specific conditions under which the command steps are to be exclusively executed. This step is optional. For more information, see Special Command Conditions on page 34 .
Description	The command description. This step is optional. For more information, see Special Command Conditions on page 34 .
Timeout(s)	The number of minutes to run the command before stopping. The Timeout(s) setting is useful if a command hangs or takes too long to execute.
Steps	Enter at least one command step.
Enable	Yes/No radio buttons. Enables or disables the command. Yes allows the command to be run.

- Click **Tokens** to open the Token Builder window. The Token Builder window allows you to find and add a token to the command step. Tokens are variables used to facilitate the creation of general objects. For more information concerning tokens, see [Chapter 4, Using Tokens, on page 47](#).
 - Click **Special Cmds** to open the Special Command Builder. The Special command Builder allows you to find and add a special command to the command step. For more information concerning tokens, see [Chapter 3, Using Special Commands, on page 31](#).
 - Click **Show/Hide Desc** to open (show) or hide (close) a **Descriptions** field in the **Steps** field. When the **Descriptions** field is visible, you can add a description to the command step.
- d. Save the newly configured command.
- Click **OK** to add the command to the **Commands** tab and close the New Command window.
 - Click **Add** to add the command to the **Commands** tab and leave open the New Command window.
 - Click **Cancel** to stop all work on the command and close the New Command window.

8. Configure the **Ownership** tab.

a. To open the **Ownership** tab, click the **Ownership** tab.

The **Ownership** tab opens.



9. Select **Only groups listed below that have the Edit Special Commands Access Grant**.

The **Add** button is enabled.

10. Click **Add**.

The Add Security Groups window opens.

11. In the Security Group window, select the security groups.

12. In the Add security group window, click **OK**.

On the **Ownership** tab, the security groups are listed.

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

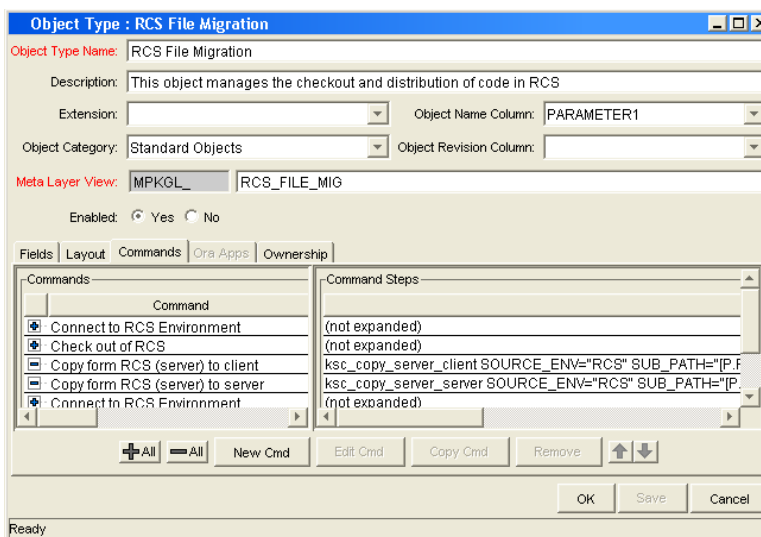
- To save the security group and close the Special Command window, click **OK**.
- To save the security group and leave the Special Command window open, click **Save**.

14. Configure the **Used By** tab.
 - a. To view a list of entities that reference the selected special command, click the **Used By** tab.
15. Save the Special Command.
 - To save the special command and close the Special Command window, click **OK**.
 - To save the special command and leave the Special Command window open, click **Save**.
 - To stop work on the special command and close the Special Command window, click **Cancel**.

Using Special Commands

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

Figure 3-2. RCS File Migration object type



Using the Special Command Builder

Special commands can be added to any set of command steps in the following entities:

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

Access the Special Command Builder in the **Commands** tab for each of these entities.

To build a command step using the Special Command Builder:

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

The Object Type Workbench opens.

2. Open an object type.

The Object Type Window opens.

3. Select the **Commands** tab.

The **Commands** tab is displayed.

4. Click **New Cmd** or **Edit Cmd**.

The Command window opens.

5. Click **Special Cmd**.

The Special Command Builder window opens.

6. In the **Command Name** field, select the special command.

When selecting a command name from the auto-complete, its parameters are listed in the Special Command Builder.



Note

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

7. Replace the associated default token value with parameter information.
 - a. To view the default tokens, click **Show Default Tokens**.
 - b. To hide the default tokens, click **Hide Default Tokens**.
 - c. Copy the text in the Special Command Builder window **Command** field to the Command window **Steps** field.
 - d. Select the text in the Special Command Builder window **Command** field.
 - e. Type **ctrl + c** to copy the text in the **Command** field.
 - f. Click **Close** to close the Special Command Builder window.
 - g. In the Command window, click inside the **Steps** field.
 - h. Paste the copied text by typing **ctrl + v**.
8. Enter information in the remaining fields in the Command window.
9. For the **Enabled** option, click **Yes**.
10. 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

Special commands can be used in an execution workflow step source. After the workflow step source is created (which contains the special commands), it can be dragged and dropped into a workflow.

Nesting Special Commands

Special commands can be used within other special commands, but must be used within a command step. However, a special command cannot refer to itself.

Listing all of the Special Commands

Mercury IT Governance Center comes with several 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 the special command, the parameters of the special command, and where the special command is used.

To view the existing special commands on your instance:

1. Log on to Mercury IT Governance Center.
2. From the menu bar, select **Reports > Create a Report**.
The Reports window appears.
3. In the Report Category, select **Administrative**.
A list of the Administrative reports is displayed.
4. From the list of Administrative reports, select **Special Command Details Report**.
The Special Command Details Report window opens.
5. Complete the fields in the Special Command Details Report window.
 - To view all of the special commands, leave the fields **Special Command From** and **Special Command To** empty.
 - Under Report Parameters, select **Yes** for **Show References**.
6. Click **Submit** and wait for the report to be displayed.

Examples of Using Special Commands

This section provides examples of special commands.

To copy a file from one server to another server:

Special Command Name:

```
copy_server_server
```

Special Command Example:

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

To import using a given password:

Special Command Name:

```
ksc_mig_import
```

Special Command Example:

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

To change the status of a project:

Special Command Name:

```
ksc_run_java
```

Special Command Example:

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

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

Special Command Name:

```
ksc_connect_dest_server
```

Special Command Example:

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



Chapter
4
Using Tokens

In This Chapter:

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

About Tokens

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

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

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

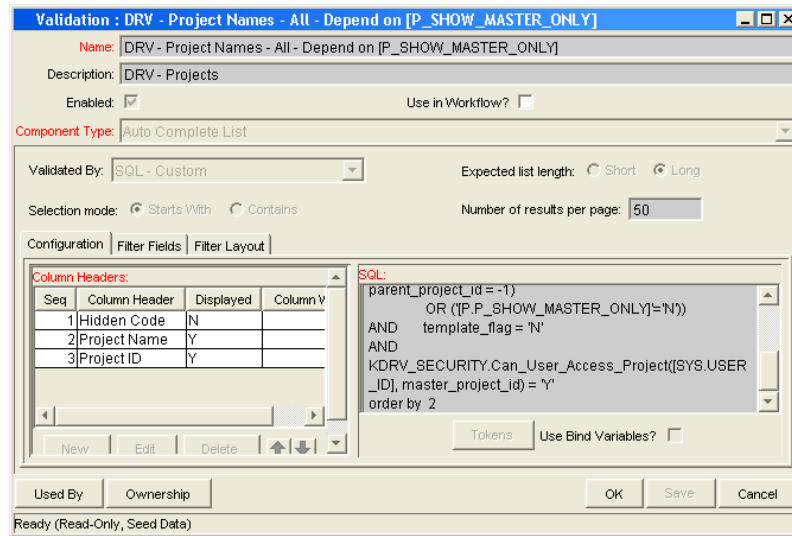
In addition, numerous standard tokens are available that provide other useful pieces of information related to the system. For example, Mercury IT Governance Center has a token that represents the users currently logged onto the system.

Where to Use Tokens

You can use tokens in the following entity windows:

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

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



Token Evaluation

Tokens are evaluated at the point when Mercury IT Governance 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. Tokens that are invalid for any reason (such as the token is misspelled or no context is available) are left alone.

For example, suppose 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
```

At the time of execution, [PKGL.P.P_SUB_PATH] = Forms and [PKGL.P.P_BASE_FILENAME] = obj_maint. After token evaluation, this command step would reduce to:

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

As another example, suppose a user data field has been generated for all users called MANAGER. The email address of the manager of the person who generated a request could be found using the token:

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

The token evaluation engine would first evaluate the innermost token ([REQ.CREATED_BY_NAME]). Once that is complete, the next token ([USR="<name>".VUD.MANAGER]) is evaluated. Finally, the outermost token is evaluated, giving the manager's 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 Token Builder

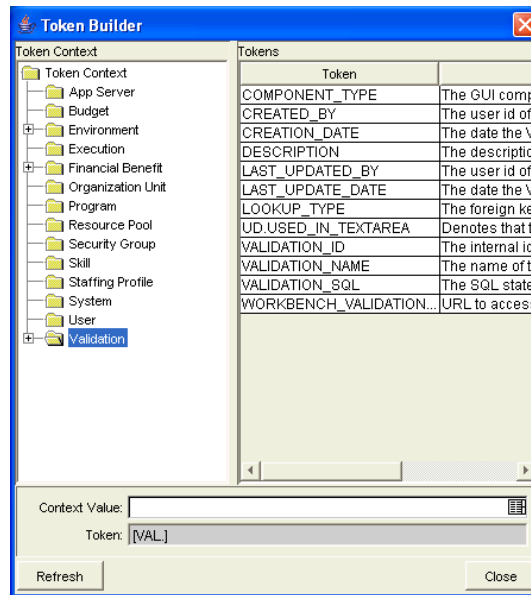
In each of the entity windows listed in *Where to Use Tokens on page 48*, a token can be created by opening the Token Builder window. Note that the available tokens found in the token builder reflect the tokens that can be built for that entity. For example, opening the token builder in the Request Type workbench excludes package tokens.

The folders displayed in the left pane of the Token Builder window contain groups of tokens that correspond to entities defined in Mercury IT Governance Center. For instance, the Packages folder contains tokens that reference various package attributes. If the Packages folder is selected, the available package tokens are displayed in the list in the right pane of the window.

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

As entity folders and the subsequent tokens in the list are selected, 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. Either copy and paste the character string, or type this string where needed.

Figure 4-2. Token Builder window



Token Formats

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

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

Table 4-1 lists the entities and the formats each entity supports. Each format is discussed in a section following the table.

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

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

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

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

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

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

Default Format

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

```
[PREFIX.TOKEN_NAME]
```

For example:

The token for the package number is expressed as:

```
[PKG.NUMBER]
```

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

```
[REQ.WORKFLOW_NAME]
```

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

```
[PREFIX.SUB-PREFIX.TOKEN_NAME]
```

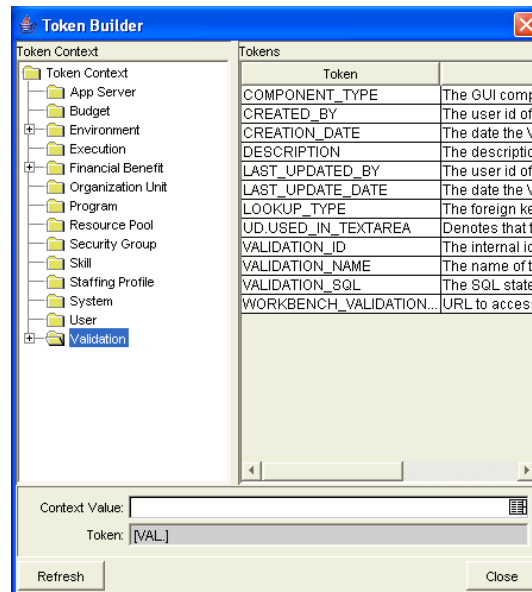
Tokens are evaluated according to the current context of Mercury IT Governance Center, which is derived based on information known at the time of evaluation. For more information, see [Token Evaluation on page 49](#).

Explicit Entity Format

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

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

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



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

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

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

1. Open the Token Builder window.

See *About Token Builder* on page 50.

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

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

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

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

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

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

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

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

5. Select the text in the **Token** field.
6. Copy the text in the **Token** field by typing **Ctrl + C**.
7. Type **Ctrl + V** to paste the text into another field.



Note

For a list of all explicit entity format tokens, see [Appendix A, Tokens](#), on page 133.

Nesting Explicit Entity Tokens within Other Tokens

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

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

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

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

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

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

```
description of my workflow
```

User Data Format

User data fields use tokens differently, as shown below:

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

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

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

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

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

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

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



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.

Parameter Format

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

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

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

■ ■ Note

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

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

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

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

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

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

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

■ ■ 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 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, the tokens must follow specific formats. These formats differ, depending on the item referenced within the table. [Figure 4-3](#) shows the basic elements of the table. These elements are referenced when discussing the different options for referencing data within the table using tokens.

Figure 4-3. Table component formats

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

The format `[REQD.T.<TABLE_TOKEN>]` represents the table and specific tokens will be represented as `[REQD.T.<TABLE_TOKEN>.<SPECIFIC_TOKENS>]`. The following sections provide examples of the formats used for tokens that reference items related to the table component:

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

In these examples, the following example is used. A table component named Employee with four columns:

- Name of Employee
- Years of Service of the Employee
- Department where the Employee belongs to
- Salary of the Employee.

These columns are defined as follows:

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

To access the table row count from a Request context:

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

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

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

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

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

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

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

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

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

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

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

[TE.VP.DEPARTMENT]

It is possible to 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 table's first column. This is particularly useful when a column is a list of user names, and this list can be used for sending these users notification.

Sub-Entity Format

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

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

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

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

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

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

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

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

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

Environment and Environment Application Tokens

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

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

In addition, a general `ENV` Prefix can be used in the explicit entity format to reference specific environments, as shown in the following example:

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

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

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

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

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

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

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

```
[APP.SERVER_BASE_PATH]
```

In addition, the explicit entity format can be used 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 being migrated on a package line at a given workflow step, and the line uses app code HR. The workflow step has QA as the source environment, and Prod as the destination environment. *Table 4-2* shows other attributes of the environments and applications.

Table 4-2. Sample environment and application attributes

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

Given this setup, *Table 4-3* shows some sample tokens and how each would evaluate.

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 any Mercury IT Governance Center Extensions are installed, there are more environment tokens with the prefix 'AC.' For information about these tokens, see the Mercury IT Governance Center Extensions documentation.

Using Token Builder

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

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

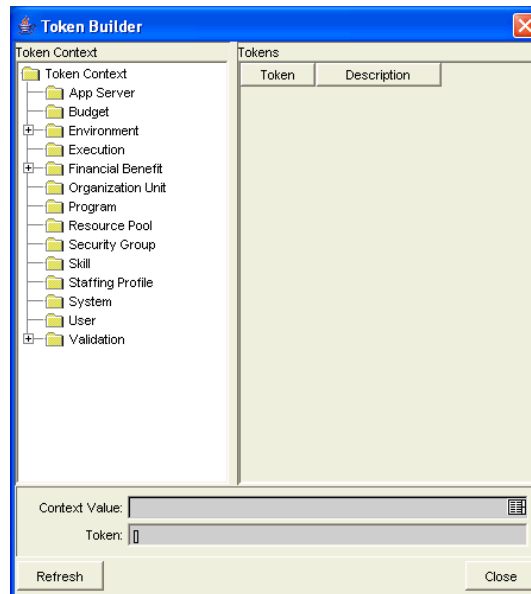
Token Builder generates tokens in the explicit entity format by providing a list of possible entity name values. When such a list is available, the **Context Value** field at the bottom of the Token Builder is enabled. You can either type in the field to reduce the list, or click the auto-complete icon to open the Validate window (see *About Token Builder*). 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 Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
The 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.
The **Commands** tab is displayed.
6. In the **Commands** tab, click **New Cmd**.
The Commands window opens.
7. In the Commands window, click **Tokens**.
The Token Builder window opens.



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

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

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

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

A Validate window opens with a list of users.

10. Scroll down to and select **jsmith**.

11. In the Validate window, 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].

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

13. Select the text in the **Token** field.
14. Copy the text in the **Token** field by typing **Ctrl + C**.
15. Type **Ctrl + V** to paste the text into another field.

A decorative graphic consisting of four colored squares (orange, teal, dark red, and a larger dark red square) and a large white number '5' on a dark red background. The word 'Chapter' is written in dark red above the '5', and 'Using Validations' is written in dark red below the '5'.

Chapter 5

Using Validations

In This Chapter:

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

- *Configuring Directory Chooser Validations*
 - *Configuring File Chooser Validations*
 - *Configuring Date Field Validations*
 - *Configuring 1800 Character Text Areas*
 - *Configuring the Table Validations*
 - *Configuring Table Components*
 - *Configuring Table Rules*
-

About Validations

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

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

Every Mercury IT Governance Center installation includes predefined system validations. As you configure your system, you can use these system validations. If no system validation meets your specific requirements, you can use the Validation Workbench to create your own validation. (For details, see *Configuring Validations* on page 74.)

Validation Component Types

The following table summarizes the available types of field components. You can only use certain component types in a workflow step source validation.

Table 5-1. Component Types (page 1 of 3)


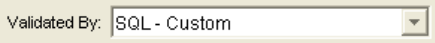
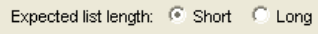
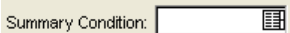
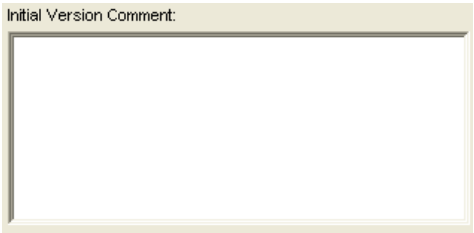
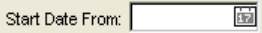
Component Type	Use In Workflow?	Description and Example
Text field	Yes	Text entry fields displayed on a single line. Text fields can be configured to display the data according to a certain format. For example, you can configure a text field to accept and format a hyphenated nine-digit social security number or a ten-digit phone number. 
Drop Down List	Yes	Field that displays a list of values. 
Radio Buttons (Yes/No)	No	Field that accepts yes/no input. 
Auto-Complete List	Yes	Field that lets you open a dialog box that lists choices. 
Text Area	No	Text entry field that can include multiple lines. 
Date Field	No	Control that lets you specify date and time in one long, medium, short, or no format. 

Table 5-1. Component Types (page 2 of 3)








Component Type	Use In Workflow?	Description and Example
Web Address (URL)	No	Text entry field for entering a URL. Clicking U opens a browser window to the specified Web address. 
File Chooser	No	Used only in object types. Requires that two fields be defined with the following tokens: P_FILE_LOCATION and P_SUB_PATH. See Accessing Validations Through Packages and Requests on page 71 for configuration details. 
Directory Chooser	No	Used only in object types. Requires that a parameter field be defined with the token P_FILE_LOCATION. 
Attachment	No	Field used to locate and attach files. 

Table 5-1. Component Types (page 3 of 3)

Component Type	Use In Workflow?	Description and Example
Password Field	No	Field used to capture passwords. 
Table Component	No	Used to enter multiple records into a single component. The table component can be configured to include multiple columns of varied data types. This component supports rules for populating elements within the table and provides functionality for capturing column totals. For details, see Configuring the Table Validations on page 120 . You can only add fields of this component type to request types, request header types and request user data. Hardware Information 2 Entries 
Budget, Staffing Profile, Financial Benefit	No	Field that you can add to the request type to enable access to view, edit or create budgets, staffing profiles, or financial benefits associated with a request, project, or work plan. You can only add fields of this component type to a request type. Budget: Team T Allocations 

Accessing Validations Through Packages and Requests

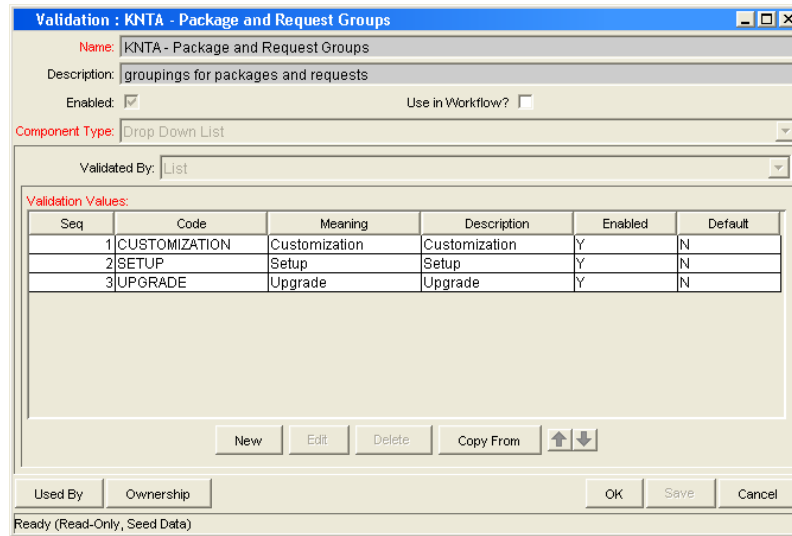
You can access the package and request group validations directly from the Package window. To specify that a package belongs to a new or unique package group that is not named in the auto-complete validation list, you are not required to go through the Validation Workbench.

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

- From the Workbench menu, select **Package > New Package Group**.

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

Figure 5-1. Validation window



Note

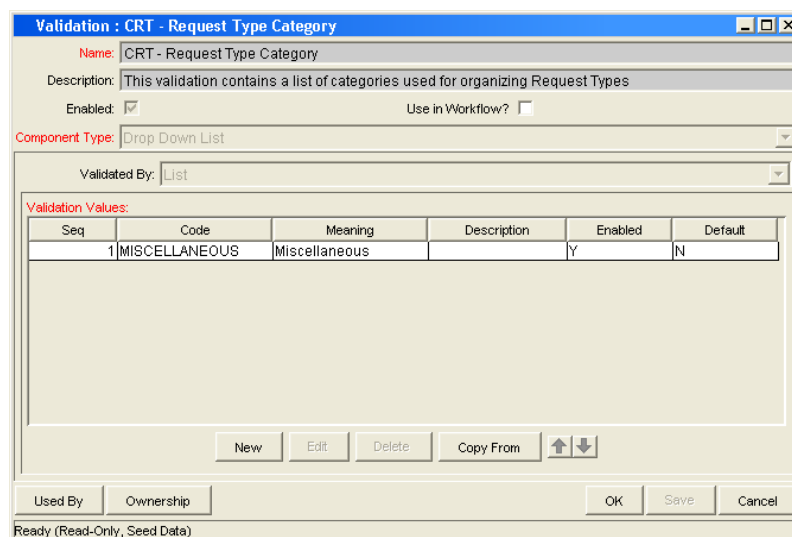
Although all users can view this window, only users with the required security privileges can change the package and request groups validation list.

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

- From the Workbench menu, select **Request Type > Request Type Category Setup**.

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

Figure 5-2. Validation window





Note

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

Validations and Special Characters

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

Viewing System Validations

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

To view the existing validations on your instance:

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

The Reports window appears.

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

A list of the Administrative reports is displayed.

4. From the list of Administrative reports, select **Validations Report**.

The Validations Report window opens.

5. Complete the fields in the Validations Report window.

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

6. Click **Submit** and wait for the report to be 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. See the *Security Model Guide and Reference* document for more information concerning access grants.

You cannot delete a validation if it is:

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



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 any new workflow or entity definitions.

To configure a new validation:

1. Log on to Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
The Workbench opens.
3. From the shortcut bar, select **Configuration > Validations**.
The Validations Workbench opens.
4. Click **New Validation**.
The Validation window opens.

Validation : Untitled7

Name: Engineering Teams

Description:

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

Configuration | Filter Fields | Filter Layout

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

SQL:
 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

Used By: Ownership

OK Save Cancel

Ready

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

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

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

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

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

- a. Click **Ownership**.

The Ownership window opens.

- b. In the Ownership window, select **Only groups listing below that have the Edit Validations Access Grant**.

The **Add** button is enabled.

- c. In the Ownership window, click **Add**.

The Add Security Group window opens.

- d. In the Add Security Group window, add security groups to the validation.

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

8. Save the validation.

- To save changes to the validation without closing the window, click **Save**.
- To save changes and close the window, click **OK**.

Configuring Static List Validations

A static list validation can be a drop-down 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 can create a validation for their engineering teams. They create a validation called Engineering Teams, consisting of the following values: New Product Introduction, Product One, and Product Two.

To create a static list validation:

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

The Validations Workbench opens.

2. Open a validation.

The Validations window opens.

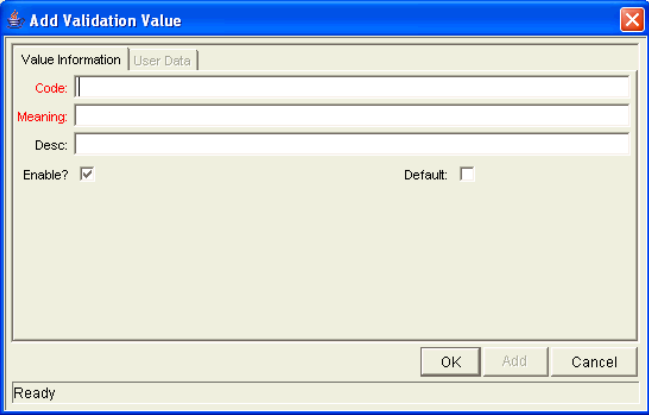
3. In the **Component Type** field, select **Drop Down List** or **Auto Complete List**.

The Validation window dynamically changes fields.

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

5. Click **New** and add a value.

The Add Validation Window opens.



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

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

7. Set the validation value as the default by checking the **Default** field.

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

8. To add the value to the validation:

- To add the value to the validation and close the Add Validation Value window, click **OK**.
- To add the value and keep the Add Validation Value window open, click **Add**.

9. Click **OK** to save the changes to the validation and close the Validation window.

Validation values can be re-ordered using the up and down pointers. The sequence of the validation values determines the order that the values are displayed in the list.



Note

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

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

Configuring Dynamic List Validations

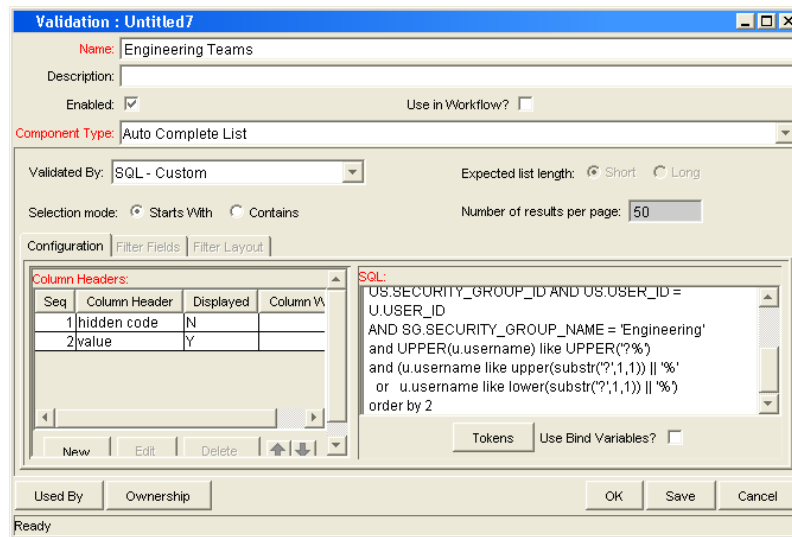
A dynamic list validation can be created using a drop-down or an auto-complete component. You can create validations that provide a dynamic list to the user. This is often a better approach than defining static list validations. Each time a static list validation needs to be updated, a manual update has to occur. Dynamic list validations can often be constructed in such a way as to automatically pick up and display the altered values.

For example, XYZ Corporation needs a field validation that lists all users who are on their Support Team. They could construct a validation that is validated by a list of users, but any time the Support Team changed (members join or leave the department) the list would have to be manually updated. XYZ decides instead to create a dynamic list validation. They create an auto-complete validation that is validated by a SQL statement. The SQL statement returns all users who are a member of the Support Team security group. When the security group membership is altered, the validation is automatically updated with the correct values.

A dynamic list validation can be created using a drop-down or an auto-complete component.

Configuring SQL Validations

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



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

Table 5-2. Column Headers

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

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

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


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



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

SQL Validation Tips

The following guidelines are helpful when writing a SQL statement for a SQL-validated validation:

- The SQL statement must query at least two columns. The first column is a hidden value which is never displayed, and is often 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 in the auto-complete window that appear are constrained by what was first typed in the field. Generally, the constraint is case insensitive. This is accomplished by writing the SQL statement to query only values that match what was typed.

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

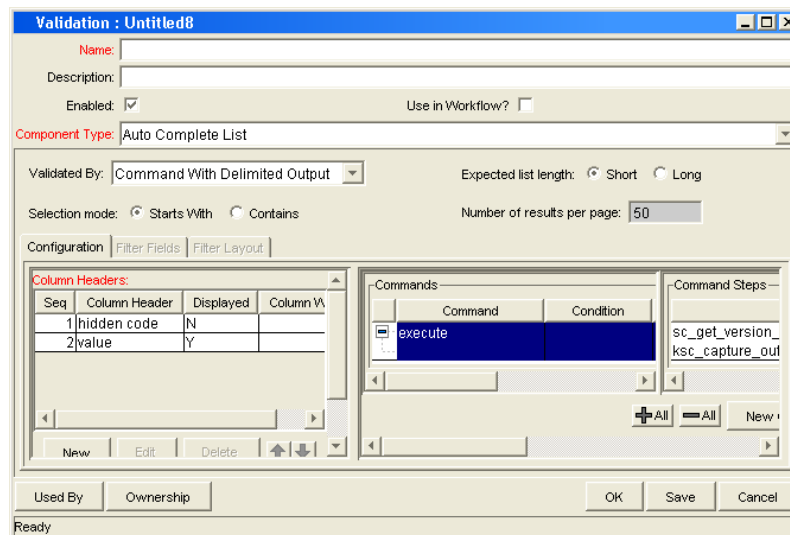
```
where UPPER(<displayed_column>) like UPPER('??')
and (<displayed_column> like upper(substr('?',1,1)) || '%'
or <displayed_column> like lower(substr('?',1,1)) || '%')
```

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

Command Validations

An auto-complete can contain command line executions that return and display a list of values. To define a dynamic list of choices, set an auto-complete to Validated By - Command with Delimited Output or Command with Fixed Width Output. Then enter commands the **Commands** section. See *Configuring the Auto-Complete Values* on page 94 for detailed instructions.

Figure 5-3. Auto-complete using command validation

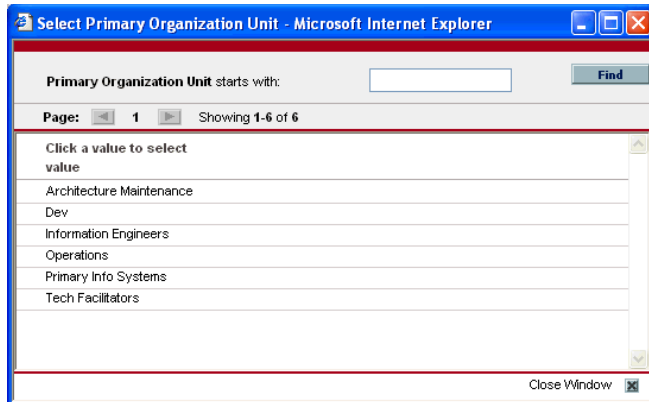


Configuring Short List Auto-Complete Field Validations

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

Auto-complete fields configured to display a short list of entries, displaying all of the values on a single page. *Figure 5-4* shows the Select window for a short list auto-complete field.

Figure 5-4. Short list auto-complete



To configure a short list auto-complete field:

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



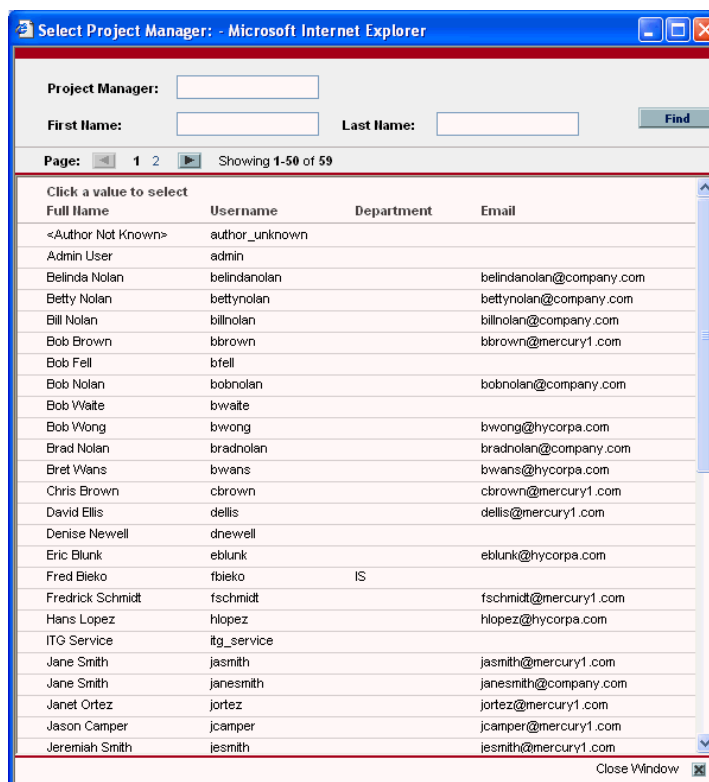
Auto-completes configured as short lists load all values when the window is opened. This can lead to a slower load time and an unfavorable user experience. For fields with many possible values, consider formatting the auto-complete using the long list format.

Configuring Long List Auto-Complete Field Validations

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

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

Figure 5-5. Long list auto-complete



To configure a long list auto-complete field:

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

The Validations Workbench opens.

2. Open a validation.

The Validations window opens.

3. Create a new validation or open an existing validation.

The Validation window opens.

4. In the **Component Type** field, select **Auto Complete List**.

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

6. Click **Save**.



Note

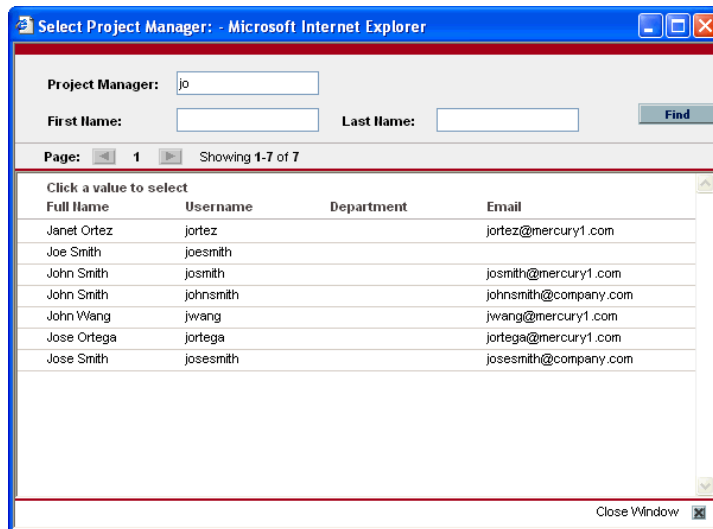
Auto-completes configured as long lists only load a limited set of values when the window is opened. For extremely long lists or lists that are at risk of loading slowly (for example the values are obtained from an alternate database), consider using the long list format.

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

Configuring Automatic Value Matching and Interactive Select Pages

This section provides instructions for configuring auto-complete fields to filter a list of possible values based on a matching character string. It also provides instructions for configuring the automatic value-limiting that occurs on the auto-complete's Select page. *Figure 5-6* shows an auto-complete field that has opened to display matching values.

Figure 5-6. Auto-complete field and matching values on the Select page



An Overview of Matching for “Starts with” or “Contains”

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

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

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

Table 5-3. Automatic character matching field behavior

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

Table 5-4. Automatic character matching Select page behavior

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

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

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

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

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

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

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

This feature is only available for short lists.

3. From the **Selection** option, select **Starts With**.
4. Save the validation.



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

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

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

The Validations Workbench opens.

2. Open the auto-complete Validation.

The Validation window opens.

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

This feature is only available for short lists.

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

5. Save the validation.



Note

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

Configuration Tips

Consider the following tips when configuring the “starts with” versus “contains” functionality for auto-complete fields and the Select page.

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

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

Using “contains” can be useful here. The user knows that he needs to log a bug against one of the IS-supported Financial applications. The user types

“bug” into the auto-complete field and types the **Tab** key. The following items are returned:

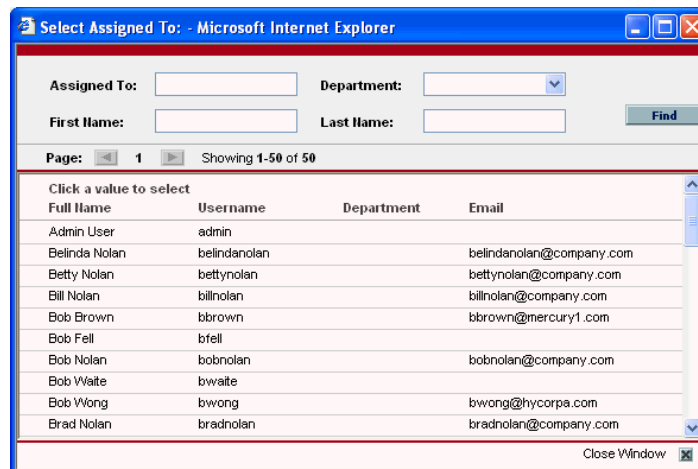
```
Development Bug
IS Bug
```

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

Adding Search Fields to Long List Auto-Complete Validations

Auto-completes with a long list of values can be configured to display additional filter fields in the Select window. These fields can be used to search other properties than the primary values in the list. Users can enter values in the filter fields, and then click **Find** to display only the values that match the search criteria. *Figure 5-7* shows the Select window with additional filter fields.

Figure 5-7. Filter fields in the auto-complete select window



Note

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

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

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

The Validations Workbench opens.

2. Open the validation for the auto-complete.

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

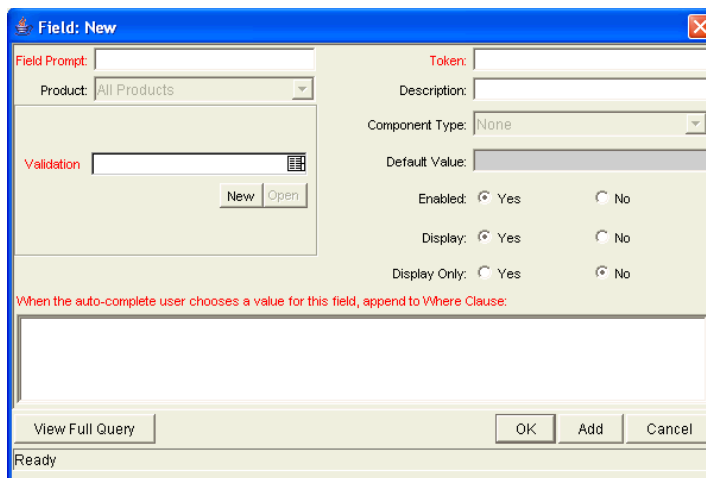
3. In the **Expected list length** field, select **Long**.

Only long formatted auto-completes can include filter fields.

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

5. Click **New**.

The Field: New window opens.



6. Enter the required information.

Table 5-5 defines all of the controls in this window.

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

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

7. Click **OK**.

■ ■ Note

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

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

Configuring the Filter Field Layout

To modify the filter field layout:

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

The Validations Workbench opens.

2. Open the auto-complete validation that includes filter fields on the **Filter Fields** tab.
3. Click the **Filter Layout** tab.

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

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

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

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

■ ■ Note

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

6. To switch the positions of two fields:
 - a. Select the first field, and then 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 an Auto-Complete List of Users (Special Case)

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

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

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

To configure a user auto-complete validation:

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

The Validations Workbench opens.

2. Create a new validation.

The Validation window opens.

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

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

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

See *Configuring the Auto-Complete Values* on page 94 for details.

6. Click **Save**.

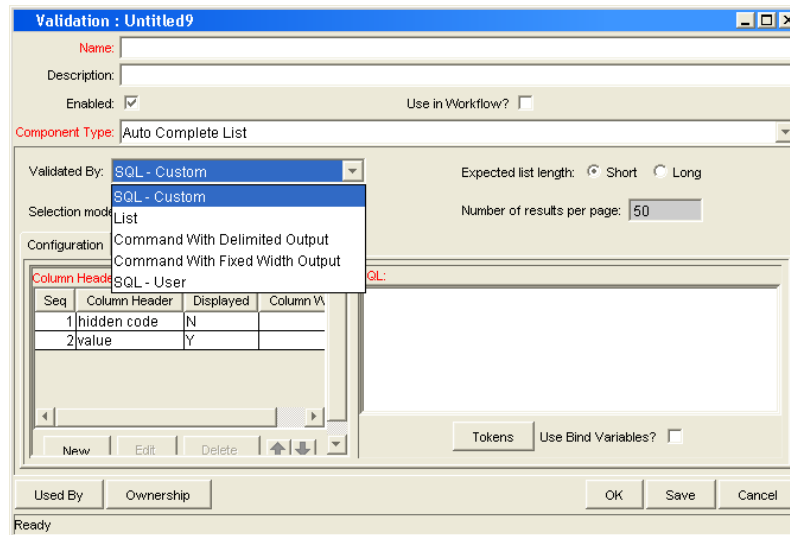
Configuring the Auto-Complete Values

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

- **List**. Used to enter specific values.
- **SQL**. Uses a 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 5-8. Auto-Complete List



For more information on creating auto-completes validated by List or SQL, see [Configuring Static List Validations on page 77](#) and [Configuring Dynamic List Validations on page 79](#).

Configuring Validations by Commands With Delimited Output

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

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

To configure a validation by command with delimited output:

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

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

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

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

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

Figure 5-9. Validation by command with delimited output

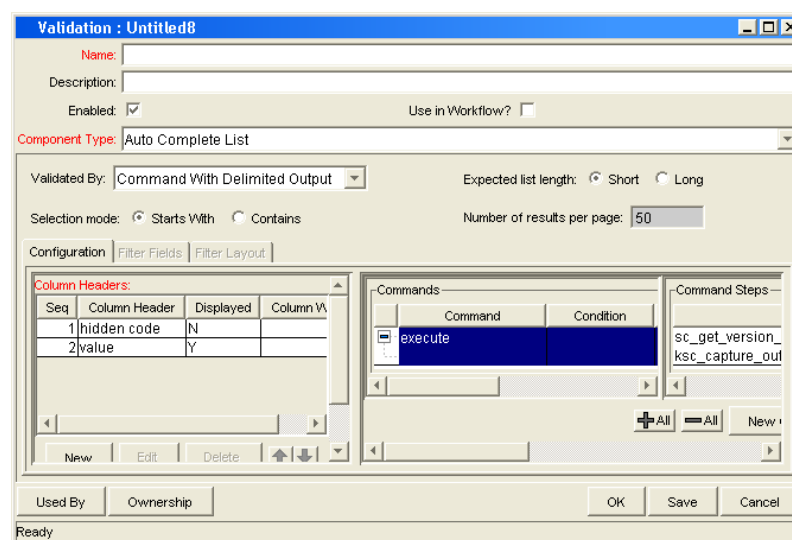


Table 5-6. Validation by command with delimited output

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

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

Table 5-7. Column headers

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

Configuring Validations by Commands with Fixed Width Output

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

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

In the Validation Workbench, under **Validated By**, choose **Command With Fixed Width Output** and enter the width information. Then, under **New Command**, type the command steps. These can include special commands. Include the special command `ksc_capture_output` in you commands. This command captures and parses the delimited command output. If you place the `ksc_capture_output` between `ksc_connect` and `ksc_disconnect`, the command

is run on the remote system. Otherwise, it is run locally on the Mercury IT Governance Server (as `ksc_local_exec` is).

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

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

Figure 5-10. Validation by command with fixed width output

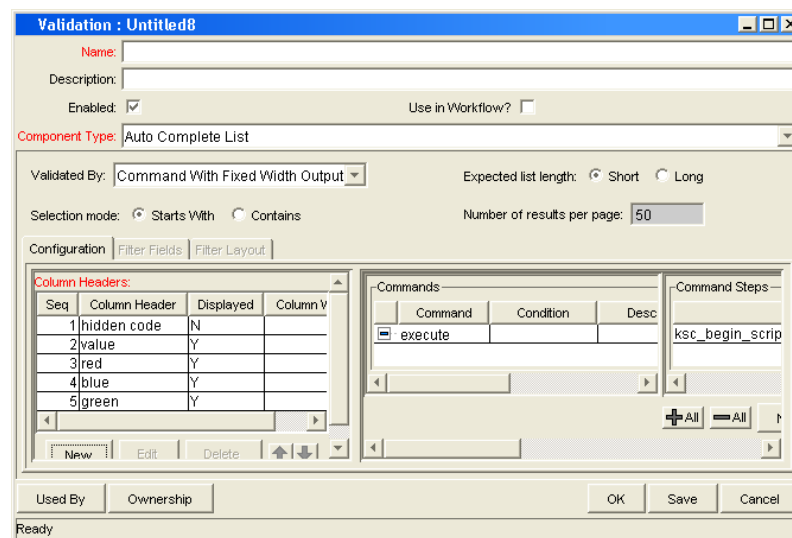


Table 5-8. Validation by command with fixed width output

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

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

Table 5-9. Column headers

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

Configuring User-Defined Multi-Select Auto-Complete Fields

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

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

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

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

- Request header types
- Object types

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

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

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

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

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

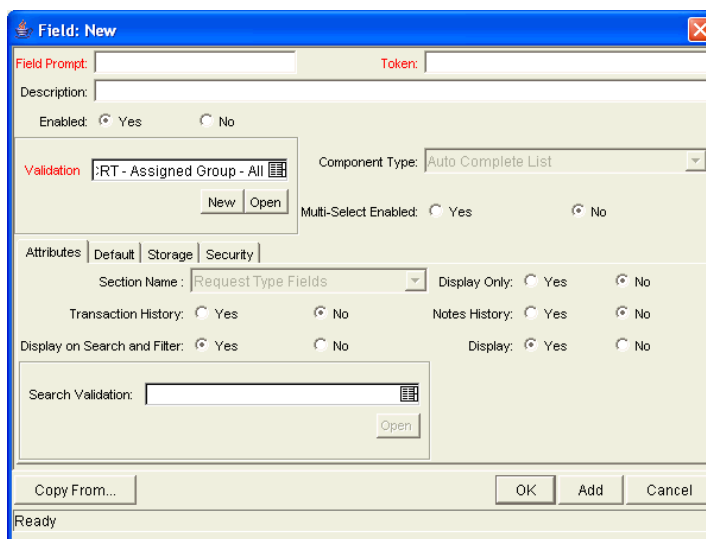
The Request Types Workbench opens.

2. Open a Request Type.

The Request Type window opens.

3. Click **New**.

The Field: New window opens.



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

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

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

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

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

7. Click **OK**.

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

Example of Token Evaluation and Validation by Command with Delimited Output

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

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

1. Generate a validation and set the following parameters as shown here:
 - a. Name: **demo_client_token_parsing**
 - b. Component Type: **Auto Complete List**
 - c. Validated By: **Command With Delimited Output**
 - d. Data Delimiter: | (bar)
 - e. Command
 - Command: **Validate_from_file**
 - Steps

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

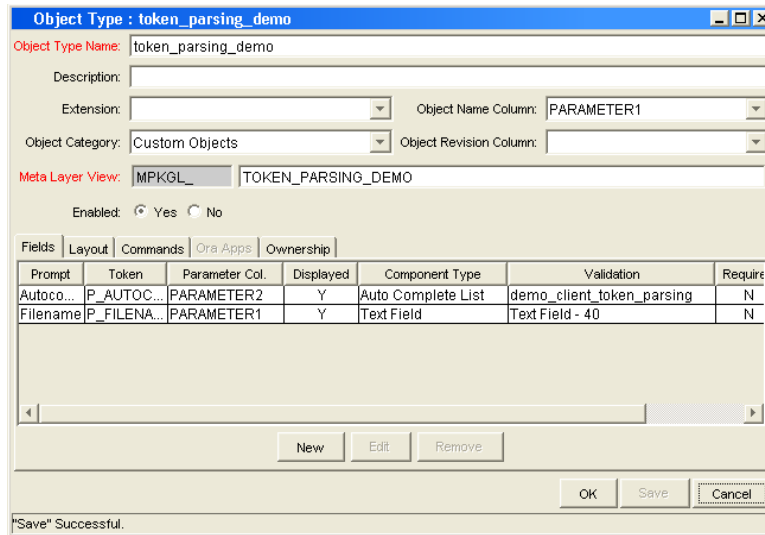
The screenshot shows the 'Validation: demo_client_token_parsing' configuration window. The 'Name' field is set to 'demo_client_token_parsing'. The 'Component Type' is 'Auto Complete List'. The 'Validated By' is 'Command With Delimited Output'. The 'Expected list length' is 'Short'. The 'Selection mode' is 'Starts With'. The 'Number of results per page' is '50'. The 'Data Delimiter' is '|'. The 'Commands' section shows a table with one command: 'Validate_from_file'. The 'Command Steps' section shows the following steps: 'ksc_connect_source_server SC', 'ksc_capture_output cat [P.P_Fil', and 'ksc_exit'. The 'Used By' and 'Ownership' fields are empty. The 'OK', 'Save', and 'Cancel' buttons are visible at the bottom. A status message at the bottom left reads '"Save" Successful.'

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

Command	Command Steps
Validate_from_file	ksc_connect_source_server SC ksc_capture_output cat [P.P_Fil ksc_exit

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

2. Generate an object type named token_parsing_demo.



- a. Generate a new field with the following parameter settings:
 - Name: Filename
 - Token: P_FILENAME
 - Validation: Text Field - 40
- b. Generate a new field with the following parameter settings:
 - Name: AutoComp
 - Token: P_AUTOCOMP
 - Validation: demo_client_token_parsing (the validation defined in [step 2](#))

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

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

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

To test the configuration sample:

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

The Packages Workbench opens.

2. Open a new package.

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

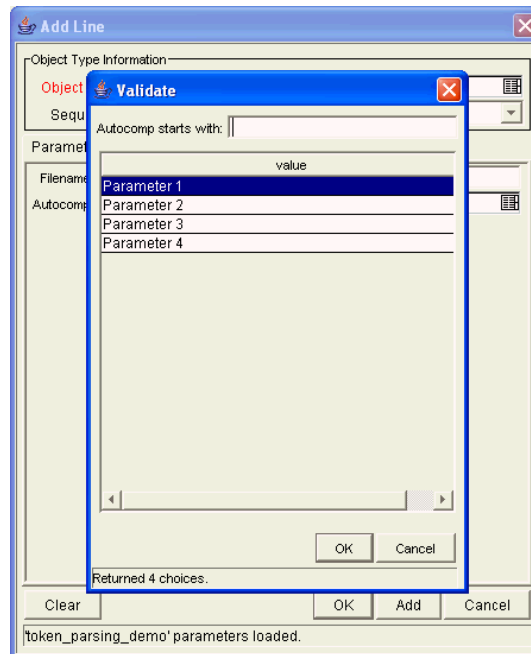
The Add Line window opens.

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

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

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

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



Configuring Text Field Validations

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

To create a text field validation:

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

The Validations Workbench opens.

2. Open a validation.

The Validations window opens.

3. In the **Name** field, type the validation name.
4. In the **Component Type** field, select **Text Field**.
5. In the **Data Mask** field, select the data mask that represents the format you want for the field data.

For more information, see *Text Data Masks for Validations* on page 105.

6. (Optional) Configure the selected data mask.

For information about a data mask, see [Text Data Masks for Validations on page 105](#).

7. To view the results of your data mask settings:
 - a. In the **Sample Input** field, enter a value to preview based on your settings.
 - b. Click **Format**.

The Formatted Output window displays the results.

8. Click **OK**.

Text Data Masks for Validations

Mercury IT Governance Center includes a number of preconfigured data masks that can be used when creating text field validations. Each of these data masks can be configured to meet your specific data requirements. [Table 5-10](#) defines the data masks delivered with Mercury IT Governance Center.

Table 5-10. Data Mask Formats (page 1 of 2)

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

Table 5-10. Data Mask Formats (page 2 of 2)

Data Mask	Description
Currency	<p>Field is used to display currency data and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none"> ■ Range of valid values (maximum and minimum) for the field ■ Whether a zero is displayed if no data is entered ■ Whether a group separator such as a comma is used to display large numbers ■ Negative number display ■ Number of decimal places <p>For more detailed information, see Configuring the Currency Data Mask on page 108.</p>
Percentage	<p>Field is used to display percentages and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none"> ■ Range of valid values (maximum and minimum) for the field ■ Whether a zero is displayed if no data is entered ■ Whether a group separator such as a comma is used to display large numbers ■ Negative number display ■ Number of decimal places <p>For more detailed information, see Configuring the Percentage Data Mask on page 111.</p>
Telephone	<p>Field is used to display telephone numbers and accepts only numeric characters. You can specify the following characteristics for this data mask:</p> <ul style="list-style-type: none"> ■ Format - specify the number of digits included, and the delimiter to be used between groups of numbers. For example, you can specify dashes (-) or periods (.) between numbers (555-555-5555 or 555.555.5555). ■ Maximum and minimum number of digits <p>For more detailed information, see Configuring the Telephone Data Mask on page 112.</p>
Custom	<p>Field allows a range of custom inputs. You can customize the field to accept digits, letters, spaces, and custom delimiters. For more detailed information, see Configuring a Custom Data Mask on page 114.</p>

Configuring the Numeric Data Mask

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

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

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

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

The screenshot shows a dialog box titled "Validation : Untitled5". It has several sections:

- Name:** An empty text field.
- Description:** An empty text field.
- Enabled:** A checked checkbox.
- Use in Workflow?:** An unchecked checkbox.
- Component Type:** A dropdown menu set to "Text Field".
- Data Mask:** A dropdown menu set to "Numeric".
- Maximum Value:** A text field containing "99999".
- Minimum Value:** A text field containing "-9999".
- If Data not Entered, then display a zero:** Two radio buttons, "Yes" is selected.
- Use Group Separator:** Two radio buttons, "Yes" is selected.
- Negative Number looks like:** A dropdown menu set to "-1000".
- Number of Decimal Places:** A text field containing "2".
- Sample Input:** A text field containing "45000.22".
- Formatted Output:** A text field containing "45,000.22".
- Buttons:** "Used By", "Ownership", "OK", "Save", and "Cancel".

Table 5-11. Fields for configuring the numeric data mask for text fields

Field Name	Description
Maximum Value	Largest value allowed for this field. You can specify a positive or negative number.
Minimum Value	Smallest accepted value for the field. You can specify positive or negative number.
If Data not Entered, then display a zero	Determines whether a field with no data displays a zero.
Use Group Separator	Determines if the field uses a group separator (such as a comma) to divide characters within large numbers (for example, whether 1000000 is displayed as 1,000,000). The default character used as the separator depends on the locale setting for the machine, but you can use the Regional Settings window in the Workbench (select Edit > Regional Settings) to change the delimiter.
Negative Number looks like	Used to select one of the following four formats for the display of negative numbers: <ul style="list-style-type: none"> ■ (1000)—parentheses and black text ■ (1000)—parentheses and red text ■ -1000—minus character (-) and black text ■ -1000—minus character (-) and red text
Number of Decimal Places	Determines the maximum number of decimal places used to display values.

Configuring the Currency Data Mask

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

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

Figure 5-12 shows the fields that can be configured for this data mask. Table 5-12 defines these fields.

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

Table 5-12. Fields configuring the currency data mask for text fields
(page 1 of 2)

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

Table 5-12. Fields configuring the currency data mask for text fields
(page 2 of 2)

Field Name	Description
Use Group Separator	Determines if the field should use a group separator (such as a comma) to divide characters within large numbers. For example: 1000000 versus 1,000,000. The character used for the separator defaults based on the machine's local, but can be configured in the Regional Settings window in the Workbench. Select Edit > Regional Settings to access this window.
Negative Number looks like	Determines the text used to display negative numbers. The four possible options are: <ul style="list-style-type: none"> ■ (1000)—parentheses and black text ■ (1000)—parentheses and red text ■ -1000—minus character (-) and black text ■ -1000—minus character (-) and red text
Number of Decimal Places	The maximum number of decimal places displayed.

■ ■ Note

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

```
INSTALLATION_CURRENCY=$;RIGHT
```

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

Configuring the Percentage Data Mask

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

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

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

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

The screenshot shows a dialog box titled "Validation : Untitled5" with the following fields and options:

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

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

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

Configuring the Telephone Data Mask

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

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

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

Figure 5-14. Validation window for the telephone data mask

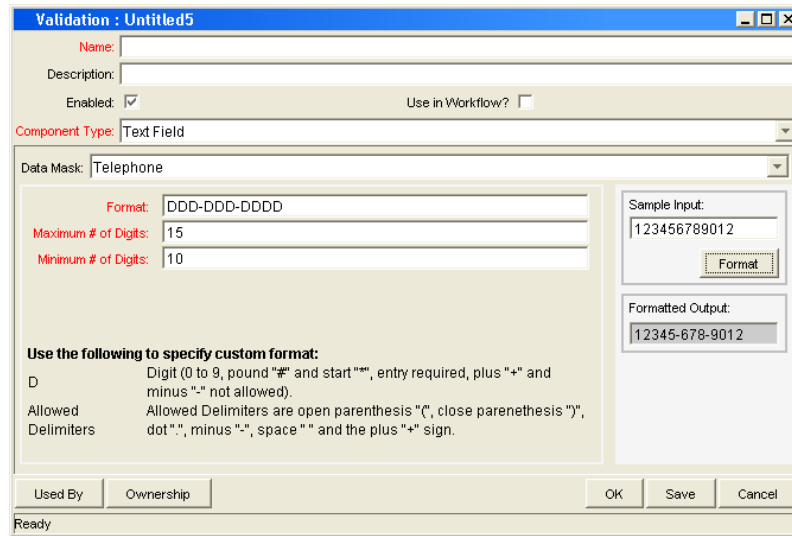


Table 5-14. Fields configuring the telephone data mask for text fields

Field Name	Description
Format	The rule that determines how digits are formatted, including the use of spaces or delimiters. The format definition can include the following delimiters: <ul style="list-style-type: none"> ■ Parentheses () ■ Period (.) ■ Dash (-) ■ Space ■ Plus character (+) For telephone format examples, see Table 5-15 on page 113 .
Maximum # of Digits	The maximum number of digits that the field accepts.
Minimum # of Digits	The minimum number of digits that the field accepts.

Table 5-15. Sample telephone data mask formats

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

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

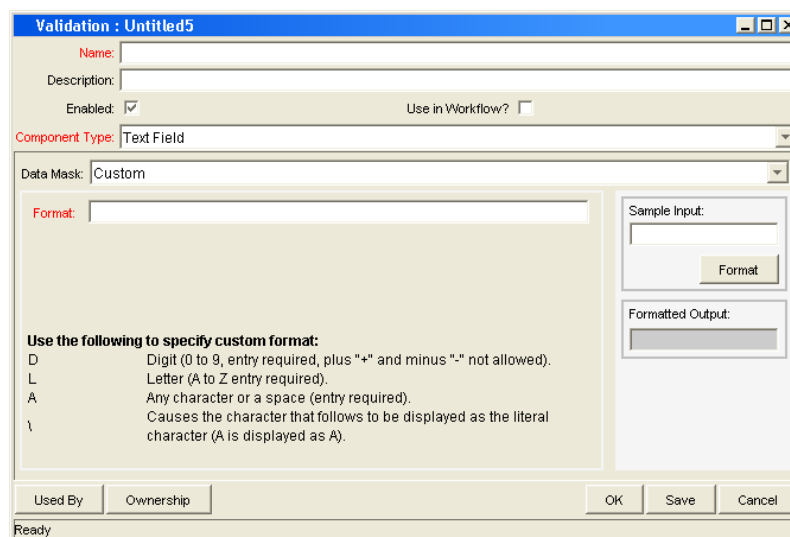
```
Format: DDD-DDD-DDDD
Min: 10
Max: 15
Input: 1234567890
Output: 123-456-7890
Input 2: 12345678901
Output 2: 1234-567-8901
```

Configuring a Custom Data Mask

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

Figure 5-15 shows the fields that you can configure for this data mask.

Figure 5-15. Validation window for the custom data mask



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

- Use D to specify that the user must enter a numeric value between 0 and 9.
- Use L to specify that the user must enter an alphabetic character between A and Z.

- Use A to specify that the user must type a character or space.
- Use a \ (backslash) to specify that the next character is to be displayed as the literal character. For example: “\A” is displayed as “A”.

Table 5-16 illustrates two examples of custom formatting.

Table 5-16. Sample custom data mask formats

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

Configuring Directory Chooser Validations

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

When implementing the Directory Chooser, note the following:

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

Configuring File Chooser Validations

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

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

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

Figure 5-16. Validation window for static environment override in file chooser.

Table 5-17. File chooser field

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

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

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

Table 5-18. Static environment override

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

Token-based Environment Override provides the ability to select a token that is to resolve to the overriding environment. The fields for **Token-based Environment Override** are shown in *Figure 5-17* and defined in *Table 5-19*.

Figure 5-17. Validation window for token-based environment override in file chooser.

The screenshot shows a validation window titled "Validation : Untitled5". It contains the following elements:

- Name:** A text input field.
- Description:** A text input field.
- Enabled:** A checked checkbox.
- Use in Workflow?:** An unchecked checkbox.
- Component Type:** A dropdown menu set to "File Chooser".
- Base File Name Only:** An unchecked checkbox.
- Environment Override Behavior:** A dropdown menu set to "Token-based Environment Override".
- Environment Token:** A text input field.
- Overriding Server Basepath:** A text input field.
- Overriding Client Basepath:** A text input field.
- Buttons:** "Used By", "Ownership", "OK", "Save", and "Cancel".
- Status:** "Ready" at the bottom left.

Table 5-19. Token-based environment override

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

Configuring Date Field Validations

Date fields can accept a variety of formats. The current date field validations are separated into two categories: all systems and systems using only the English language. These formats are defined in Table 3-14.

Table 5-20. Date field

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

Configuring 1800 Character Text Areas

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

To create a validation with a character length of 1800:

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

The Validations Workbench opens.

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

You can use the new Text Area validation (with a length of 1800 characters) when defining a custom field in the product.



Note

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

Configuring the Table Validations

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

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

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

Figure 5-18. Hardware information window

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

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

Configuring Table Components

To create a table component field:

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

The Validations Workbench opens.

2. Click **New Validation**.

The Validation window opens.

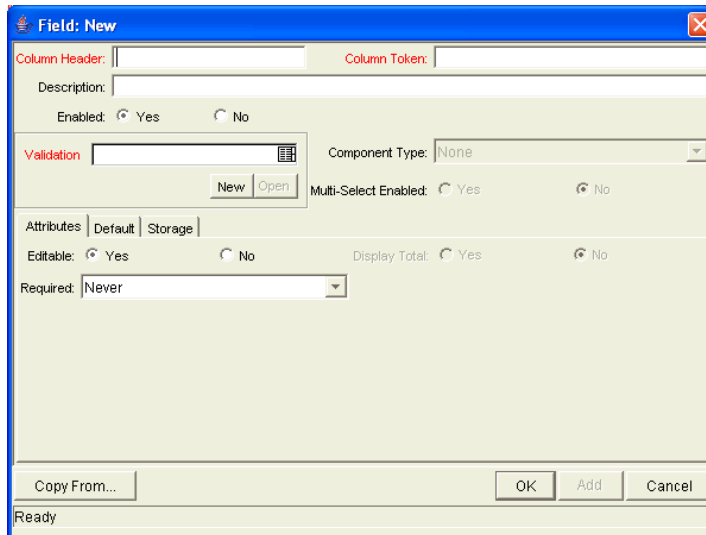
3. Select **Table Component** from the Component Type drop-down list.

4. Enter a validation name and description.
5. Enter any user instructions to display on the top of the table entry page.
6. Create the table columns, as follows:
 - a. Click **New** in the **Table Columns** tab.
The Field window opens.
 - b. Define the type of information to store that column.



Note

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



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

Validation : Hardware Table

Name: Hardware Table

Description:

Enabled: Use in Workflow?

Component Type: Table Component

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

Meta Layer View: MREQ_ HARDWARE_TABLE

Column Seq.	Column Header	Column Token	Parameter Col.	Enabled	Component Type	Validation
1	Products	PRODUCTS	PARAMETER1	Y	Auto Complete List	Hardware Products
2	Quantity	QUANTITY	PARAMETER2	Y	Text Field	Text Field - 20
3	Price	PRICE	PARAMETER3	Y	Text Field	Text Field - 20
4	Total	TOTAL	PARAMETER4	Y	Text Field	Text Field - 20

Used By: Ownership

OK Save Cancel

Ready

7. Configure the form layout, as follows:

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

Validation : Hardware Table

Name: Hardware Table

Description:

Enabled: Use in Workflow?

Component Type: Table Component

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

Meta Layer View: MREQ_ HARDWARE_TABLE

Table Columns Form Layout Rules

- Products
- Quantity
- Price

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

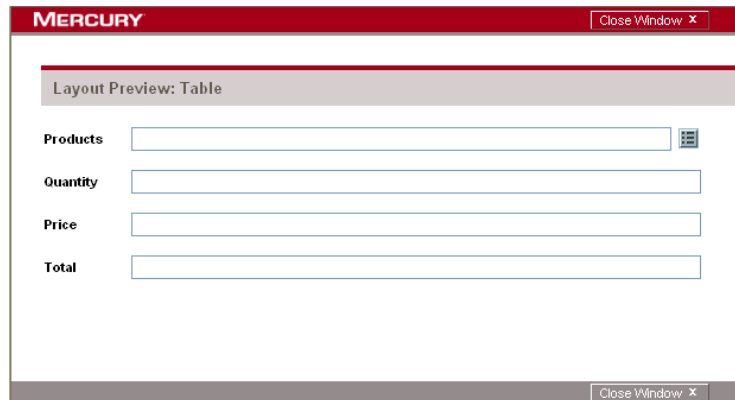
Preview

Used By: Ownership

OK Save Cancel

Ready

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



Note

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

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



Note

For detailed instructions on how to create a rule, see [Configuring Table Rules on page 125](#).

9. Click **OK**.

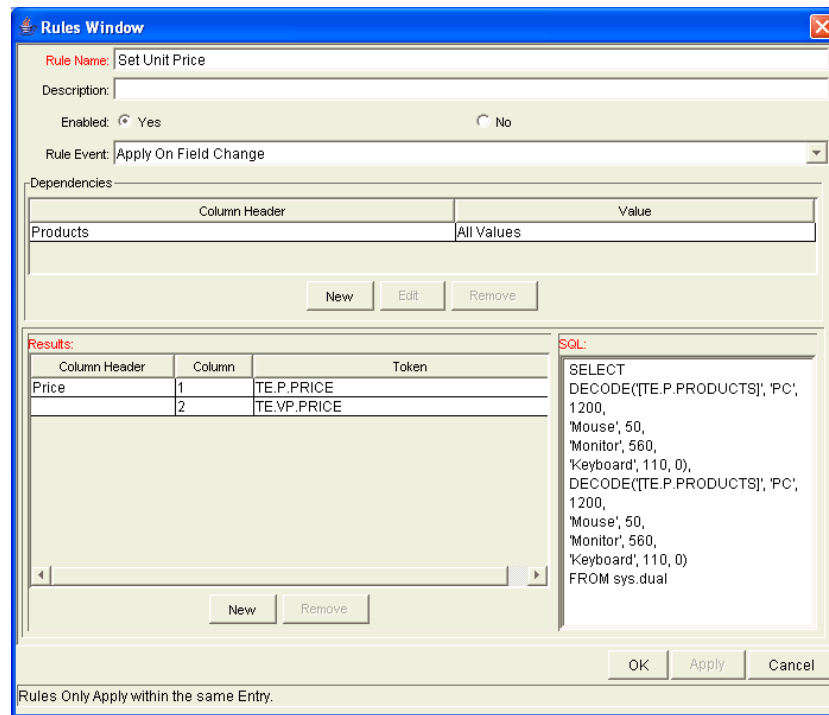
The new **Table Component** field can be included on a request type, request header type or request user data field.

Configuring Table Rules

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

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

Figure 5-19. Rules window accessed from the Rules tab



Example of Using a Table Component on an Order Form

The following example illustrates the table component rules functionality.

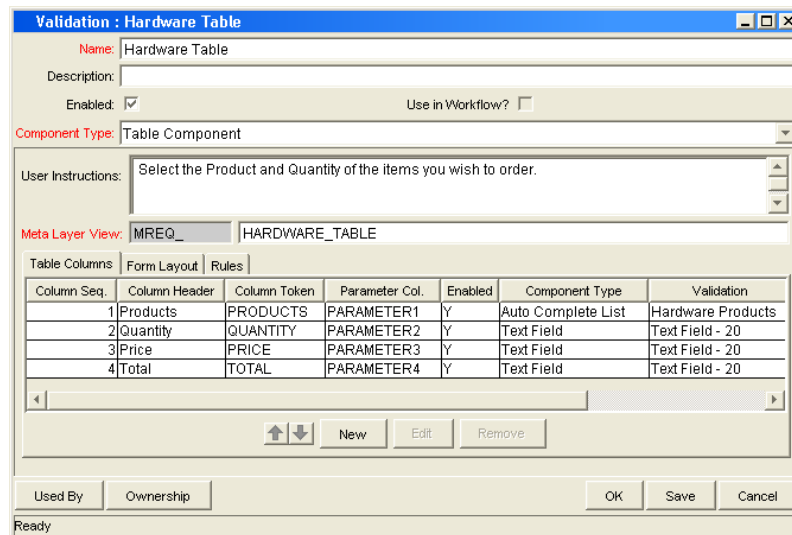
XYZ Corporation uses a request for creating and tracking employee computer hardware equipment orders. XYZ has included a table component field on their request type for gathering the order information. When the employee selects a Product, the Unit Price is automatically updated. Then, when they update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

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

Table 5-21. Example, table component validation settings

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

Figure 5-20. Validations window



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

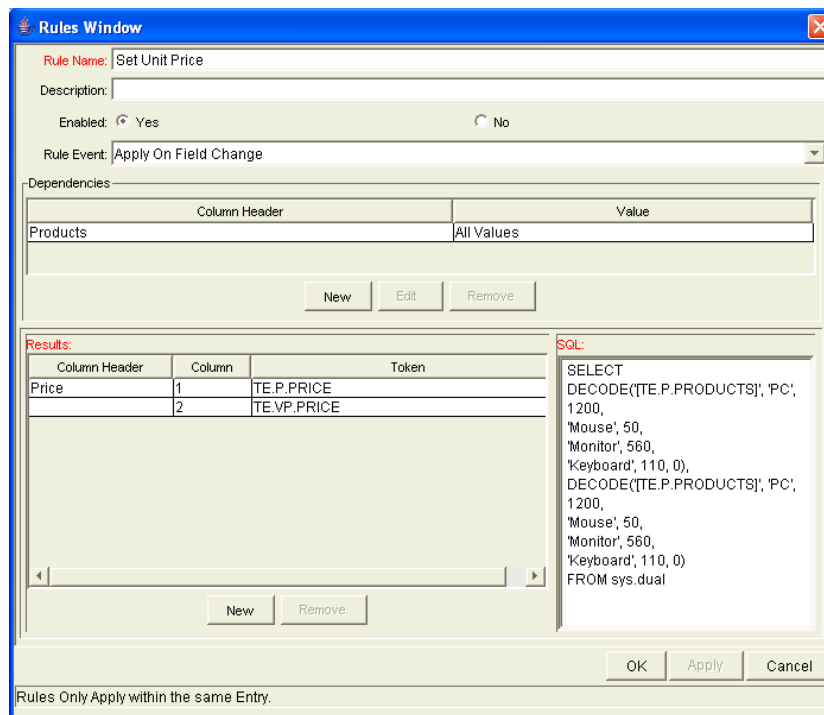
Example of Setting Unit Prices

XYZ Corporation uses the rule described in *Table 5-22* and show in *Table* to set the default unit price based on the product selected.

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

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

Figure 5-21. Rules window



Example of Calculating Totals

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

Table 5-23. Example - Calculate Total rule settings

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

Using Table Components

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

Figure 5-22. Hardware information window

Using Tokens in Table Components

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

Calculating Column Totals

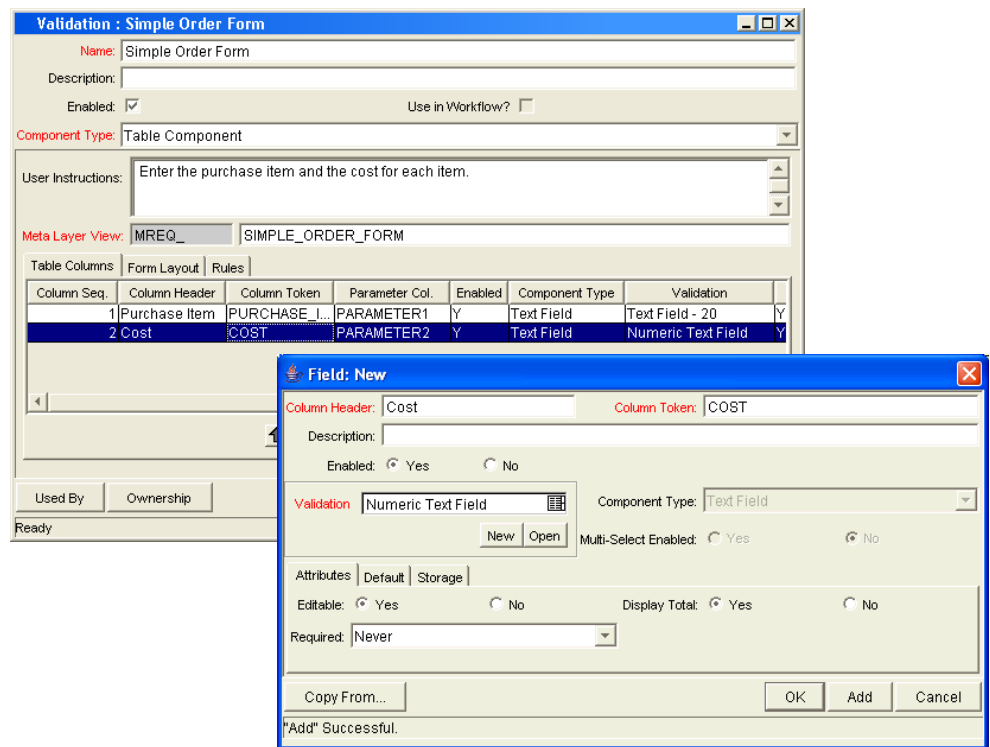
You can configure columns that are validated by a number to calculate the total for that column. This is configured in the validation's Field window. The following example illustrates how to configure a column to calculate and display the column total.

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

XYZ creates a validation with the following settings:

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

Figure 5-23. Sample validation for a Simple Order table component.



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

Figure 5-24. Sample table component displaying a column total.

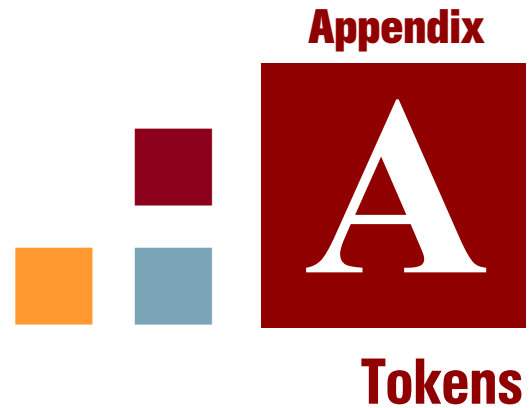
MERCURY Close Window X

Simple Order Form

Enter the purchase item and the cost for each item.

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

Close Window X



In This Appendix:

- *Overview of Tokens*
- *Application Server Tokens*
- *Budget Tokens*
- *Contact Tokens*
- *Distribution Tokens*
- *Document Management Tokens*
- *Environment Tokens*
 - *Environment > Dest Env Tokens*
 - *Environment > Dest Env > App Tokens*
 - *Environment > Dest Env > Env Tokens*
 - *Environment > Env Tokens*
 - *Environment > Env > App Tokens*
 - *Environment > Env > Env Tokens*
 - *Environment > Source Env Tokens*
 - *Environment > Source Env > App Tokens*
 - *Environment > Source Env > Env Tokens*
- *Command Tokens*
- *Financial Benefit Tokens*
- *Notification Tokens*
- *Organization Unit Tokens*
- *Package Tokens*
 - *Package > Package Line Tokens*
 - *Package > Pending Reference Tokens*
- *Package Line Tokens*
- *Program Tokens*

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

Overview of Tokens

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

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

Application Server Tokens

Table A-1. Application Server tokens

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

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

Budget Tokens

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

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

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

Prefix	Tokens	Description
BGT	BUDGET_IS_FOR_NAME	The name of the work plan/program/org unit to which the budget is linked.
BGT	BUDGET_NAME	The name of the budget.
BGT	BUDGET_ROLLS_UP_TO_ID	The ID of the budget into which this budget rolls up.
BGT	BUDGET_ROLLS_UP_TO_NAME	The name of the budget into which this budget rolls up.
BGT	BUDGET_URL	The URL used to view this budget.
BGT	CREATED_BY	The username of the user who created the budget.
BGT	CREATION_DATE	The date the budget was created.
BGT	DESCRIPTION	The budget description.
BGT	END_PERIOD	The budget end period.
BGT	INITIATION_REQ	The budget initiation request ID.
BGT	PERIOD_SIZE	The budget period size.
BGT	REGION	Region associated with the budget.
BGT	START_PERIOD	The budget start period.
BGT	STATUS_CODE	The budget status code.
BGT	STATUS_NAME	The budget status name.

Contact Tokens

Table A-3. Contact tokens

Prefix	Tokens	Description
CON	COMPANY	The company ID for which the contact works.
CON	COMPANY_NAME	The name of the company for which the contact works.
CON	CONTACT_ID	The contact ID (defined in the table KCRT_CONTACTS).
CON	CREATED_BY	The ID of the user who created the contact.
CON	CREATION_DATE	The date the contact was created.
CON	EMAIL_ADDRESS	The email address of the contact.
CON	FIRST_NAME	The first name of the contact.
CON	FULL_NAME	The full name of the contact.
CON	LAST_NAME	The last name of the contact.
CON	LAST_UPDATED_BY	The ID of the user who last updated the contact.
CON	LAST_UPDATE_DATE	The date the contact was last updated.
CON	PHONE_NUMBER	The phone number of the contact.
CON	USERNAME	The contact username (if applicable). This can be the username for an external system, and not Mercury IT Governance Center.
CON	USER_ID	The userID of the contact, if the contact is a Mercury IT Governance Center user.

Distribution Tokens

Table A-4. Distribution tokens

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

Document Management Tokens

Table A-5. Document Management tokens

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

Environment Tokens

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

Environment > Dest Env Tokens

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

Prefix	Tokens	Description
DEST_ENV	CLIENT_BASE_PATH	The base (root) path of the client.
DEST_ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
DEST_ENV	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connect protocol.
DEST_ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
DEST_ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
DEST_ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
DEST_ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
DEST_ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
DEST_ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
DEST_ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.
DEST_ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
DEST_ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
DEST_ENV	CREATED_BY	The ID of the user who created the environment.
DEST_ENV	CREATION_DATE	The date the environment was created.

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

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

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

Prefix	Tokens	Description
DEST_ENV	LAST_UPDATED_BY	The ID of the user who last updated the environment.
DEST_ENV	LAST_UPDATE_DATE	The date the environment was last updated.
DEST_ENV	LOCATION	The environment location.
DEST_ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
DEST_ENV	SERVER_BASE_PATH	The base (root) path of the server.
DEST_ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
DEST_ENV	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
DEST_ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
DEST_ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
DEST_ENV	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
DEST_ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
DEST_ENV	SERVER_NAME	The DNS name or IP address of the server computer.
DEST_ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
DEST_ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
DEST_ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
DEST_ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
DEST_ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Environment > Dest Env > App Tokens

Table A-7. Environment > Dest Env > App tokens (page 1 of 2)

Prefix	Tokens	Description
DEST_ENV.APP	APP_CODE	The short name (code) for the application.
DEST_ENV.APP	APP_NAME	The descriptive name for the application.
DEST_ENV.APP	CLIENT_BASE_PATH	The application-specific base (root) path of the client.
DEST_ENV.APP	CLIENT_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
DEST_ENV.APP	CLIENT_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the client.
DEST_ENV.APP	CLIENT_CON_PROTOCOL	The application-specific protocol used to connect to this client.
DEST_ENV.APP	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connection protocol.
DEST_ENV.APP	CLIENT_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this client.
DEST_ENV.APP	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
DEST_ENV.APP	CREATED_BY	The ID of the user who created the application.
DEST_ENV.APP	CREATION_DATE	The date the application was created.
DEST_ENV.APP	DB_LINK	For Oracle database type, the application-specific database link from the Mercury IT Governance Center schema to the database schema for the environment.
DEST_ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.
DEST_ENV.APP	DB_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the database. This value is encrypted.

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

Prefix	Tokens	Description
DEST_ENV.APP	DB_USERNAME	The application-specific username or schema name that Mercury IT Governance Center uses to log on to or access the database.
DEST_ENV.APP	DESCRIPTION	The application description.
DEST_ENV.APP	ENABLED_FLAG	The flag that indicates whether the application is enabled and available for selection in package lines.
DEST_ENV.APP	ENVIRONMENT_APP_ID	The ID of the application in the table KENV_ENVIRONMENT_APPS.
DEST_ENV.APP	ENVIRONMENT_ID	The ID of the environment with which the application is associated.
DEST_ENV.APP	ENVIRONMENT_NAME	The name of the environment with which the application is associated.
DEST_ENV.APP	LAST_UPDATED_BY	The ID of the user who last updated the application.
DEST_ENV.APP	LAST_UPDATE_DATE	The date the application was last updated.
DEST_ENV.APP	SERVER_CON_PROTOCOL	The application-specific protocol used to connect to this server.
DEST_ENV.APP	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
DEST_ENV.APP	SERVER_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this server.
DEST_ENV.APP	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
DEST_ENV.APP	SERVER_BASE_PATH	The application-specific base (root) path of the server.
DEST_ENV.APP	SERVER_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
DEST_ENV.APP	SERVER_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the server.
DEST_ENV.APP	WORKBENCH_ENVIRONMENT_URL	The URL of the environment window in the Workbench.

Environment > Dest Env > Env Tokens

Table A-8. Environment > Dest Env > Env tokens (page 1 of 3)

Prefix	Tokens	Description
DEST_ENV.ENV	CLIENT_BASE_PATH	The base (root) path of the client.
DEST_ENV.ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
DEST_ENV.ENV	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connect protocol.
DEST_ENV.ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
DEST_ENV.ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
DEST_ENV.ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
DEST_ENV.ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
DEST_ENV.ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
DEST_ENV.ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
DEST_ENV.ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.
DEST_ENV.ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
DEST_ENV.ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
DEST_ENV.ENV	CREATED_BY	The ID of the user who created the environment.
DEST_ENV.ENV	CREATION_DATE	The date the environment was created.
DEST_ENV.ENV	DATABASE_ENABLED_FLAG	The flag that indicates whether the database portion of the environment is enabled.
DEST_ENV.ENV	DATABASE_TYPE	The validation value code of the database type.

Table A-8. Environment > Dest Env > Env tokens (page 2 of 3)

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

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

Prefix	Tokens	Description
DEST_ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
DEST_ENV.ENV	SERVER_BASE_PATH	The base (root) path of the server.
DEST_ENV.ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
DEST_ENV.ENV	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
DEST_ENV.ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
DEST_ENV.ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
DEST_ENV.ENV	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
DEST_ENV.ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
DEST_ENV.ENV	SERVER_NAME	The DNS name or IP address of the server computer.
DEST_ENV.ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
DEST_ENV.ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
DEST_ENV.ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
DEST_ENV.ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
DEST_ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Environment > Env Tokens

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

Prefix	Tokens	Description
ENV	CLIENT_BASE_PATH	The base (root) path of the client.
ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
ENV	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connect protocol.
ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.
ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
ENV	CREATED_BY	The ID of the user who created the environment.
ENV	CREATION_DATE	The date the environment was created.
ENV	DATABASE_ENABLED_FLAG	The flag that indicates whether the database portion of the environment is enabled.
ENV	DATABASE_TYPE	The validation value code of the database type.

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

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

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

Prefix	Tokens	Description
ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
ENV	SERVER_BASE_PATH	The base (root) path of the server.
ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
ENV	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
ENV	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
ENV	SERVER_NAME	The DNS name or IP address of the server computer.
ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Environment > Env > App Tokens

Table A-10. Environment > Env > App tokens (page 1 of 2)

Prefix	Tokens	Description
ENV.APP	APP_CODE	The short name (code) for the application.
ENV.APP	APP_NAME	The descriptive name for the application.
ENV.APP	CLIENT_BASE_PATH	The application-specific base (root) path of the client.
ENV.APP	CLIENT_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
ENV.APP	CLIENT_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the client.
ENV.APP	CLIENT_CON_PROTOCOL	The application-specific protocol used to connect to this client.
ENV.APP	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connection protocol.
ENV.APP	CLIENT_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this client.
ENV.APP	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
ENV.APP	CREATED_BY	The ID of the user who created the application.
ENV.APP	CREATION_DATE	The date the application was created.
ENV.APP	DB_LINK	For Oracle database type, the application-specific database link from the Mercury IT Governance Center schema to the database schema for the environment.
ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.
ENV.APP	DB_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the database. This value is encrypted.

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

Prefix	Tokens	Description
ENV.APP	DB_USERNAME	The application-specific username or schema name that Mercury IT Governance Center uses to log on to or access the database.
ENV.APP	DESCRIPTION	The application description.
ENV.APP	ENABLED_FLAG	The flag that indicates whether the application is enabled and available for selection in package lines.
ENV.APP	ENVIRONMENT_APP_ID	The ID of the application in the table KENV_ENVIRONMENT_APPS.
ENV.APP	ENVIRONMENT_ID	The ID of the environment with which the application is associated.
ENV.APP	ENVIRONMENT_NAME	The name of the environment with which the application is associated.
ENV.APP	LAST_UPDATED_BY	The ID of the user who last updated the application.
ENV.APP	LAST_UPDATE_DATE	The date the application was last updated.
ENV.APP	SERVER_CON_PROTOCOL	The application-specific protocol used to connect to this server.
ENV.APP	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
ENV.APP	SERVER_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this server.
ENV.APP	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
ENV.APP	SERVER_BASE_PATH	The application-specific base (root) path of the server.
ENV.APP	SERVER_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
ENV.APP	SERVER_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the server.
ENV.APP	WORKBENCH_ENVIRONMENT_URL	The URL of the environment window in the Workbench.

Environment > Env > Env Tokens

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

Prefix	Tokens	Description
ENV.ENV	CLIENT_BASE_PATH	The base (root) path of the client.
ENV.ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
ENV.ENV	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connect protocol.
ENV.ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
ENV.ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
ENV.ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
ENV.ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
ENV.ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
ENV.ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
ENV.ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.
ENV.ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
ENV.ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
ENV.ENV	CREATED_BY	The ID of the user who created the environment.
ENV.ENV	CREATION_DATE	The date the environment was created.
ENV.ENV	DATABASE_ENABLED_FLAG	The flag that indicates whether the database portion of the environment is enabled.
ENV.ENV	DATABASE_TYPE	The validation value code of the database type.

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

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

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

Prefix	Tokens	Description
ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
ENV.ENV	SERVER_BASE_PATH	The base (root) path of the server.
ENV.ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
ENV.ENV	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
ENV.ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
ENV.ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
ENV.ENV	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
ENV.ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
ENV.ENV	SERVER_NAME	The DNS name or IP address of the server computer.
ENV.ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
ENV.ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
ENV.ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
ENV.ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Environment > Source Env Tokens

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

Prefix	Tokens	Description
SOURCE_ENV	CLIENT_BASE_PATH	The base (root) path of the client.
SOURCE_ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
SOURCE_ENV	CLIENT_CON_PROTOCOL_MEANING	The visible value of the client connect protocol.
SOURCE_ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
SOURCE_ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
SOURCE_ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
SOURCE_ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
SOURCE_ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
SOURCE_ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
SOURCE_ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.
SOURCE_ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
SOURCE_ENV	CLIENT_TRANSFER_PROTOCOL_MEANING	The visible value of the client transfer protocol.
SOURCE_ENV	CREATED_BY	The ID of the user who created the environment.
SOURCE_ENV	CREATION_DATE	The date the environment was created.
SOURCE_ENV	DATABASE_ENABLED_FLAG	The flag that indicates whether the database portion of the environment is enabled.
SOURCE_ENV	DATABASE_TYPE	The validation value code of the database type.

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

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

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

Prefix	Tokens	Description
SOURCE_ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
SOURCE_ENV	SERVER_BASE_PATH	The base (root) path of the server.
SOURCE_ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
SOURCE_ENV	SERVER_CON_PROTOCOL_MEANING	The visible value of the server connection protocol.
SOURCE_ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
SOURCE_ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
SOURCE_ENV	SERVER_TRANSFER_PROTOCOL_MEANING	The visible value of the server transfer protocol.
SOURCE_ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
SOURCE_ENV	SERVER_NAME	The DNS name or IP address of the server computer.
SOURCE_ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
SOURCE_ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
SOURCE_ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
SOURCE_ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
SOURCE_ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Environment > Source Env > App Tokens

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

Prefix	Token	Description
SOURCE_ ENV.APP	APP_CODE	The short name (code) for the application.
SOURCE_ ENV.APP	APP_NAME	The descriptive name for the application.
SOURCE_ ENV.APP	CLIENT_BASE_PATH	The application-specific base (root) path of the client.
SOURCE_ ENV.APP	CLIENT_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
SOURCE_ ENV.APP	CLIENT_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the client.
SOURCE_ ENV.APP	CLIENT_CON_PROTOCOL	The application-specific protocol used to connect to this client.
SOURCE_ ENV.APP	CLIENT_CON_PROTOCOL_ MEANING	The visible value of the client connection protocol.
SOURCE_ ENV.APP	CLIENT_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this client.
SOURCE_ ENV.APP	CLIENT_TRANSFER_PROTOCOL_ MEANING	The visible value of the client transfer protocol.
SOURCE_ ENV.APP	CREATED_BY	The ID of the user who created the application.
SOURCE_ ENV.APP	CREATION_DATE	The date the application was created.
SOURCE_ ENV.APP	DB_LINK	For Oracle database type, the application-specific database link from the Mercury IT Governance Center schema to the database schema for the environment.
SOURCE_ ENV.APP	DB_NAME	For a Microsoft SQL Server database, the application-specific database name used to access the database from the command line.

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

Prefix	Token	Description
SOURCE_ ENV.APP	DB_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the database. This value is encrypted.
SOURCE_ ENV.APP	DB_USERNAME	The application-specific username or schema name that Mercury IT Governance Center uses to log on to or access the database.
SOURCE_ ENV.APP	DESCRIPTION	The application description.
SOURCE_ ENV.APP	ENABLED_FLAG	The flag that indicates whether the application is enabled and available for selection in package lines.
SOURCE_ ENV.APP	ENVIRONMENT_APP_ID	The ID of the application in the table KENV_ENVIRONMENT_APPS.
SOURCE_ ENV.APP	ENVIRONMENT_ID	The ID of the environment with which the application is associated.
SOURCE_ ENV.APP	ENVIRONMENT_NAME	The name of the environment with which the application is associated.
SOURCE_ ENV.APP	LAST_UPDATED_BY	The ID of the user who last updated the application.
SOURCE_ ENV.APP	LAST_UPDATE_DATE	The date the application was last updated.
SOURCE_ ENV.APP	SERVER_CON_PROTOCOL	The application-specific protocol used to connect to this server.
SOURCE_ ENV.APP	SERVER_CON_PROTOCOL_ MEANING	The visible value of the server connection protocol.
SOURCE_ ENV.APP	SERVER_TRANSFER_PROTOCOL	The application-specific protocol used to transfer files to and from this server.
SOURCE_ ENV.APP	SERVER_TRANSFER_PROTOCOL_ MEANING	The visible value of the server transfer protocol.
SOURCE_ ENV.APP	SERVER_BASE_PATH	The application-specific base (root) path of the server.

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

Prefix	Token	Description
SOURCE_ ENV.APP	SERVER_PASSWORD	The application-specific password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
SOURCE_ ENV.APP	SERVER_USERNAME	The application-specific username Mercury IT Governance Center uses to log on to or access the server.
SOURCE_ ENV.APP	WORKBENCH_ENVIRONMENT_ URL	The URL of the environment window in the Workbench.

Environment > Source Env > Env Tokens

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

Prefix	Tokens	Description
SOURCE_ ENV.ENV	CLIENT_BASE_PATH	The base (root) path of the client.
SOURCE_ ENV.ENV	CLIENT_CON_PROTOCOL	The protocol used to connect to this client.
SOURCE_ ENV.ENV	CLIENT_CON_PROTOCOL_ MEANING	The visible value of the client connect protocol.
SOURCE_ ENV.ENV	CLIENT_NAME	The DNS name or IP address of the client computer.
SOURCE_ ENV.ENV	CLIENT_NT_DOMAIN	The domain name for the client, if the client machine is running Windows.
SOURCE_ ENV.ENV	CLIENT_ENABLED_FLAG	The flag that indicates whether the client portion of the environment is enabled.
SOURCE_ ENV.ENV	CLIENT_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the client. This value is encrypted.
SOURCE_ ENV.ENV	CLIENT_SQL_COMMAND	The default command line SQL*Plus command name.
SOURCE_ ENV.ENV	CLIENT_TYPE_CODE	The validation value code of the client machine type.
SOURCE_ ENV.ENV	CLIENT_USERNAME	The username Mercury IT Governance Center uses to log on to or access the client.

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

Prefix	Tokens	Description
SOURCE_ ENV.ENV	CLIENT_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this client.
SOURCE_ ENV.ENV	CLIENT_TRANSFER_PROTOCOL_ MEANING	The visible value of the client transfer protocol.
SOURCE_ ENV.ENV	CREATED_BY	The ID of the user who created the environment.
SOURCE_ ENV.ENV	CREATION_DATE	The date the environment was created.
SOURCE_ ENV.ENV	DATABASE_ENABLED_FLAG	The flag that indicates whether the database portion of the environment is enabled.
SOURCE_ ENV.ENV	DATABASE_TYPE	The validation value code of the database type.
SOURCE_ ENV.ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.
SOURCE_ ENV.ENV	DB_JDBC_URL	The JDBC URL used in Oracle 9i RAC configuration.
SOURCE_ ENV.ENV	DB_LINK	For Oracle database type, the database link from the Mercury IT Governance Center schema to the environment's database schema.
SOURCE_ ENV.ENV	DB_NAME	The DNS name or IP address of the database server.
SOURCE_ ENV.ENV	DB_ORACLE_SID	For Oracle database type, the SID of the database (often the same as the DB_CONNECT_STRING).
SOURCE_ ENV.ENV	DB_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the database. This value is encrypted.
SOURCE_ ENV.ENV	DB_PORT_NUMBER	For Oracle database type, the port number on which SQL*Net is listening for remote SQL connections on the database server.
SOURCE_ ENV.ENV	DB_USERNAME	The username or schema name Mercury IT Governance Center uses to log on to or access the database.

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

Prefix	Tokens	Description
SOURCE_ ENV.ENV	DB_VERSION	The database version (such as 8.1.7).
SOURCE_ ENV.ENV	DESCRIPTION	The environment description.
SOURCE_ ENV.ENV	ENABLED_FLAG	The flag that Indicates whether the environment is enabled and available for use in workflows.
SOURCE_ ENV.ENV	ENVIRONMENT_ID	The ID of the environment in the table KENV_ENVIRONMENTS.
SOURCE_ ENV.ENV	ENVIRONMENT_NAME	The environment name.
SOURCE_ ENV.ENV	LAST_UPDATED_BY	The ID of the user who last updated the environment.
SOURCE_ ENV.ENV	LAST_UPDATE_DATE	The date the environment was last updated.
SOURCE_ ENV.ENV	LOCATION	The environment location.
SOURCE_ ENV.ENV	MSSQL_DB_NAME	For a Microsoft SQL Server database type, the database name used to access the database from the command line.
SOURCE_ ENV.ENV	SERVER_BASE_PATH	The base (root) path of the server.
SOURCE_ ENV.ENV	SERVER_CON_PROTOCOL	The protocol used to connect to this server.
SOURCE_ ENV.ENV	SERVER_CON_PROTOCOL_ MEANING	The visible value of the server connection protocol.
SOURCE_ ENV.ENV	SERVER_SQL_COMMAND	The default command line SQL*Plus command name.
SOURCE_ ENV.ENV	SERVER_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.
SOURCE_ ENV.ENV	SERVER_TRANSFER_PROTOCOL_ MEANING	The visible value of the server transfer protocol.
SOURCE_ ENV.ENV	SERVER_ENABLED_FLAG	The flag that indicates whether the server portion of the environment is enabled.
SOURCE_ ENV.ENV	SERVER_NAME	The DNS name or IP address of the server computer.

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

Prefix	Tokens	Description
SOURCE_ENV.ENV	SERVER_NT_DOMAIN	The domain name for the server, if the server machine type is Windows.
SOURCE_ENV.ENV	SERVER_PASSWORD	The password Mercury IT Governance Center uses to log on to or access the server. This value is encrypted.
SOURCE_ENV.ENV	SERVER_TYPE_CODE	The validation value code of the server machine type.
SOURCE_ENV.ENV	SERVER_USERNAME	The username Mercury IT Governance Center uses to log on to or access the server.
SOURCE_ENV.ENV	WORKBENCH_ENVIRONMENT_URL	URL to access the Environment window for this environment in the Workbench.

Command Tokens

Table A-15. Command tokens

Prefix	Tokens	Description
EXEC	EXIT_CODE	The exit code of a command execution.
EXEC	OUTPUT	The last line of output from a command execution.

The command execution tokens, `[EXEC.OUTPUT]` and `[EXEC.EXIT_CODE]`, can be used in the following contexts:

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

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

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

Financial Benefit Tokens

Table A-16. Financial Benefit tokens (page 1 of 2)

Prefix	Tokens	Description
FBEN	ACTIVE_FLAG	The active flag of the financial benefit.
FBEN	BENEFIT_ID	The ID of the financial benefit.
FBEN	BENEFIT_IS_FOR_ENTITY_NAME	The entity name to which the financial benefit is linked.
FBEN	BENEFIT_IS_FOR_ID	The ID of the asset, project, or proposal to which the financial benefit is linked.
FBEN	BENEFIT_IS_FOR_NAME	The name of the asset, project, or proposal to which the financial benefit is linked.
FBEN	BENEFIT_NAME	The name of the financial benefit.
FBEN	BENEFIT_URL	The URL to view the financial benefit.
FBEN	CREATED_BY	The username of the user who created the financial benefit.
FBEN	CREATION_DATE	The date when the financial benefit was created.
FBEN	DESCRIPTION	The description of the financial benefit.
FBEN	END_PERIOD	The end period of the financial benefit.
FBEN	INITIATION_REQ	The initiation request ID of the financial benefit.
FBEN	PERIOD_SIZE	The period size of the financial benefit.

Table A-16. Financial Benefit tokens (page 2 of 2)

Prefix	Tokens	Description
FBEN	REGION	The region associated with the financial benefit.
FBEN	START_PERIOD	The start period of the financial benefit.
FBEN	STATUS_CODE	The status code of the financial benefit.
FBEN	STATUS_NAME	The status name of the financial benefit.

Notification Tokens

Table A-17. Notification tokens

Prefix	Tokens	Description
NOTIF	CC_USERS	The list of users on the Cc: header of the notification.
NOTIF	CHANGED_FIELD	The field that changed to trigger a notification.
NOTIF	EXCEPTION_RULE	The exception rule that was met by the task exception that caused the notification to be sent.
NOTIF	EXCEPTION_RULE_NAME	The name of the task exception that caused the notification to be sent.
NOTIF	EXCEPTION_VIOLATION	The specific violation of the exception that caused the notification to be sent.
NOTIF	NEW_VALUE	The new value of the changed field.
NOTIF	NOTIFICATION_DETAILS	Notification details for linked tokens.
NOTIF	OLD_VALUE	The previous value of the changed field.
NOTIF	TO_USERS	The list of users on the To: header of the notification.

Organization Unit Tokens

Table A-18. Organization Unit tokens

Prefix	Tokens	Description
ORG	BUDGET_ID	The ID of the budget linked to this org unit.
ORG	BUDGET_NAME	The name of the budget linked to this org unit.
ORG	CATEGORY_CODE	The lookup code of the org unit category (lookup type = RSC - org unit Category)
ORG	CATEGORY_NAME	The category name of the org unit.
ORG	CREATED_BY	The ID of the user that created the org unit.
ORG	CREATED_BY_USERNAME	The name of the user that created the org unit.
ORG	CREATION_DATE	The date that the org unit was created.
ORG	DEPARTMENT_CODE	The lookup code of the org unit department (lookup type = DEPT)
ORG	DEPARTMENT_NAME	The department name of the org unit.
ORG	LOCATION_CODE	The lookup code of the org unit location (lookup type = RSC - Location)
ORG	LOCATION_NAME	The location name of the org unit.
ORG	MANAGER_ID	The ID of the org unit manager.
ORG	MANAGER_USERNAME	The name of the org unit manager.
ORG	ORG_UNIT_ID	The org unit ID (defined in table KRSC_ORG_UNITS).
ORG	ORG_UNIT_NAME	The org unit name.
ORG	PARENT_ORG_UNIT_ID	The parent org unit ID.
ORG	PARENT_ORG_UNIT_NAME	The parent org unit name.
ORG	REGIONAL_CALENDAR	The name of the regional calendar for the org unit.
ORG	REGION	The region associated with the Org Unit.
ORG	TYPE_CODE	The lookup code of the org unit category (lookup type = RSC - org unit Category)
ORG	TYPE_NAME	The type name of the org unit.

Package Tokens

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

Prefix	Tokens	Description
PKG	ASSIGNED_TO_EMAIL	The email address of the user to whom the package is assigned.
PKG	ASSIGNED_TO_GROUP_ID	The ID of the security group to which the package is assigned.
PKG	ASSIGNED_TO_GROUP_NAME	The security group to which the package is assigned.
PKG	ASSIGNED_TO_USERNAME	The name of the user to whom the package is assigned.
PKG	ASSIGNED_TO_USER_ID	The ID of the user to whom the package is assigned.
PKG	CREATED_BY	The ID of the user who created the package.
PKG	CREATED_BY_EMAIL	The email address of the user who created the package.
PKG	CREATED_BY_USERNAME	The Mercury IT Governance Center username of the user who created the package.
PKG	CREATION_DATE	The date the package was created.
PKG	DESCRIPTION	The package description.
PKG	ID	The package ID in the table KDLV_PACKAGES.
PKG	LAST_UPDATED_BY	The ID of the user who last updated the package.
PKG	LAST_UPDATED_BY_EMAIL	The email address of the user who last updated the package.
PKG	LAST_UPDATED_BY_USERNAME	The Mercury IT Governance Center username of the user who last updated the package.
PKG	LAST_UPDATE_DATE	The date the package was last updated.
PKG	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last names of the author of the most recent note.
PKG	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.

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

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

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

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

Package > Package Line Tokens

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

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

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

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

Package > Pending Reference Tokens

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

Prefix	Tokens	Description
PKG.PEND	ID	The ID of the entity that is being blocked by the package.
PKG.PEND	NAME	The name of the entity that is being blocked by the package.
PKG.PEND	DETAIL	Detail information for the entity that is being blocked by the package.
PKG.PEND	DESCRIPTION	The description of the entity that is being blocked by the package.
PKG.PEND	STATUS_ID	The ID of the state or code of the status of the entity that is being blocked by the package.
PKG.PEND	STATUS_NAME	The name of the status (or state) of the entity that is being blocked by the package.
PKG.PEND	STATE	The name of the state of the entity of the request that is being blocked by the package.

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

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

Package Line Tokens

Table A-22. Package Line tokens

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

Program Tokens

Table A-23. Program tokens

Prefix	Tokens	Description
PRG	CREATED_BY	The ID of the user that created the program.
PRG	CREATED_BY_USERNAME	The name of the user that created the program.
PRG	LAST_UPDATED_BY	The ID of the user that last updated the program.
PRG	LAST_UPDATED_BY_USERNAME	The name of the user that last updated the program.
PRG	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent (chronological) note.
PRG	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
PRG	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent (chronological) note.
PRG	MOST_RECENT_NOTE_TEXT	Text of the most recent (chronological) note.
PRG	PROGRAM_MANAGER	The ID(s) of the user(s) assigned to manage this program.

Project Tokens

Table A-24. Project tokens (page 1 of 4)

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

Table A-24. Project tokens (page 2 of 4)

Prefix	Tokens	Description
PRJ	BUDGET_NAME	The name of the budget linked to the work plan.
PRJ	CONFIDENCE_CODE	The code of the confidence value entered by the user.
PRJ	CONFIDENCE_NAME	The name of the confidence value entered by the user.
PRJ	CREATED_BY	The user who created the work plan.
PRJ	CREATED_BY_EMAIL	The email address of the user who created the work plan.
PRJ	CREATED_BY_USERNAME	The username of the person who created the work plan.
PRJ	CREATION_DATE	The creation date of the work plan.
PRJ	DEPARTMENT_CODE	The code of the department value entered by the user.
PRJ	DEPARTMENT_NAME	The name of the department value entered by the user.
PRJ	DESCRIPTION	The description of the work plan.
PRJ	ESTIMATED_REMAINING_DURATION	The estimated remaining duration of the work plan.
PRJ	ESTIMATED_REMAINING_EFFORT	The estimated remaining effort involved in the work plan.
PRJ	ESTIMATED_FINISH_DATE	The estimated finish date of the work plan.
PRJ	LAST_UPDATE_DATE	The date on which the work plan was last updated.
PRJ	LAST_UPDATED_BY	The last person to update the work plan.
PRJ	LAST_UPDATED_BY_EMAIL	The email address of the last person to update the project plan.
PRJ	LAST_UPDATED_BY_USERNAME	The username of the last person to update the work plan.
PRJ	MASTER_PROJECT_ID	The ID of the master project.
PRJ	MASTER_PROJECT_NAME	The name of the master project.
PRJ	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.

Table A-24. Project tokens (page 3 of 4)

Prefix	Tokens	Description
PRJ	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
PRJ	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
PRJ	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
PRJ	MOST_RECENT_NOTE_TYPE	Type of the most recent note (USER or FIELD CHANGE).
PRJ	MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_AUTHOR_USERNAME	Username of the author of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_AUTHORED_DATE	Date of the most recent user note.
PRJ	MOST_RECENT_USER_NOTE_TEXT	Text of the most recent user note.
PRJ	PARENT_PROJECT_ID	The ID of the parent work plan.
PRJ	PARENT_PROJECT_NAME	The name of the parent work plan.
PRJ	PERCENT_COMPLETE	The percent of the work plan completed.
PRJ	PRIORITY	The priority of the work plan.
PRJ	PROGRAM_ID	The delimited list of program ids of all programs associated with the project.
PRJ	PROGRAM_ID_MANAGER	The delimited list of manager ids of all programs associated with the project.
PRJ	PROGRAM_ID_MANAGER_USERNAME	The delimited list of manager usernames of all programs associated with the project.
PRJ	PROGRAM_NAME	The delimited list of program names of all programs associated with the project.
PRJ	PROJECT_ID	The number that uniquely identifies the work plan (same as PROJECT_NUMBER) in the table KDRV_PROJECTS.
PRJ	PROJECT_MANAGER	The manager of the work plan.
PRJ	PROJECT_MANAGER_EMAIL	The email address of the project manager.
PRJ	PROJECT_MANAGER_USERNAME	The username of the project manager.

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

Prefix	Tokens	Description
PRJ	PROJECT_NAME	The work plan name.
PRJ	PROJECT_NAME_LINK	A standard hyperlink to the work plan in HTML-formatted notifications.
PRJ	PROJECT_NUMBER	The number that uniquely identifies the work plan (same as PROJECT_ID).
PRJ	PROJECT_PATH	The work plan path. This is a hierarchy of parent work plans that contain this work plan.
PRJ	PROJECT_REQUEST_ID	The request ID for this project.
PRJ	PROJECT_STAKEHOLDER	The delimited list of user ids of the project stakeholders.
PRJ	PROJECT_STAKEHOLDER_EMAIL	The delimited list of emails of the project stakeholders.
PRJ	PROJECT_STAKEHOLDER_USERNAME	The delimited list of usernames of the project stakeholders.
PRJ	PROJECT_STATE	The work plan state.
PRJ	PROJECT_TEMPLATE	The name of the project template used to create the project plan.
PRJ	PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
PRJ	PROJECT_URL	The URL for the Project Overview page of the work plan.
PRJ	REGIONAL_CALENDAR	The name of the regional calendar for the work plan
PRJ	SCHEDULED_EFFORT	The scheduled effort defined in the work plan.
PRJ	SCHEDULED_DURATION	The scheduled duration for the work plan.
PRJ	SCHEDULED_FINISH_DATE	The finish date scheduled for the work plan.
PRJ	SCHEDULED_START_DATE	The start date scheduled for the work plan.
PRJ	SUMMARY_CONDITION	The summary condition of the work plan.
PRJ	WORKBENCH_PROJECT_URL	The URL used to access this work plan in Workbench.

Project Detail Tokens

Table A-25. Project Detail tokens

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

Parameters are accessible with this prefix (similar to request detail):
 [PRJD.P.CUSOM_TOKEN].

Release Tokens

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

Prefix	Tokens	Description
REL	RELEASE_ID	The ID of the release in the KREL_RELEASES table.
REL	RELEASE_NAME	The release name.
REL	RELEASE_STATUS	The release status.
REL	CREATED_BY	The ID of the user who created the release.
REL	CREATED_BY_USERNAME	The Mercury IT Governance Center username of the user who created the release.
REL	LAST_UPDATED_BY	The ID of the user who last updated the release.
REL	LAST_UPDATED_BY_USERNAME	The Mercury IT Governance Center username of the user who last updated the release.
REL	LAST_UPDATE_DATE	The date on which the release was last updated.
REL	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.

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

Prefix	Tokens	Description
REL	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
REL	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
REL	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
REL	RELEASE_MANAGER	The Mercury IT Governance Center user who is designated the release manager.
REL	RELEASE_TEAM	The group of Mercury IT Governance Center users associated with the release.
REL	RELEASE_GROUP	The high-level categorization of the release.
REL	DESCRIPTION	The release description.
REL	NOTES	The notes contained within the release.

Release > Distribution Tokens

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

Prefix	Tokens	Description
REL.DIST	CREATED_BY	The user ID of the user that created the distribution.
REL.DIST	CREATE_BE_USERNAME	The username of the user that created the distribution.
REL.DIST	DESCRIPTION	The description of the release.
REL.DIST	DISTRIBUTION_ID	The internal identifier of the distribution for the release.
REL.DIST	DISTRIBUTION_NAME	The name of the distribution for the release.
REL.DIST	DISTRIBUTION_STATUS	The status of the distribution for the release.
REL.DIST	FEEDBACK_FLAG	The feedback flag for the release.
REL.DIST	FEEDBACK_VALUE	The feedback value for the release.
REL.DIST	LAST_UPDATED_BY	The user ID of the user that last updated the distribution.

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

Prefix	Tokens	Description
REL.DIST	LAST_UPDATED_BY_USERNAME	The username of the user that last updated the distribution.
REL.DIST	LAST_UPDATE_DATE	The last update date of the distribution.
REL.DIST	RELEASE_ID	The internal identifier of the release.
REL.DIST	RELEASE_NAME	The name of the release.
REL.DIST	WORKFLOW	The workflow assigned to the release.

Request Tokens

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

Prefix	Tokens	Description
REQ	APPLICATION_CODE	The validation value code for the application to which the request is assigned.
REQ	APPLICATION_NAME	The validation value meaning of the application to which the request is assigned.
REQ	ASSIGNED_TO_EMAIL	The email address of the user to whom the request is assigned.
REQ	ASSIGNED_TO_GROUP_ID	The ID of the security group to which the request is assigned.
REQ	ASSIGNED_TO_GROUP_NAME	The name of the security group to which the request is assigned.
REQ	ASSIGNED_TO_USERNAME	The Mercury IT Governance Center username of the user to whom the request is assigned.
REQ	ASSIGNED_TO_NAME	The full name of the assigned user.
REQ	ASSIGNED_TO_USER_ID	The ID of the user to whom the request is assigned.
REQ	COMPANY	The company employing the user who created the request.

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

Prefix	Tokens	Description
REQ	COMPANY_NAME	The name of the company employing the user that created the request.
REQ	CONTACT_EMAIL	The email address of the contact for the request.
REQ	CONTACT_NAME	The full name of the contact for the request.
REQ	CONTACT_PHONE_NUMBER	The phone number of the contact for the request.
REQ	CREATED_BY	The ID of the user who created the request.
REQ	CREATED_BY_EMAIL	The email address of the user who created the request.
REQ	CREATED_BY_NAME	The full name of the created by user.
REQ	CREATED_BY_USERNAME	The Mercury IT Governance Center username of the user who last updated the request.
REQ	CREATION_DATE	The date on which the request was created.
REQ	DEPARTMENT_CODE	The validation value code of the department for the request.
REQ	DEPARTMENT_NAME	The validation value meaning of the department for the request.
REQ	DESCRIPTION	The request description.
REQ	LAST_UPDATED_BY	The ID of the user who last updated the request.
REQ	LAST_UPDATED_BY_EMAIL	The email address of the user who last updated the request.
REQ	LAST_UPDATED_BY_USERNAME	The Mercury IT Governance Center username of the user who last updated the request.
REQ	LAST_UPDATE_DATE	The date on which the request was last updated.
REQ	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.

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

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

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

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

Request > Pending Reference Tokens

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

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

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

Prefix	Tokens	Description
REQ.PEND	RESOURCE_GROUP_ID	The ID of the assigned group (or resource group) associated with the entity that is being blocked by the request.
REQ.PEND	PERCENT_COMPLETE	The current percent complete value associated with the entity that the request is blocking.
REQ.PEND	ENTITY_TYPE_ID	The ID of the type of entity that the request is blocking.
REQ.PEND	ENTITY_TYPE_NAME	The 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 206](#).

Request Detail Tokens

Table A-30. Request Detail tokens (page 1 of 2)

Prefix	Tokens	Description
REQD	CREATED_BY	The ID of the user who created the request detail.
REQD	CREATION_DATE	The date on which the request detail was created.
REQD	LAST_UPDATED_BY	The ID of the user who last updated the request detail.
REQD	LAST_UPDATE_DATE	The date on which request detail was last updated.

Table A-30. Request Detail tokens (page 2 of 2)

Prefix	Tokens	Description
REQD	REQUEST_DETAIL_ID	The ID for the request detail in the table KCRT_REQUEST_DETAILS.
REQD	REQUEST_ID	The ID of the request for the request detail.
REQD	REQUEST_TYPE_ID	The ID of the request type for the request detail.

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

Request Detail > Field Tokens

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

Resource Pool Tokens

Table A-31. Resource Pool tokens (page 1 of 2)

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

Table A-31. Resource Pool tokens (page 2 of 2)

Prefix	Tokens	Description
RSCP	RSC_POOL_IS_FOR_NAME	The name of the program or org unit to which the resource pool is linked.
RSCP	RSC_POOL_NAME	The resource pool name.
RSCP	START_PERIOD	The resource pool start period.

Security Group Tokens

Table A-32. Security Group tokens

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

Skill Tokens

Table A-33. Skill tokens

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

Staffing Profile Tokens

Table A-34. Staffing Profile tokens (page 1 of 2)

Prefix	Tokens	Description
STFP	ACTIVE_FLAG	The active flag of the staffing profile.
STFP	CREATED_BY	The username of the user who created the staffing profile.
STFP	CREATION_DATE	The date on which the staffing profile was created.
STFP	DESCRIPTION	The description of the staffing profile.
STFP	END_PERIOD	The end period of the staffing profile.
STFP	PERIOD_SIZE	The period size of the staffing profile.
STFP	STAFFING_PROFILE_URL	The URL to view this staffing profile.
STFP	STAFF_PROF_ID	The ID of the staffing profile in table KRSC_STAFF_PROFS.
STFP	STAFF_PROF_IS_FOR_ENTITY_NAME	The entity name to which the staffing profile is linked.

Table A-34. Staffing Profile tokens (page 2 of 2)

Prefix	Tokens	Description
STFP	STAFF_PROF_IS_FOR_ID	The ID of the work plan, program or org unit to which the staffing profile is linked.
STFP	STAFF_PROFL_IS_FOR_NAME	The name of the work plan, program or org unit to which the staffing profile is linked (work plan, program, or org unit).
STFP	STAFF_PROF_NAME	The staffing profile name.
STFP	START_PERIOD	The staffing profile start period.
STFP	STATUS_CODE	The staffing profile status code.
STFP	STATUS_NAME	The staffing profile status name.

Step TXN (Transaction) Tokens

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

Prefix	Tokens	Description
WST	CONCURRENT_REQUEST_ID	The identifier for the Oracle Concurrent Request for the Workflow Step Transaction.
WST	CREATED_BY	The user ID of the user that created the Workflow Step Transaction.
WST	CREATION_DATE	The date the Workflow Step Transaction was created.
WST	ERROR_MESSAGE	Any system level error message associated with the Workflow Step Transaction.
WST	EXECUTION_BATCH_ID	The execution batch ID for the Workflow Step Transaction.
WST	HIDDEN_STATUS	The hidden status of the Workflow Step Transaction.
WST	LAST_UPDATED_BY	The user ID of the user that last updated the Workflow Step Transaction.
WST	LAST_UPDATED_BY_EMAIL	The email address of the user that last updated the Workflow Step Transaction.

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

Prefix	Tokens	Description
WST	LAST_UPDATED_BE_USERNAME	The username of the user that last updated the Workflow Step Transaction.
WST	LAST_UPDATE_DATE	The date the Workflow Step Transaction was last updated.
WST	STATUS	The status of the Workflow Step Transaction.
WST	STEP_TRANSACTION_ID	The transaction ID for the Workflow Step Transaction.
WST	TIMEOUT_DATE	The date of the last timeout on the Workflow Step Transaction.
WST	USER_COMMENT	Any comments a user added to the Workflow Step Transaction.
WST	WORKFLOW_ID	The workflow ID for the Workflow Step Transaction.
WST	WORKFLOW_STEP_ID	The workflow step ID for the Workflow Step Transaction.
WST	NEW_HIDDEN_STATUS	The new hidden status of the Workflow Step Transaction.
WST	OLD_HIDDEN_STATUS	The old hidden status of the Workflow Step Transaction.
WST	NEW_STATUS	The new status of the Workflow Step Transaction.
WST	OLD_STATUS	The old status of the Workflow Step Transaction.

System Tokens

Table A-36. System tokens

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

Task Tokens

Table A-37. Tasks tokens (page 1 of 4)

Prefix	Tokens	Description
TSK	ACTUAL_DURATION	The actual duration of the task.
TSK	ACTUAL_EFFORT	The actual effort associated with the task.
TSK	ACTUAL_FINISH_DATE	The actual finish date of the task.
TSK	ACTUAL_START_DATE	The actual start date of the task.
TSK	CONFIDENCE_CODE	The code of the confidence value entered by the user.

Table A-37. Tasks tokens (page 2 of 4)

Prefix	Tokens	Description
TSK	CONFIDENCE_NAME	The name of the confidence value entered by the user.
TSK	CONSTRAINT_DATE	The task's constraint date.
TSK	CREATED_BY	The user who created the task.
TSK	CREATED_BY_EMAIL	The email address of the user who created the task.
TSK	CREATED_BY_USERNAME	The username of the person who created the task.
TSK	CREATION_DATE	The creation date of the task.
TSK	DEPARTMENT_CODE	The code of the department value entered by the user.
TSK	DEPARTMENT_NAME	The name of the department value entered by the user.
TSK	DESCRIPTION	The description of the task.
TSK	ESTIMATED_REMAINING_DURATION	The estimated remaining duration of the task.
TSK	ESTIMATED_REMAINING_EFFORT	The estimated remaining effort involved in the task.
TSK	ESTIMATED_FINISH_DATE	The estimated finish date of the task.
TSK	HAS_EXCEPTIONS	The flag to show whether or not the task has exceptions.
TSK	LAST_UPDATE_DATE	The date on which the task was last updated.
TSK	LAST_UPDATED_BY	The last user to update the task.
TSK	LAST_UPDATED_BY_EMAIL	The email address of the last person to update the task.
TSK	LAST_UPDATED_BY_USERNAME	The username of the last person to update the task.
TSK	MASTER_PROJECT_ID	The ID of the master project.
TSK	MASTER_PROJECT_NAME	The name of the master project.
TSK	MOST_RECENT_NOTE_AUTHOR_FULL_NAME	First and last name of the author of the most recent note.

Table A-37. Tasks tokens (page 3 of 4)

Prefix	Tokens	Description
TSK	MOST_RECENT_NOTE_AUTHOR_USERNAME	Username of the author of the most recent note.
TSK	MOST_RECENT_NOTE_AUTHORED_DATE	Date of the most recent note.
TSK	MOST_RECENT_NOTE_TEXT	Text of the most recent note.
TSK	PARENT_PROJECT_ID	The ID of the parent work plan.
TSK	PARENT_PROJECT_NAME	The name of the parent work plan.
TSK	PERCENT_COMPLETE	The task's completed percentage.
TSK	PRIORITY	The priority of the task.
TSK	PROJECT_PATH	The work plan path. Hierarchy of parent work plans that contain this task.
TSK	PROJECT_TEMPLATE	The name of the project template used to create the work plan containing the task.
TSK	PROJECT_TYPE_CODE	Returns TASK for tasks and PROJECT for work plans.
TSK	RESOURCE_ID	The ID of the resource assigned to the task.
TSK	RESOURCE_EMAIL	The email address of the resource.
TSK	RESOURCE_GROUP_ID	The ID of the resource group assigned to the task.
TSK	RESOURCE_GROUP_NAME	The name of the resource group assigned to the task.
TSK	RESOURCE_USERNAME	The username of the resource.
TSK	SCHEDULED Effort	The scheduled effort involved in the task.
TSK	SCHEDULED_DURATION	The duration scheduled for the task.
TSK	SCHEDULED_FINISH_DATE	The finish date scheduled for the task.
TSK	SCHEDULED_START_DATE	The start date scheduled for the task.
TSK	SCHEDULING CONSTRAINT	The scheduling constraint for the task.
TSK	TASK_ID	The number that uniquely identifies the task (same as TASK_NUMBER). This corresponds to the PROJECT_ID column in table KDRV_PROJECTS.

Table A-37. Tasks tokens (page 4 of 4)

Prefix	Tokens	Description
TSK	TASK_NAME	The task name.
TSK	TASK_NAME_LINK	Standard hyperlink to the task in HTML-formatted notifications.
TSK	TASK_NUMBER	The number that uniquely identifies the task (same as TASK_ID).
TSK	TASK_STATE	The task state.
TSK	TASK_URL	The URL for the task Detail page.
TSK	WORKBENCH_TASK_URL	The URL to access this task in the Workbench.

Tasks > Pending Tokens

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

Prefix	Tokens	Description
TSK.PEND	ID	The ID of the entity that is being blocked by the task.
TSK.PEND	NAME	The name of the entity that is being blocked by the task.
TSK.PEND	DETAIL	Detail information for the entity that is being blocked by the task as shown in the References field.
TSK.PEND	DESCRIPTION	The description of the entity that is being blocked by the task.
TSK.PEND	STATUS_ID	The ID of the state or the code of the status of the entity that is being blocked by the task.
TSK.PEND	STATUS_NAME	The name of the status (or state) of the entity that is being blocked by the task.
TSK.PEND	STATE	The name of the state of the entity that is being blocked by the task.
TSK.PEND	ASSIGNED_TO_USERNAME	The name of the assigned user (or resource) of the entity that is being blocked by the task.

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

Prefix	Tokens	Description
TSK.PEND	ASSIGNED_TO_USER_ID	The username of the assigned user (or resource) of the entity that is being blocked by the task.
TSK.PEND	ASSIGNED_TO_GROUP_NAME	The name of the assigned group (or resource group) of the entity that is being blocked by the task.
TSK.PEND	ASSIGNED_TO_GROUP_ID	The ID of the assigned group (or resource group) of the entity that is being blocked by the task.
TSK.PEND	RESOURCE_USERNAME	The name of the resource associated with the entity that is being blocked by the task.
TSK.PEND	RESOURCE_ID	The username of the resource (or assigned user) associated with the entity that is being blocked by the task.
TSK.PEND	RESOURCE_GROUP_NAME	The name of the resource group (or assigned user) associated with the entity that is being blocked by the task.
TSK.PEND	RESOURCE_GROUP_ID	The ID of the resource group (or assigned group) associated with the entity that is being blocked by the task.
TSK.PEND	PERCENT_COMPLETE	The current percent complete value associated with the entity that is being blocked by the task.
TSK.PEND	ENTITY_TYPE_ID	The ID of the type of entity that is being blocked by the task.
TSK.PEND	ENTITY_TYPE_NAME	The name of the type of entity that is being blocked by the task.

Time Management Notification Tokens

Table A-39. Time Management Notification tokens

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

User Tokens

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

Prefix	Tokens	Description
USR	AUTHENTICATION_MODE_CODE	The authentication mode for the user (such as LDAP).
USR	AUTHENTICATION_MODE_NAME	The authentication mode for the user (such as LDAP).
USR	COMPANY	The company that employs the user.
USR	COMPANY_NAME	The name of the company that employs the user.
USR	CREATED_BY	The ID of the user who created the user.
USR	CREATED_BY_FULL_NAME	The full name of the created by user.
USR	CREATED_BY_USERNAME	The Mercury IT Governance Center username of the user that created the user.
USR	CREATION_DATE	The date on which the user was created.
USR	DEPARTMENT_CODE	The lookup code of the department the user belongs to (lookup type = DEPT).
USR	DEPARTMENT_NAME	The name of the department to which the user belongs.
USR	EMAIL_ADDRESS	The email address of the user.

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

Prefix	Tokens	Description
USR	END_DATE	The date on which the user is made inactive in the application.
USR	FIRST_NAME	The first name of the user.
USR	LAST_NAME	The last name of the user.
USR	LAST_UPDATED_BY	The ID of the user that last updated the user.
USR	LAST_UPDATE_BY_FULL_NAME	The full name of the last updated by user.
USR	LAST_UPDATED_BY_USERNAME	The Mercury IT Governance Center username of the user that last updated the user.
USR	LAST_UPDATE_DATE	The date the user was last updated.
USR	LOCATION_CODE	The lookup code of the user's location (lookup type = RSC - Location).
USR	LOCATION_NAME	The name of the user's location.
USR	MANAGER_USERNAME	The username of the user's manager.
USR	MANAGER_USER_ID	The ID of the user's manager.
USR	PASSWORD	The password for the user to use to log on to Mercury IT Governance Center. This value is encrypted.
USR	PASSWORD_EXPIRATION_DATE	The date the password needs to be reset for the user.
USR	PASSWORD_EXPIRATION_DAYS	The number of days until the password must be reset for the user.
USR	PHONE_NUMBER	The phone number of the user.
USR	PRIMARY_ROLE_ID	The ID of the primary role associated with the user.
USR	PRIMARY_ROLE_NAME	The name of the primary role associated with the user.
USR	REGION	Region associated with the user.
USR	REGIONAL_CALENDAR	The name of the regional calendar for the user.
USR	RESOURCE_CATEGORY_CODE	The lookup code of resource category (lookup type = RSC - Category) to which the user belongs.

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

Prefix	Tokens	Description
USR	RESOURCE_CATEGORY_NAME	The name of the category to which the user belongs.
USR	RESOURCE_TITLE_CODE	the lookup code of the user's resource title (lookup type = RSC - Resource Title).
USR	RESOURCE_TITLE_NAME	The name of the user's resource title.
USR	START_DATE	The date the user is made active in the application.
USR	USERNAME	The username for the user to use to log on to Mercury IT Governance Center.
USR	USER_ID	The ID of the user in the table KNTA_USERS.
USR	WORKLOAD_CAPACITY	The workload capacity of the user (% of FTE)

Validation Tokens

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

Prefix	Tokens	Description
VAL	COMPONENT_TYPE	The component type associated with the validation.
VAL	CREATED_BY	The ID of the user that created the validation.
VAL	CREATION_DATE	The date the validation was created.
VAL	DESCRIPTION	The description of the validation.
VAL	LAST_UPDATED_BY	The ID of the user that last updated the validation.
VAL	LAST_UPDATE_DATE	The date the validation was last updated.
VAL	LOOKUP_TYPE	The lookup type associated with the validation (if applicable).
VAL	VALIDATION_ID	The ID of the validation in the table KNTA_VALIDATIONS.

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

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

Validation > Value Tokens

Table A-42. Validation > Value tokens

Prefix	Tokens	Description
VAL.VALUE	CREATED_BY	The ID of the user that created the value.
VAL.VALUE	CREATION_DATE	The date the value was created.
VAL.VALUE	DEFAULT_FLAG	The flag to indicate whether the value is the default value for the associated lookup type.
VAL.VALUE	DESCRIPTION	The description of the value.
VAL.VALUE	ENABLED_FLAG	The flag that indicates whether the value is available for selection in a list.
VAL.VALUE	LAST_UPDATED_BY	The ID of the user that last updated the value.
VAL.VALUE	LAST_UPDATE_DATE	The date the value was last updated.
VAL.VALUE	LOOKUP_CODE	The code associated with the value.
VAL.VALUE	LOOKUP_TYPE	The value lookup type.
VAL.VALUE	MEANING	The meaning associated with the value.
VAL.VALUE	SEQ	The sequence relative to other values in the associated lookup type in which this value will be displayed in a drop-down list.

Workflow Tokens

Table A-43. Workflow tokens

Prefix	Tokens	Description
WF	CREATED_BY	The ID of the user that created the workflow.
WF	CREATION_DATE	The date the workflow was created.
WF	DESCRIPTION	The description of the workflow.
WF	ENABLED_FLAG	The flag indicating whether the workflow is enabled and available to use in packages and/or requests.
WF	FIRST_WORKFLOW_STEP_ID	The ID of the first workflow step in the workflow.
WF	FIRST_WORKFLOW_STEP_NAME	The name of the first workflow step in the workflow.
WF	ICON_NAME	The name of the workflow step icon.
WF	LAST_UPDATED_BY	The ID of the user that last updated the workflow.
WF	LAST_UPDATE_DATE	The date the workflow was last updated.
WF	PRODUCT_SCOPE_CODE	The validation value code for the product scope of the workflow.
WF	REOPEN_WORKFLOW_STEP_ID	The ID of the reopened workflow step.
WF	REOPEN_WORKFLOW_STEP_NAME	The name of the reopened workflow step.
WF	SUBWORKFLOW_FLAG	An indicator that specifies whether this workflow can be used as a Subworkflow.
WF	WORKFLOW_ID	The ID of the workflow defined in the table KWFL_WORKFLOWS.
WF	WORKFLOW_NAME	The name of the workflow.
WF	WORKBENCH_WORKFLOW_URL	The URL to open the workflow in the Workbench.

Workflow > Workflow Step Tokens

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

Prefix	Tokens	Description
WF.WFS	ACTION_BUTTON_LABEL	The label displayed on the package or request action button for the workflow step.
WF.WFS	AVERAGE_LEAD_TIME	The average lead time in days defined for the workflow step.
WF.WFS	CREATED_BY	The ID of the user that created the workflow step.
WF.WFS	CREATION_DATE	The date the workflow step was created.
WF.WFS	DESCRIPTION	The description of the workflow step.
WF.WFS	DEST_ENV_GROUP_ID	The ID of the destination environment group for the workflow step.
WF.WFS	DEST_ENV_GROUP_NAME	The name of the destination environment group for the workflow step.
WF.WFS	DEST_ENVIRONMENT_ID	The ID of destination environment for the workflow step.
WF.WFS	DEST_ENVIRONMENT_NAME	The name of the destination environment for the workflow step.
WF.WFS	ENABLED_FLAG	The flag indicating whether the workflow step is enabled and able to be traversed in a package or request.
WF.WFS	GL_ARCHIVE_FLAG	For GL object migration, the flag indicating whether to save the GL object being migrated to the Mercury GL Migrator archive.
WF.WFS	INFORMATION_URL	The workflow step's information URL.
WF.WFS	JUMP_RECEIVE_LABEL_CODE	The code for a Jump/Receive workflow step.
WF.WFS	JUMP_RECEIVE_LABEL_NAME	The name of a Jump/Receive workflow step.
WF.WFS	LAST_UPDATED_BY	The ID of the user that last updated the workflow step.
WF.WFS	LAST_UPDATE_DATE	The date the workflow step was last updated.

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

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

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

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

Workflow Step Tokens

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

Prefix	Tokens	Description
WFS	ACTION_BUTTON_LABEL	The label displayed on the package or request action button for the workflow step.
WFS	AVERAGE_LEAD_TIME	The average lead time in days defined for the workflow step.
WFS	CREATED_BY	The ID of the user that created the workflow step.
WFS	CREATION_DATE	The date the workflow step was created.
WFS	DESCRIPTION	The description of the workflow step.

Table A-45. Workflow Step tokens (page 2 of 4)

Prefix	Tokens	Description
WFS	DEST_ENV_GROUP_ID	The ID of the destination environment group for the workflow step.
WFS	DEST_ENV_GROUP_NAME	The name of the destination environment group for the workflow step.
WFS	DEST_ENVIRONMENT_ID	The ID of destination environment for the workflow step.
WFS	DEST_ENVIRONMENT_NAME	The name of the destination environment for the workflow step.
WFS	ENABLED_FLAG	The flag indicating whether the workflow step is enabled and able to be traversed in a package or request.
WFS	GL_ARCHIVE_FLAG	For GL object migration, the flag indicating whether to save the GL object being migrated to the Mercury GL Migrator archive.
WFS	INFORMATION_URL	The workflow step's information URL.
WFS	JUMP_RECEIVE_LABEL_CODE	The code for a Jump/Receive workflow step.
WFS	JUMP_RECEIVE_LABEL_NAME	The name of a Jump/Receive workflow step.
WFS	LAST_UPDATED_BY	The ID of the user that last updated the workflow step.
WFS	LAST_UPDATE_DATE	The date the workflow step was last updated.
WFS	OM_ARCHIVE_FLAG	For AOL object migration, the flag indicating whether to save the AOL object being migrated to the Object*Migrator archive.
WFS	PARENT_ASSIGNED_TO_GROUP_ID	The ID of the security group that the current package or request is assigned to (determined by context at time of evaluation).
WFS	PARENT_ASSIGNED_TO_GROUP_NAME	The security group that the current package or request is assigned to (determined by context at time of evaluation).

Table A-45. Workflow Step tokens (page 3 of 4)

Prefix	Tokens	Description
WFS	PARENT_ASSIGNED_TO_USERNAME	The name of the user that the current package or request is assigned to (determined by context at time of evaluation).
WFS	PARENT_ASSIGNED_TO_USER_ID	The ID of the user that the current package or request is assigned to (determined by context at time of evaluation).
WFS	PARENT_STATUS	The validation value code of the status of the request that is using the workflow step.
WFS	PARENT_STATUS_NAME	The validation value meaning of the status of the request that is using the workflow step.
WFS	PRODUCT_SCOPE_CODE	The validation value code for the product scope of the workflow containing the workflow step.
WFS	RESULT_WORKFLOW_PARAMETER_ID	The ID of the workflow parameter to which the result of the workflow step is written.
WFS	RESULT_WORKFLOW_PARAMETER_NAME	The name of the workflow parameter to which the result of the workflow step is written.
WFS	SORT_ORDER	The display sequence of the workflow step relative to all other steps in the workflow.
WFS	SOURCE_ENV_GROUP_ID	The ID of the source environment group for the workflow step.
WFS	SOURCE_ENV_GROUP_NAME	The name of the source environment group for the workflow step.
WFS	SOURCE_ENVIRONMENT_ID	The ID of the source environment for the workflow step.
WFS	SOURCE_ENVIRONMENT_NAME	The name of the source environment for the workflow step.
WFS	STEP_NAME	The workflow step name.
WFS	STEP_NO	The display sequence of the workflow step relative to all other steps in the workflow.
WFS	STEP_SOURCE_NAME	The name of the workflow step source.

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

Prefix	Tokens	Description
WFS	STEP_TYPE_NAME	The name of the workflow step source type.
WFS	WORKFLOW_ID	The ID of the workflow containing the workflow step.
WFS	WORKFLOW_NAME	The name of the workflow containing the workflow step.
WFS	WORKFLOW_STEP_ID	The 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 Mercury IT Governance Center best practice functionality. You will only have access to field groups associated with products that are licensed at your site.

CMDB Application Tokens

Table A-46. CMDB Application tokens

Prefix	Tokens	Description
REQ.P	KNTA_CMDB_APPLICATION	The CMDB application referenced by the request.

Demand Management SLA Tokens

Table A-47. Demand Management SLA tokens

Prefix	Tokens	Description
REQ.P	KNTA_SLA_LEVEL	SLA Level.
REQ.P	KNTA_SLA_VIOLATION_DATE	SLA Violation Data.
REQ.P	KNTA_SLA_SERV_REQUESTED_ON	Service Request Date.
REQ.P	KNTA_SLA_SERV_SATISFIED_ON	Service Satisfied Date.

Demand Management Scheduling Tokens

Table A-48. Demand Management Scheduling tokens

Prefix	Tokens	Description
REQ.P	KNTA_EST_START_DATE	Estimated Start Date.
REQ.P	KNTA_EFFORT	Estimated Effort.
REQ.P	KNTA_REJECTED_DATE	Reject Date.
REQ.P	KNTA_DEMAND_SATISFIED_DATE	Demand Satisfied Date.

MAM Impact Analysis Tokens

Table A-49. MAM Impact Analysis tokens

Prefix	Tokens	Description
REQ.P	KNTA_MAM_RFC_ID	MAM RFC ID number.
REQ.P	KNTA_MAM_IMPACT_RESULT	MAM impact results.

Portfolio Management Asset Tokens

Table A-50. Portfolio Management Asset tokens

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

Portfolio Management Project Tokens

Table A-51. Portfolio Management Project tokens

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

Portfolio Management Proposal Tokens

Table A-52. Portfolio Management Proposal tokens

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

Program Issue Tokens

Within token builder, Program Issue is an empty folder.

Program Reference Tokens

Table A-53. Program Reference tokens

Prefix	Tokens	Description
REQ.P	KNTA_PROGRAM_REFERENCE	Program Reference.

Project Issue Tokens

Table A-54. Project Issue tokens

Prefix	Tokens	Description
non-app	KNTA_ESCALATION_LEVEL	Escalation Level.

Project Reference Tokens

Table A-55. Project Issue tokens

Prefix	Tokens	Description
REQ.P	KNTA_MASTER_PROJ_REF	Master Project Reference.

Project Risk Tokens

Table A-56. Project Issue tokens

Prefix	Tokens	Description
REQ.P	KNTA_IMPACT_LEVEL	Impact Level.
REQ.P	KNTA_PROBABILITY	Probability Level.

Project Scope Change Tokens

Table A-57. Project Scope Change tokens

Prefix	Tokens	Description
non-app	KNTA_CR_LEVEL	Critical Level.
non-app	KNTA_IMPACT_LEVEL	Impact Level.

Quality Center Defect Information Tokens

Table A-58. Quality Center Defect Information tokens

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

Quality Center Information Tokens

Table A-59. Quality Center Information tokens (page 1 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_QC_DOMAIN	Quality Center domain name.
REQ.P	KNTA_QC_ASSIGNED_TO	Quality Center assigned to user.
REQ.P	KNTA_QC_PROJECT	Quality Center project.
REQ.P	KNTA_QC_REQUIREMENT_NO	Quality Center requirement number.
REQ.P	KNTA_QC_REQUIREMENT_STATUS	Quality Center requirement status.
REQ.P	KNTA_QC_REQUIREMENT_ATT_URL	Quality Center requirement ATT URL.

Table A-59. Quality Center Information tokens (page 2 of 2)

Prefix	Tokens	Description
REQ.P	KNTA_QC_REQUIREMENT_INT_MSG	Quality Center requirement instant messaging.
REQ.P	KNTA_QC_INSTANCE	Quality Center instance.
REQ.P	KNTA_QC_DASHBOARD_SUBJECT	Quality Center dashboard subject.
REQ.P	KNTA_QC_REQUIREMENT_COVERAGE	Quality Center requirement coverage.
REQ.P	KNTA_QC_OPEN_DEFECTS	Quality Center open defects.

Resource Management Work Item Tokens

Table A-60. Resource Management Work Item tokens

Prefix	Tokens	Description
REQ.P	KNTA_USR_SCHED_START_DATE	Scheduled Start Date
REQ.P	KNTA_USR_ACTUAL_START_DATE	Actual Start Date
REQ.P	KNTA_USR_SCHED_FINISH_DATE	Scheduled Finish Date
REQ.P	KNTA_USR_ACTUAL_FINISH_DATE	Actual Finish Date
REQ.P	KNTA_SCHED_DURATION	Scheduled Duration
REQ.P	KNTA_ACTUAL_DURATION	Actual Duration
REQ.P	KNTA_SCHED_EFFORT	Scheduled Effort
REQ.P	KNTA_ACTUAL_EFFORT	Actual Effort
REQ.P	KNTA_WORKLOAD	Workload
REQ.P	KNTA_WORKLOAD_CATEGORY	Workload Category
REQ.P	KNTA_ROLE	Role
REQ.P	KNTA_ALLOW_EXTERNAL_UPDATE	Allow External Update of Actual Effort
REQ.P	KNTA_SCHED_START_DATE	Scheduled Start Date
REQ.P	KNTA_ACTUAL_START_DATE	Actual Start Date
REQ.P	KNTA_SCHED_FINISH_DATE	Scheduled Finish Date
REQ.P	KNTA_ACTUAL_FINISH_DATE	Actual Finish Date
REQ.P	KNTA_SCHED_EFF_OVER_DUR	Scheduled Effort Over Duration

Service Catalog Tokens

Table A-61. Service Catalog tokens

Prefix	Tokens	Description
REQ.P	KNTA_SRC_CATALOG_NAME	The catalog name of the service catalog.
REQ.P	KNTA_SRC_SERVICE_NAME	The service name of the service catalog.
REQ.P	KNTA_SRC_SERVICE_ID	The service ID of the service catalog order.
REQ.P	KNTA_SRC_REQUESTOR	The requestor for the service catalog order.
REQ.P	KNTA_SRC_REQUESTED_FOR	The requestor for of the service catalog order.
REQ.P	KNTA_SRC_QUANTITY	The quantity of the service item in the service catalog order.
REQ.P	KNTA_SRC_PRICE	The price of the service item in the service catalog order.
REQ.P	KNTA_SRC_TIME_EXTIMATE	The time estimate for the service item in the service catalog order.
REQ.P	KNTA_SRC_ORDER_ID	The order ID of the service catalog order.
REQ.P	KNTA_SRC_SEQ	The sequence number of the service item in the service catalog order.

A

- application server properties 135
- auto-complete
 - command with delimited output 95
 - command with fixed width output 97
 - configuring the values 94
 - example 101
 - list of users 94
 - long lists 84
 - search fields 89
 - short lists 82
 - user-defined multi-select 99

C

- command conditions 23, 34
 - examples 23
- command language 22, 33
- command with delimited output 95
- command with fixed width output 97
- commands
 - create new command 26, 38
 - edit existing command 26, 38
 - remove 27, 38
 - special 22

- triggering from workflow 20
- validation 82
- component types 69
 - auto-complete 99
 - directory chooser 115
 - file chooser 116
 - file chooser (static environment override) 117
 - file chooser (token-based environment override) 117
- contact 135, 137
- creating
 - custom data masks 114
- currency data mask 108
- custom data masks
 - creating 114

D

- date field
 - valid format 119
- directory chooser 115
- dynamic list validations 79
 - command 82
 - SQL 79

E

- entity token
 - application server properties 135
 - command execution 164
 - contacts 135, 137
 - demand management fields 207
 - distributions 138
 - document management 139
 - environment applications 143
 - environments 140, 145
 - extension 64
 - notifications 165, 166, 179
 - organization units 167
 - package lines 170, 173
 - package pending 171
 - program 174, 178
 - project plan 174
 - releases 178
 - requests 180
 - requests pending 184
 - resource pools 186
 - security groups 187
 - skills 188
 - staffing profile 188, 189, 196
 - tasks 191
 - tasks pending 194
 - users 196
 - workflow steps 201, 203
 - workflows 200
- entity tokens
 - validation values 199
 - validations 198

F

- field group tokens 206
 - asset 208
 - demand management 207
 - PMO 206, 207, 212
 - program reference 211
 - project 209
 - proposal 210
 - work item 213, 214

fields

- preview layout 93
- file chooser 116
 - static environment override 117
 - token-based environment override 117
- format masks
 - for text fields 105

N

- nesting tokens 56
- New Command window 26, 38
- numeric data mask 107

O

- object types
 - commands and workflow 20

P

- percentage data mask 111

R

- request field tokens 59
 - prefixes 59
 - table components 59

S

- special command
 - parameters tab 32
- special command builder 35
 - using to build steps 42
- special commands 22
 - about 32
 - building steps with command builder 42
 - nesting 43
 - using 41
- SQL validations 79
 - tips 81
- static list validations 77
- swap mode 93

system special commands 22

T

table component validations 120

column totals 129

creating rules 125

defining 121

rules example 125

tokens 129

table components

using tokens in 59

telephone data mask 112

text fields

configuring 104

currency 106

custom format 106

format masks 105

numeric 105

percentage 106

telephone 106

token

evaluation example 101

Token Builder window 50

token evaluation 49

tokens

building 56, 65

default format 54

environment tokens 63

explicit entity format 55

field groups 206

formats 51

nesting within tokens 56

overview 48

parameter format 58

request fields 59

sub-entity format 62

user data format 57

where to use 48

U

user-defined special commands 22

V

validations

command 82

command with delimited output 95

command with fixed width output 97

creating 74

date format 119

defined 68

directory chooser 115

dynamic list 79

file chooser 116

file chooser (static environment
override) 117

file chooser (token-based environment
override) 117

list of all 73

overview 69

package and request group 71

seeded 73

special characters and 73

SQL 79

SQL tips 81

static lists 77

system 73

table component 120

text area 1800 120

