Mercury™ IT Governance Center

Commands and Tokens Guide and Reference

Version 5.5.0

MERCURY INTERACTIVE

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA
Tel: (408) 822-5200
Fax: (408) 822-5300

If you have any comments or suggestions regarding this document, please send them via e-mail to documentation@merc-int.com.

Publication Number: CommandsAndTokens-0304A

Table of Contents

Chapter 1 Introduction

Commands and Tokens are used throughout the Mercury IT Governance (ITG) Center implementation to enable advanced automation and defaulting.

Commands define the heart of the execution layer within the deployment system and determine which steps to execute at a specific Workflow step. This can involve activities such as migrating a file, executing a script, performing some data analysis, or compiling code.

Tokens are variables that can be used to reference information that is undefined until Mercury ITG Center is actually used in a particular context. This includes such things as setting variables in commands or using Tokens within Notifications to specify the recipients.

About This Document

This document provides information on using commands and Tokens. Each chapter or appendix covers specific topics on commands and Tokens:

Using Commands	Provides an overview and examples for using commands.
Special Commands	Discusses the interface for creating, editing and using Special Commands in the Mercury ITG Center.
Using Tokens	Provides an overview of how to use Tokens
System Special Commands	Discusses pre-defined Special Commands.
Tokens	Provides a list of all entity Tokens.

Intended Audience

The intended audience for this document include:

- Configuration experts configuring a deployment system.
- Configuration experts configuring a Request resolution system.
- Business modelers who need to modify the following entities: Workflows, Object Types, Request Types, Validations, Notifications, and Report Types.

Note

Users must have the proper Power license to access the screens and windows described in this document.

Document Conventions

Table 1-1 lists the types of conventions used in this document.

Table 1-1. Document conventions

Convention	Description	Example
Button, menu, tabs	Names of interface components that can be clicked (such as buttons, menus, and tabs) are shown in bold.	**Apply** button
Fields, Windows, Pages	Names of windows, fields, and pages are shown as displayed.	New Request window
`Code`	Code input and output are shown as displayed.	`CauchoConfigFile C:/ITG_Home/conf/ resin.conf`
Link	Linked URLs, filenames, and cross references are shown as blue italicized text.	*www.merc-int.com*
Variable	Variables are shown as italicized text.	*ITG_Home*/bin directory

Table 1-1. Document conventions

Convention	Description	Example
Note	Used to identify note boxes that contain additional information.	Note
Caution	Used to identify caution boxes that contain important information. Follow the instructions in all caution boxes, failure to do so may result in loss of data.	Caution
Example	Used to identify example boxes that contain examples of related procedure.	Example

Additional Resources

Mercury Interactive provides the following additional resources to help you successfully use commands and Tokens:

- *Related Documentation*
- *Customer Support*
- *Education Services*

Related Documentation

The Library includes additional documents related to the topics discussed in this guide. Access the Library through the Mercury ITG Center online help or the Download Center at *http://itg.merc-int.com/support/download/login.jsp*.

Using the Workbench	This document explains how to navigate through the Workbench interface.

Configuring a Request Resolution System	This document provides instructions for configuring a Request resolution system. This includes requirements gathering, modeling your processes in a Workflow, defining a Request Type to be integrated with the Workflow, and rolling out this system to your users.
Configuring a Deployment System (Change Management)	This document provides instructions for configuring a deployment system. This includes requirements gathering, modeling your processes in a Workflow, defining commands used by the execution engine, and rolling out this system to your users.
Configuring a Release Management System	This document provides details for configuring, defining and processing Releases.
Customizing the Standard Interface	This document provides details for customizing the interface, including the directory structure and methods of customization for changing the presentation of the standard interface.

Customer Support

Customer support and downloads for the Mercury ITG Center and additional product information can be accessed from the Mercury Interactive Support Web site at *http://support.mercuryinteractive.com*.

Education Services

Mercury Interactive provides a complete training curriculum to help you achieve optimal results using the Mercury IT Governance Center. For more information, visit the Education Services Web site at *http://www.merc-training.com/main/ITG*.

Chapter 2 Using Commands

The following sections provide an overview and examples for using commands in Mercury ITG Center:

- *Commands Overview*
- *Command Language*
- *Special Commands*
- *Command Steps*
- *Command Conditions*
- *Example Command Uses*

Commands Overview

Commands define the heart of the execution layer within the deployment system and determine which steps to execute at a specific Workflow step. This can involve activities such as migrating a file, executing a script, performing some data analysis, or compiling code.

This section contains the following topics:

- *Where Commands are Used*
- *Commands Interface*
- *Object Type Commands and Workflow*
- *Request Type Commands and Workflow*
- *Special Commands*

Where Commands are Used

Commands are used in the following entities to enhance the implementation and enable sophisticated command-line automation:

- Object Types
- Request Types
- Report Types
- Workflows
- Validations

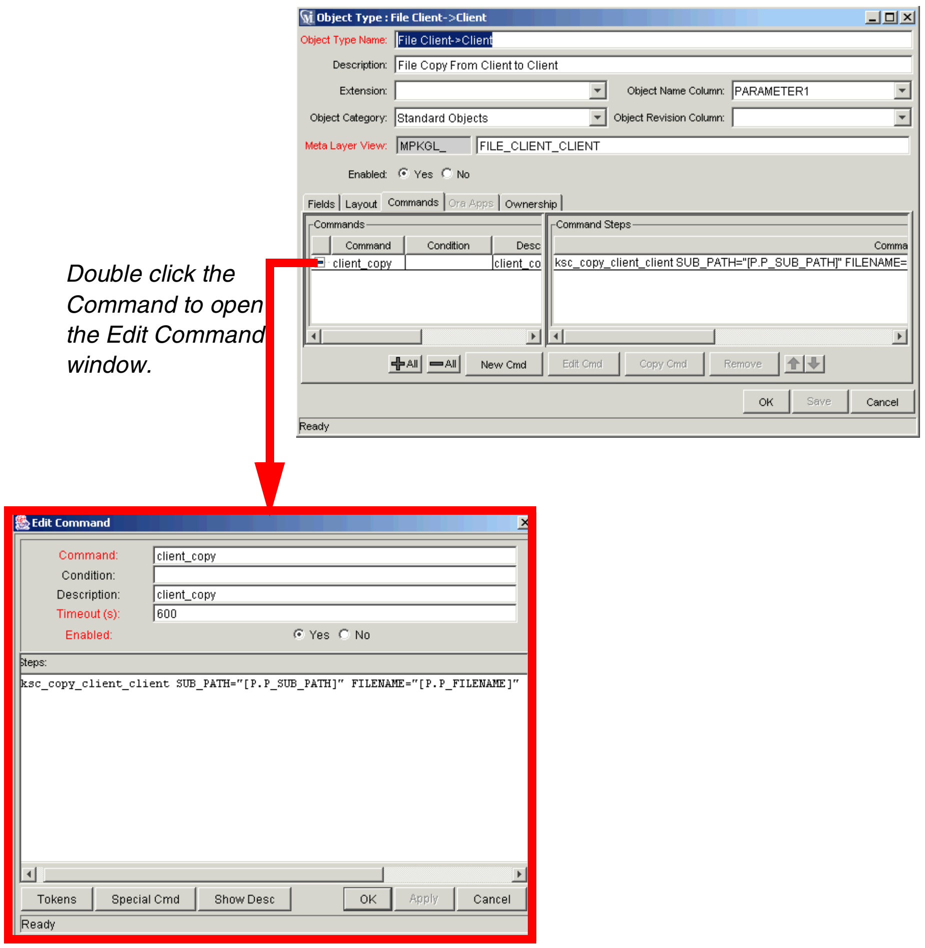
Commands Interface

Access commands through the **Commands** tab of the following screens:

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

Commands consist of command information and command steps. In this chapter, the examples are accessed through the Change Mgmt: Object Types screen, but the interface is the same in other screens where commands are configured.

Double-click the Command Step to open the Edit Command window. The Edit Command window displays the shell script code in the Steps window, as shown in *Figure 2-1*.



Object Type : File Client->Client

Object Type Name: File Client->Client
Description: File Copy From Client to Client
Extension:
Object Name Column: PARAMETER1
Object Category: Standard Objects
Object Revision Column:
Meta Layer View: MPKGL_ FILE_CLIENT_CLIENT
Enabled: [✓ Yes] No

Commands

Command	Condition	Desc
client_copy		client_co

Command Steps

Comma
ksc_copy_client_client SUB_PATH="[P.P_SUB_PATH]" FILENAME=

Ready

Edit Command

Command: client_copy
Condition:
Description: client_copy
Timeout (s): 600
Enabled: [✓ Yes] No

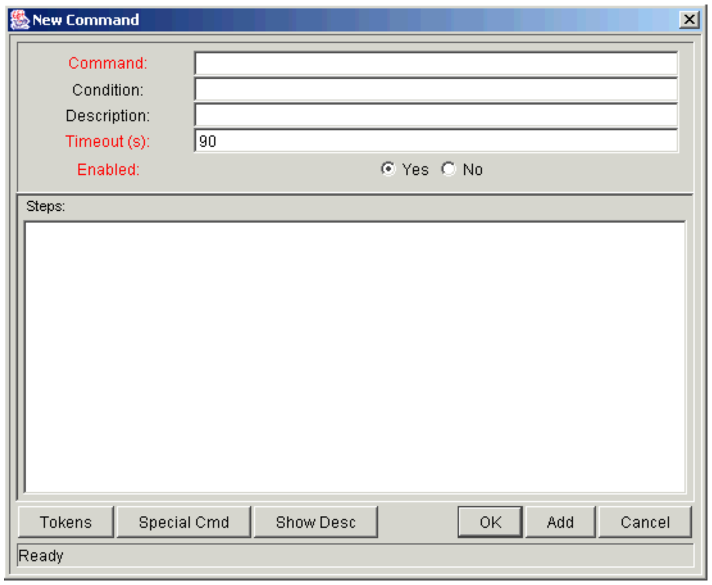
Steps:

```
ksc_copy_client_client SUB_PATH="[P.P_SUB_PATH]" FILENAME="[P.P_FILENAME]"
```

Ready

Figure 2-1 Commands Tab and Edit Command Window

To generate a new command, click **New Cmd** in the **Commands** tab. This opens the New Command window as shown in *Figure 2-2*. *Table 2-1* lists the fields contained in this window.



Timeout (s): 90

Enabled: [✓ Yes] No

Figure 2-2 New Command Window

Table 2-1. New Command Window Fields

Field	Description
Command	A simple name for the command.
Condition	A condition that determines whether the steps for the command are executed or not. (See *"Command Conditions"* on page 12 below for more information).
Description	A description of the command.
Timeout	The amount of time the command will be allowed to run before its process is terminated. This mechanism is used to abort commands that are hanging or taking an abnormal amount of time.
Enabled?	Determines whether the command is enabled for execution.

Each Object Type, Request Type, Validation, Workflow step source, or Report Type may have many commands, and each command may have many command steps. A command may be viewed as a particular function for an object. Copying a file may be one command, and checking that file into version control may be another. To perform these functions, a series of events needs to take place, and these events are defined in the command steps.

An additional level of flexibility is introduced when some commands must only be executed in certain cases. This is powered by the condition field of the commands and is discussed in *"Command Conditions"* on page 12.

Object Type Commands and Workflow

Object Type Commands are tightly integrated with the Workflow engine. The commands contained in an Object Type are executed at Execution Workflow steps in Change Management Package Lines.

It is important to note the following concepts regarding Command/Workflow interaction:

- To execute Object Type commands at a particular Workflow step, the Workflow step must be configured with the following parameters:
  - o Workflow step must be an Execution type step.
  - o Workflow Scope = **Packages**.
  - o Execution Type = **Built-in Workflow Event**.
  - o Workflow Command = **execute_object_commands**.
- When the object reaches the Workflow step (with Workflow Command = **execute_object_commands**), all Object Type commands whose conditions are satisfied will be run in the order they are entered in the Object Type's command panel.
- The Object Type can be configured to run only certain commands at a particular step. To do this, specify command conditions. For details, see *"Command Conditions"* on page 12.

Request Type Commands and Workflow

Similar to Object Type commands, Request Type commands define the execution layer within Request Management. While most of the resolution process for a Request is analytically based, cases may arise for specific Request Types where system changes are required. In these cases, Request Type commands can be used to automatically perform these changes.

Request Type Commands are tightly integrated with the Workflow engine. The commands contained in a Request Type are executed at Execution Workflow steps.

It is important to note the following concepts regarding Command/Workflow interaction:

- To execute Request Type commands at a particular Workflow step, the Workflow step must be configured with the following parameters:
    - Workflow step must be an Execution type step.
    - Workflow Scope = **Requests**
    - Execution Type = **Built-in Workflow Event**.
    - Workflow Command = **execute_request_commands**.
- When the Request reaches the Workflow step (with Workflow Command = **execute_request_commands**), all commands whose conditions are satisfied will be run in the order they are entered in the Request Type's command panel.
- The Request Type can be configured to run only certain commands at a particular step. To do this, specify command conditions. For details, see *"Command Conditions"* on page 12.

Special Commands

Object Types, Request Types, Report Types, Workflows and Validations all use commands to access the execution layer. In order to simplify the use of command executions, Mercury ITG Center contains 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 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 ITG Center features two types of Special Commands:

- System Special Commands - These commands are shipped with Mercury ITG Center. System Special Commands are read-only and have the naming convention "ksc_command_name". System Special Commands always begin with "ksc_".
- User Defined Special Commands - These commands are user-defined and have the naming convention "sc_command_name". User-defined Special Commands must begin with "sc_".

Special Commands act as sub-programs that can be reused where needed. It it often more convenient to create a Special Command for a program that will be used in multiple places, rather than placing the individual commands into every Object Type or Request Type that need them.

Command Steps

Command steps represent the actual directives that Mercury ITG Center specifies to execute the commands. A command step can be an actual command-line directive that is sent to the Mercury ITG Server or target machine, or it can be one of the many "Special Commands." *Table 2-2* describes the fields in the Command Steps region of the New/Edit Commands dialog.

Table 2-2. Command Steps

Field	Description
Steps	Defines the command-line directive or Special Command to be issued.
Description	Describes each of the command steps.

Note

The Execution Engine will execute the commands and command steps in the order they are displayed in the **Commands** tab. To change the order of the commands or the command steps, in the **Commands** tab, select the given command or command step and use the arrow buttons to move the selected item.

Command Language

The command steps in a command define the actual system-level executions that need to be performed to achieve the desired function of the command. Command steps can be UNIX commands, third party application commands, or Special Commands. Special commands are reusable routines defined in Mercury ITG Center. Mercury ITG Center also supplies a number of system Special Commands used to perform common execution events (such as connecting to Environments or copying files). Tokens can be used within command steps.

Command Conditions

In many situations, it may be necessary to run a different set of commands depending on the context of execution. This flexibility is achieved through the use of conditional commands. The Condition field for a command is used to define the situation under which the associated command steps execute.

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. 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 enormous flexibility when evaluating scenarios. Some example conditions are detailed in the following table:

Table 2-3. Example Conditions

Condition	Evaluates to
BLANK	Command will be executed in all situations.
‘[P.P_VERSION_LABEL]’ IS NOT NULL	Command will be executed if the parameter with the Token P_VERSION_LABEL in the Package line is not null.
‘[DEST_ENV.ENVIRONMENT_NAME]’ = ‘Archive’	Command will be executed when the destination Environment is named “Archive”.
‘[AS.SERVER_TYPE_CODE]’ = ‘UNIX’	Command will be executed if the application server is installed on a UNIX machine.

Tip

Be sure to place single quotes around string literals or Tokens that will evaluate strings.

The condition can include Tokens. For more information, see *“Using Tokens”* on page 37.

Example Command Uses

This section provides a number of operations that can be executed using commands. Sample code for configuring many of these cases is included in *"System Special Commands"* on page 57.

- Commands for connecting to machines.
  - o Connect to the destination Environment and run system commands
  - o Connect to an alternate Environment and run command (Environment override)
- Commands for manipulating data (fields and other information stored in files or database).
  - o Set a value in a Package Line
  - o Create, run and delete a script
  - o Extract information from a file (version number)
- Commands for running operating system-specific commands (NT and Unix).
  - o Starting a server
  - o Stopping a server
- Commands for running program-specific commands.
  - o Checking files in and out of a Version control system
- Commands for copying files.

Chapter 3 Special Commands

Object Types, Request Types, Report Types, Workflows and Validations all use commands to access the execution layer. In order to simplify the use of command executions, Mercury ITG Center contains 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 ITG Center features two types of Special Commands:

- System Special Commands - These commands are shipped with the Mercury ITG Center. System Special Commands are read-only and have the naming convention “ksc_command_name.” System Special Commands always begin with “ksc_.”
- User Defined Special Commands - These commands are user-defined and have the naming convention “sc_command_name.” User-defined Special Commands must begin with “sc_.”

This chapter discusses the interface for creating, editing and using Special Commands in Mercury ITG Center. The following topics are discussed:

- *“Special Command Interface”* on page 16
- *“Creating and Editing Special Commands”* on page 25
- *“Using Special Commands”* on page 32

See *“System Special Commands”* on page 57 for a detailed description of System Special Commands and their parameters.

Special Command Interface

Use the Special Command interface to create, view and edit Special Commands. The Special Command interface consists of the Special Command Workbench and Special Command window, as shown in *Figure 3-1*. To access the Special Command interface, click **Configuration** in the shortcut bar and click the **Special Commands** icon.

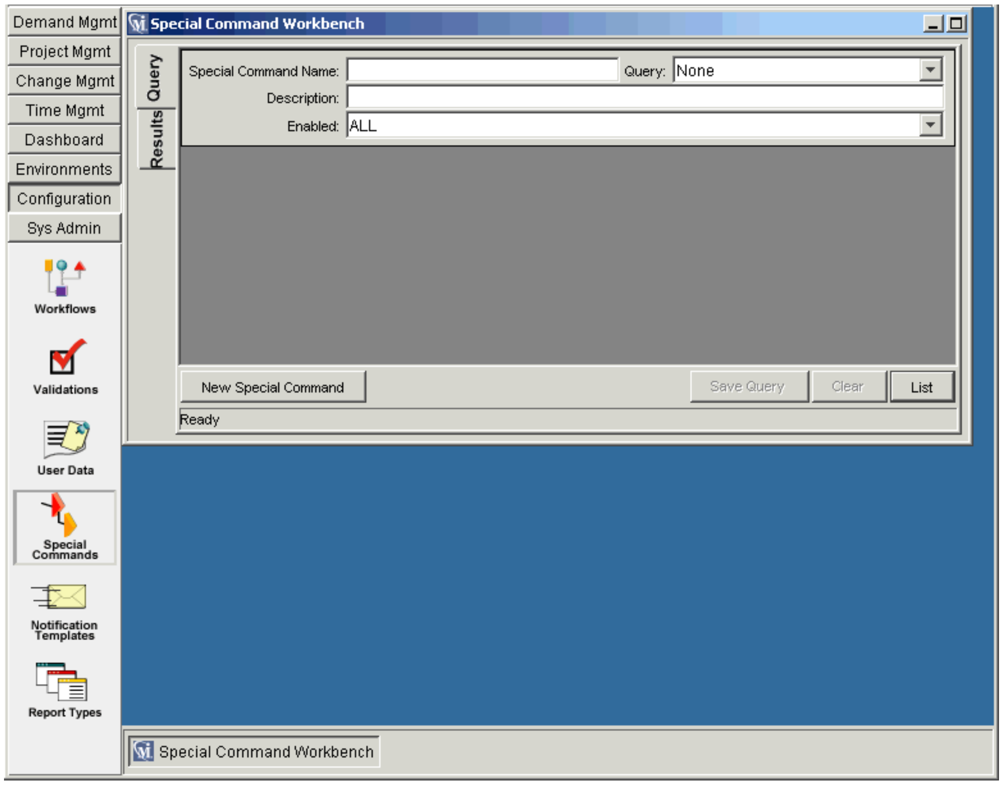
This section details the Special Command interface and discusses the following topics:

- *Special Command Workbench*
- *Special Command Window*

Special Command Workbench

Use the Special Command Workbench to search for a particular Special Command in the **Query** tab using the following criteria:

- Special Command Name - Filter for Special Commands where the name matches a given string.
- Description - Filter for Special Commands where the description matches a given string.
- Enabled - Filter for Special Commands that are enabled or disabled.



Special Command Name:
Query: None
Description:
Enabled: ALL

Figure 3-1 Special Command Workbench

Special Command Window

Use the Special Command window to define and configure Special Commands. As shown in *Figure 3-2*, the Special Command window consists of the following region and tabs:

- *Special Command General Information Region*
- *Parameters Tab*
- *Commands Tab*
- *Ownership Tab*
- *Used By Tab*

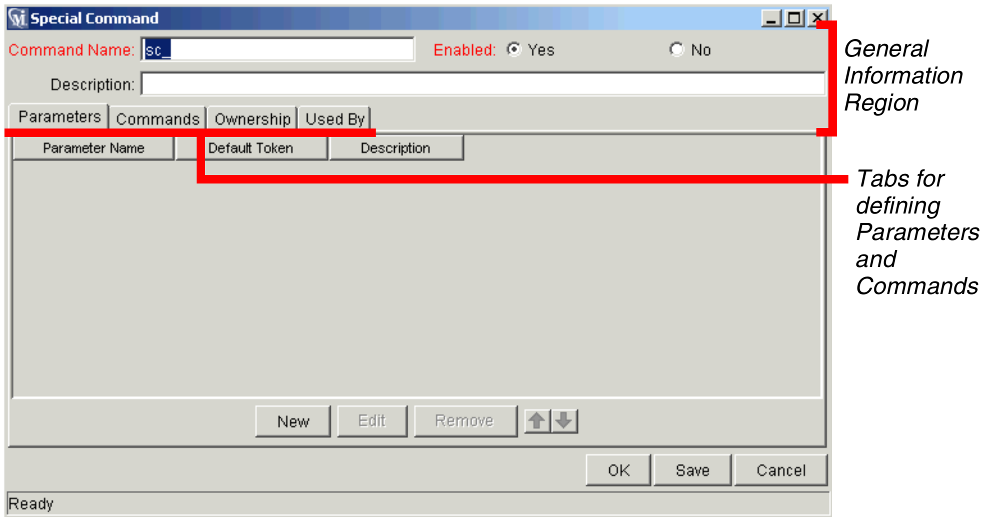


Figure 3-2 Special Command Window

Special Command General Information Region

The Special Command general information region displays the basic header information for the Special Commands. It consists of the fields described in *Table 3-1*.

Table 3-1. Special Commands Information Fields

Field			Description
Name	**Required**	**Type**	
Command Name	Y	Text Field	The name of the Special Command. This can only be updated when generating or editing a user-defined Special Command.
Enabled?	Y	Yes/No Radio Button	Determines whether or not the Special Command is enabled for use in Workflows, Object Types, Report Types, Request Types and Validations.
Description	N	Text Field	A description of the Special Command. This can only be updated when generating or editing a user-defined Special Command.

Parameters Tab

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.

Example

The ‘ksc_copy_server_server’ Special Command shown in this example 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 *“Special Command Builder”* on page 24.

Note

Custom Tokens are defined for specific Object Types, Request Types, and Report Types, and are referenced using the ‘[P.TOKEN_NAME]’ syntax. See *“System Special Commands”* on page 57 for a list of all predefined Special Command parameters and their default Tokens.

Commands Tab

Use the **Commands** tab to define and configure the commands and command steps used by each user-defined Special Command. It is also possible to view the command information for the predefined system Special Commands.

Commands are designed to have a similar look-and-feel to the UNIX and DOS operating system command structure. The specific parts of a command, the command steps, are often just command-prompt directives.

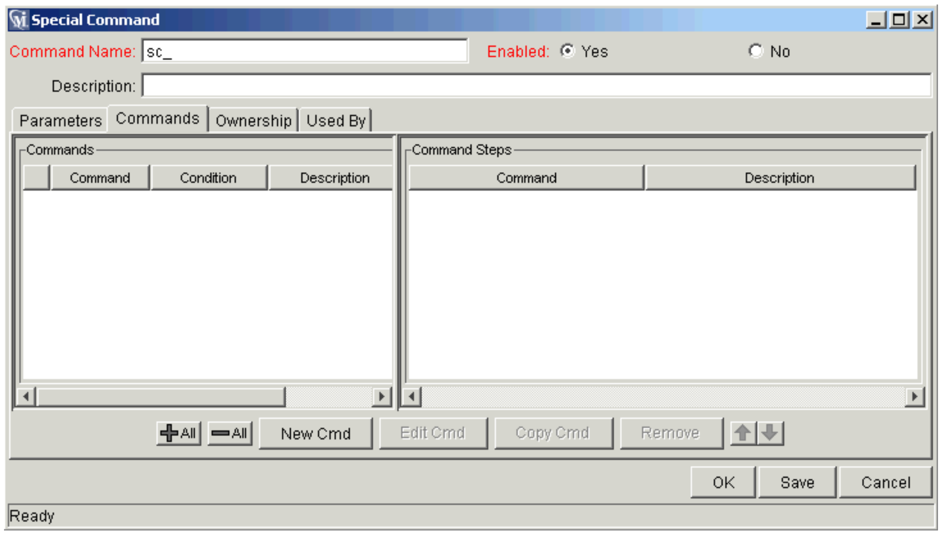


Figure 3-3 Special Commands - Commands Tab

Commands are accessible through the **Commands** tab of the Special Commands window and consist of command information and command steps.

Command Conditions

In many situations, it may be necessary to run a different set of commands depending on the context of execution. 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.

This flexibility is achieved through the use of conditional commands. The Condition field for an object command provides the ability to define the situation under which the associated command steps will execute.

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 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. Some example conditions are given in *Table 3-2*.

Table 3-2. Example Conditions

Condition	Evaluates to
BLANK	Command executes in all situations.

Table 3-2. Example Conditions

Condition	Evaluates to
'[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, strings must be enclosed by single quotes.

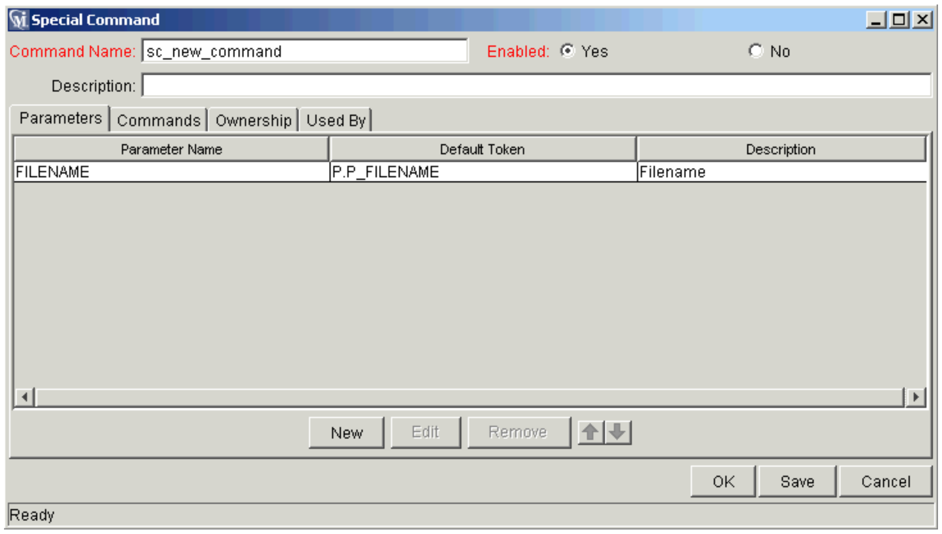
The condition can include a Token. See *"Using Tokens"* on page 37 for more information.

Parameters in Command Steps

In the command steps within a Special Command, parameters are referred to as their default Tokens. When the Special Command is executed with a value specified for a parameter, this value will replace the default Token throughout the Special Command steps.

Example - Special Command

An existing Special Command echoes a string as an HTML tag named sc_echo_html and takes the parameter RAW_TEXT. This example shows how to create another Special Command named sc_new_command. This Special Command will use sc_echo_html to echo the parameter value FILENAME, which has a default Token of [P.P_FILENAME].



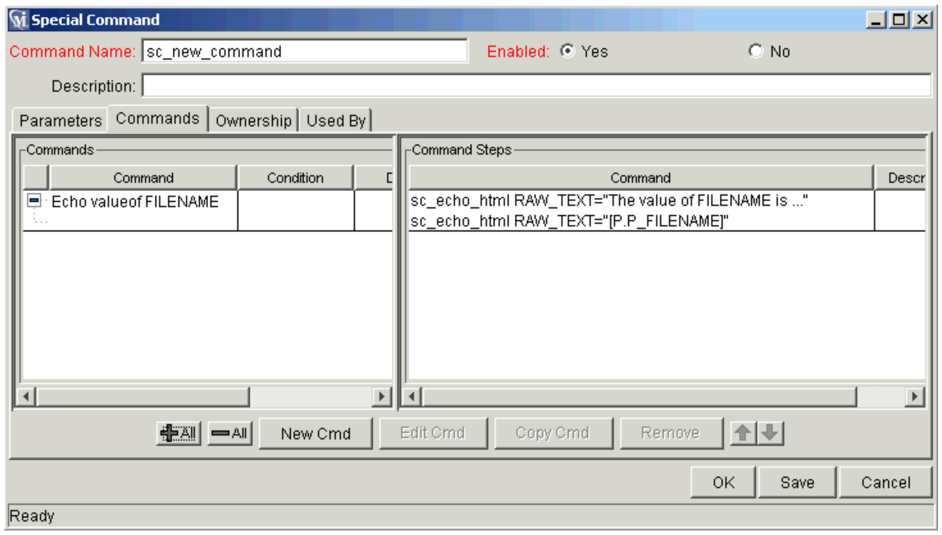
Command Name: sc_new_command Enabled: [✓ Yes] No

Description:

Parameter Name	Default Token	Description
FILENAME	P.P_FILENAME	Filename

To accomplish this, the following command steps are entered in a command for sc_new_command:

```
sc_echo_html RAW_TEXT="The value of FILENAME is..."
sc_echo_html RAW_TEXT="[P.P_FILENAME]"
```



Command Name: sc_new_command Enabled: [✓ Yes] No

Description:

Command	Condition
Echo valueof FILENAME	

Command	Descr
sc_echo_html RAW_TEXT="The value of FILENAME is ..."	
sc_echo_html RAW_TEXT="[P.P_FILENAME]"	

Note that the command step uses the default Token to refer to the value of the Special Command parameter. The parameter name is only used when invoking a Special Command.

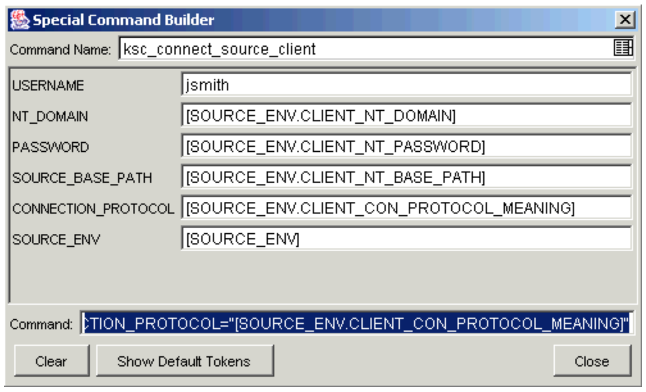
Note

Parameters cannot be used in command conditionals.

Continuing from the previous example, suppose that a Special Command has the parameter FILENAME, whose default Token is [P.P_FILENAME]. In command conditionals, the Token [P.P_FILENAME] will always be evaluated normally, regardless of whether our Special Command was called with a value for the parameter FILENAME.

Special Command Builder

The Special Command Builder is a tool designed to simplify the use of Special Commands by ensuring proper formatting of the Command Step. The Special Command Builder, shown in *Figure 3-4*, is an interface where a Special Command can be selected and appropriate parameters can be entered. The Special Command Builder outputs a line of text to the Command field which can be used as a command step.



Command Name: ksc_connect_source_client

USERNAME	jsmith
NT_DOMAIN	[SOURCE_ENV.CLIENT_NT_DOMAIN]
PASSWORD	[SOURCE_ENV.CLIENT_NT_PASSWORD]
SOURCE_BASE_PATH	[SOURCE_ENV.CLIENT_NT_BASE_PATH]
CONNECTION_PROTOCOL	[SOURCE_ENV.CLIENT_CON_PROTOCOL_MEANING]
SOURCE_ENV	[SOURCE_ENV]

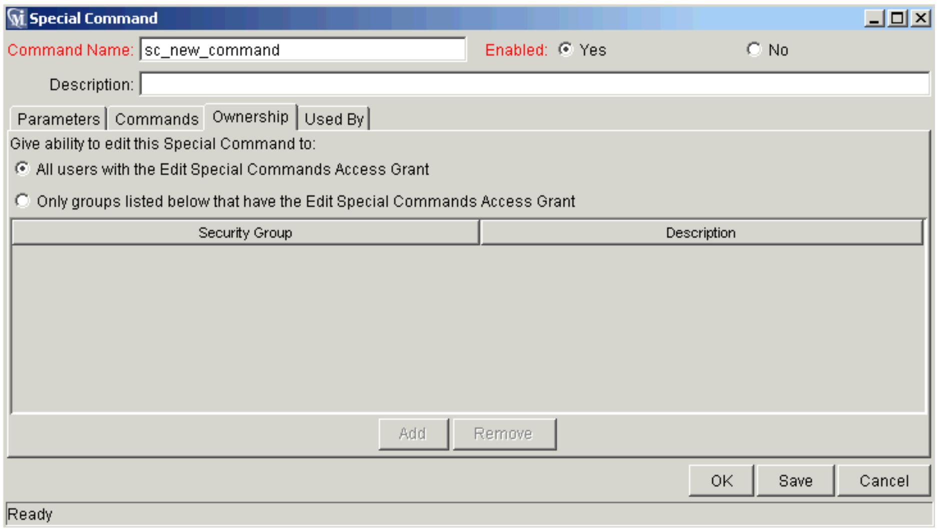
Command: TION_PROTOCOL="[SOURCE_ENV.CLIENT_CON_PROTOCOL_MEANING]"

Figure 3-4 Special Command Builder

Ownership Tab

The **Ownership** tab is used to select Ownership Groups for a specific Special Command. Members of Ownership Groups are the only users who have the right to edit, copy or delete this Special Command. This tab also displays Ownership Groups that have been linked to this entity. Ownership Groups can be deleted from this tab by selecting them and clicking **Remove**.

See *"Setting Ownership for Special Commands"* on page 31 for more information about setting Ownership for a new or existing Special Command.


Special Command

Command Name: sc_new_command
Enabled: (●) Yes ( ) No

Description:

Parameters | Commands | Ownership | Used By

Give ability to edit this Special Command to:

(●) All users with the Edit Special Commands Access Grant

( ) Only groups listed below that have the Edit Special Commands Access Grant

Security Group	Description

Add | Remove

OK | Save | Cancel

Ready

Figure 3-5 Ownership Tab

Used By Tab

Click the **Used By** tab to view a list of entities that currently refer to the selected Special Command.

Creating and Editing Special Commands

This section describes the following procedures for creating and editing Special Commands:

- *Creating a New Special Command*
- *Creating and Editing Special Command Parameters*
- *Adding Special Commands to Command Steps Using the Command Builder*

Creating a New Special Command

To create a new Special Command:

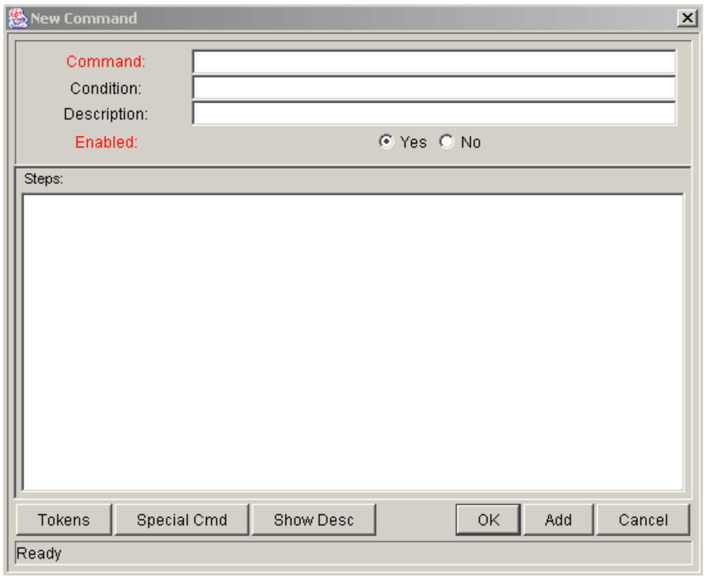
1. From the Special Command Workbench, click **New Special Command**.

   The Special Command window opens.

2. Click the **Commands** tab.

3. Click **New Cmd**.

   The New Command window opens. This window's fields are defined in *Table 3-3 on page 27*.



Enabled: [✓ Yes] No

4. Enter information in the Command, Condition and Description fields.

   See *"Command Conditions"* on page 21 for more details about defining Conditions.

5. Set the Enabled radio button to **Yes**.

6. Add Tokens to the new Special Command as desired.

   a. Click **Tokens**.

      The Token Builder window opens.

   b. Copy a Token from the Token Builder window.

   c. Paste it into the New Command window's Steps text area.

7. Add another Special Command to the new Special Command.

   a. Click **Special Cmd**.

      The Special Command Builder window opens.

b. In the Command Name field, select a Special Command and enter any required parameters.

c. Copy the Special Command from the Special Command Builder window.

d. Paste it into the New Command window's Steps text area.

8. To add the new command to the **Command** tab of the Special Command window without closing the New Command window, click **Add**.

9. To add the new command to the **Command** tab of the Special Command window and close the New Command window, click **OK**.

   The new Special Command has been created.

10. To save the new Special Command, click **Save**.

Table 3-3. New Command Window Fields

Field			Description
Name	**Required**	**Type**	
Command	Y	Text Field	The name of the command.
Condition	N	Text Field	A condition that determines whether the Command steps for the command are executed or not. (See *"Command Conditions"* on page 21 for more information).
Description	N	Text Field	A description of the command.
Enabled?	Y	Yes/No Radio Button	Determines whether the command is enabled for execution.

Creating and Editing Special Command Parameters

This section describes the following procedures for creating and editing Special Command parameters:

- *Adding Parameters to Special Commands*
- *Editing Special Command Parameters*
- *Deleting Parameters*

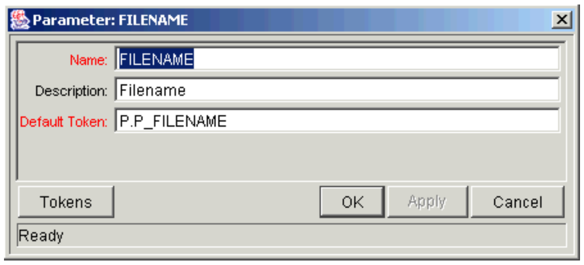
Adding Parameters to Special Commands

This section describes the procedure for adding parameters to a Special Command.

To add a new parameter to a user-defined Special Command:

1. In the **Parameters** tab of the Special Command window, click **New**.

   The Parameter window opens.



Parameter: FILENAME

Name: FILENAME

Description: Filename

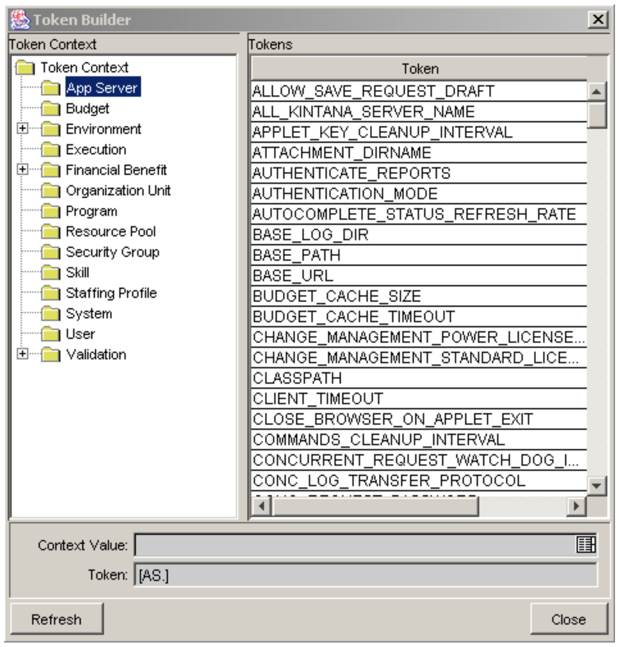
Default Token: P.P_FILENAME

2. Fill in the Name, Description and Default Token fields.

   To select an existing global Token, follow *Step 3* through *Step 9*. To manually entered a Token name in the Default Token field, go to *Step 7*.

3. To select an existing global Token, click **Tokens**.

   The Token Builder window opens.



ALLOW_SAVE_REQUEST_DRAFT
ALL_KINTANA_SERVER_NAME
APPLET_KEY_CLEANUP_INTERVAL
ATTACHMENT_DIRNAME
AUTHENTICATE_REPORTS
AUTHENTICATION_MODE
AUTOCOMPLETE_STATUS_REFRESH_RATE
BASE_LOG_DIR
BASE_PATH
BASE_URL
BUDGET_CACHE_SIZE
BUDGET_CACHE_TIMEOUT
CHANGE_MANAGEMENT_POWER_LICENSE...
CHANGE_MANAGEMENT_STANDARD_LICE...
CLASSPATH
CLIENT_TIMEOUT
CLOSE_BROWSER_ON_APPLET_EXIT
COMMANDS_CLEANUP_INTERVAL
CONCURRENT_REQUEST_WATCH_DOG_I...
CONC_LOG_TRANSFER_PROTOCOL

Token: [AS.]

4. In the Token Context pane of the window, select a folder.

   The available Tokens for each folder display in the Tokens pane of the window.

5. In the Token column, select a Token.

   When a Token is selected, it enables the Token field and displays the name of the selected Token (including its prefix).

6. Copy the Token.

   a. Select the Token in the Token field.

   b. Press Ctrl+C on the keyboard.

7. In the Parameter window, paste the Token name into the Default Token field by pressing Ctrl+V on the keyboard.

8. To add the field to the **Parameters** tab and close the Parameter window, click **OK**.

9. To add the field to the **Parameters** tab without closing the Parameter window, click **Add**.

Editing Special Command Parameters

This section describes the procedure for editing Special Command parameters.

To edit an existing parameter:

1. Open the Special Command.
2. In the **Parameters** tab, double-click the Parameter.

   The Parameter window opens.
3. Make the desired changes in the Parameter window.
4. Click **Apply** to apply the changes without closing the Parameter window.
5. Click **OK** to apply the changes and close the Parameter window.

Note

The parameter order can be altered by selecting a parameter in the **Parameters** tab and clicking either the **Up** or **Down** arrow.

Changes to parameters already used by existing Request Types, Object Types, or Report Types can affect the way these entities function.

Deleting Parameters

This section describes the procedure for deleting Special Command parameters.

To delete a parameter:

1. Open the Special Command.
2. Select the parameter in the **Parameters** tab.
3. Click **Remove**.
4. Click **OK** to save the information and close the Special Command window.
5. Click **Save** to save the information without closing the Special Command window.

The parameter is deleted from the Special Command.

Setting Ownership for Special Commands

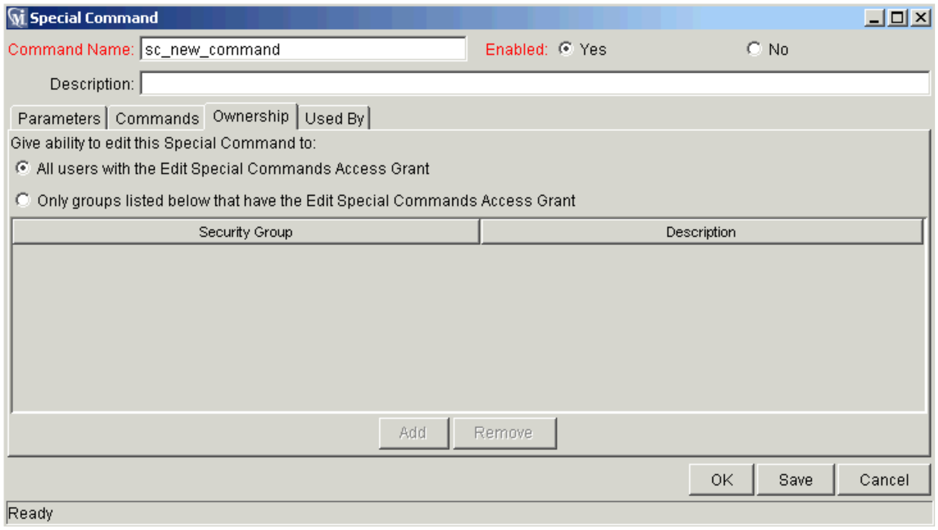
Different groups of users can have exclusive control over the Special Commands used by their group. These groups are referred to as Ownership Groups. Members of the ownership group are the only users who can edit, delete or copy the Special Commands. Each Special Command can be assigned multiple ownership groups.

Ownership groups are defined in the Security Group window in the Workbench. See *Security Model Guide and Reference* for instructions on setting up Security Groups.

To set the Ownership for a Special Command:

1. Open the Special Command window.

2. Click the **Ownership** tab.



Command Name: sc_new_command
Enabled: [✓ Yes] No
Description:
Give ability to edit this Special Command to:
[✓ All users with the Edit Special Commands Access Grant]
Only groups listed below that have the Edit Special Commands Access Grant
Ready

3. Select the Only groups listed below that have the Edit Special Commands Access Grant option.

4. Click **Add**.

   The Add Security Groups window opens.

5. In the Security Group auto-complete list, select a Security Group.

6. To close the Add Security Group window, click **OK**.

The selected Security Groups are display in the **Ownership** tab under the Security Group column.

7. To save the changes and close the window, click **OK** in the Special Command window.

   To save the selection and leave the Special Command window open, click **Save**

Only members of the Security Group(s) specified in the **Ownership** tab can edit, delete or copy this Special Command.

Note

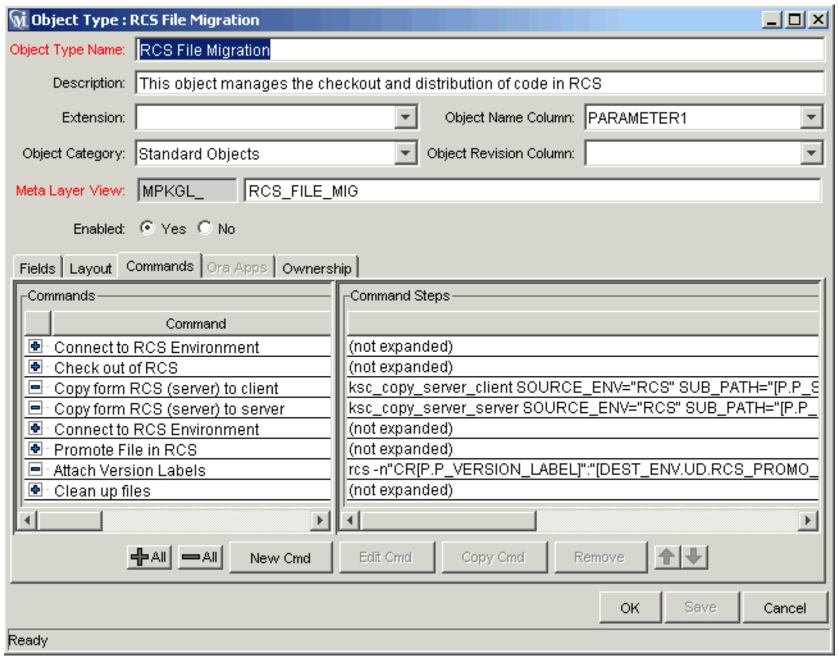
If no Ownership groups are associated with the entity, the entity is considered global and any user with the Edit Access Grant for the entity can edit, copy or delete it. For more information on Access Grants, see *Security Model Guide and Reference*.

By default, administrators have the 'Ownership Override' Access Grant and can access configuration entities even if the administrator is not a member of one of the Ownership Groups and does not have the Edit Access Grant.

If a Security Group is disabled or loses the Edit Access Grant, that group will no longer have edit access for the entity.

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 3-6* shows an Object Type that has been generated using a combination of Special Commands.



Object Type Name: RCS File Migration

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

Extension:

Object Name Column: PARAMETER1

Object Category: Standard Objects

Object Revision Column:

Meta Layer View: MPKGL_ RCS_FILE_MIG

Enabled: [✓ Yes] No

Commands: Command	Command Steps
Connect to RCS Environment	(not expanded)
Check out of RCS	(not expanded)
Copy form RCS (server) to client	ksc_copy_server_client SOURCE_ENV="RCS" SUB_PATH="[P.P_S
Copy form RCS (server) to server	ksc_copy_server_server SOURCE_ENV="RCS" SUB_PATH="[P.P_
Connect to RCS Environment	(not expanded)
Promote File in RCS	(not expanded)
Attach Version Labels	rcs -n"CR[P.P_VERSION_LABEL]":"[DEST_ENV.UD.RCS_PROMO_
Clean up files	(not expanded)

Ready

Figure 3-6 RCS File Migration Object Type

This section describes the following ways to use Special Commands:

- *Adding Special Commands to Command Steps Using the Command Builder*
- *Nesting Special Commands*

Adding Special Commands to Command Steps Using the 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. Go to the **Commands** tab for the entity which commands will be added.
2. Click **New Cmd** or edit an existing command.

   The Command window opens.
3. Click **Special Cmd**.

   The Special Command Builder window opens.
4. Enter the a command name in the Command Name field, or select it from the auto-complete list.

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

Note

Both predefined (ksc_command) and user defined (sc_command) Special Commands can be used to build the command steps line. For more information on generating Special Commands, see *"Special Command Interface"* on page 16.

5. Replace the associated default Token value with any desired parameter information.
   a. To view the default Tokens, click **Show Default Tokens**.
   b. To hide the default Tokens, click **Hide Default Tokens**.
6. When the parameters have been modified, select the text in the Command field.

   To copy the formatted Special Command, press Ctrl+C on the keyboard.
7. To close the Special Command Builder window, click **Close**.
8. To paste the Special Command step, click in the Steps text area of the New Command window and press Ctrl+V on the keyboard.

9. Fill in the remaining fields in the New Command window.

10. Set the Enabled radio button to **Yes**.

11. To add the command step to the **Command** tab, click **OK**.

The new Special Command is now ready to be used 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.

Chapter 4 Using Tokens

This chapter provides an overview of how to use Tokens. This chapter discusses the following topics:

- *What are Tokens?*
- *Where Tokens Are Used*
- *Token Builder Window Overview*
- *Token Formats*
- *Token Evaluation*

What are Tokens?

While configuring certain features, it is often necessary to reference information that is undefined until Mercury ITG Center is actually used a particular context. Instead of generating objects that are valid only in specific contexts, Mercury ITG Center uses variables to facilitate the creation of general objects that can be applied to a variety of contexts. These variables are called Tokens.

There are two types of Tokens found within Mercury ITG Center: custom Tokens and standard Tokens. Standard Tokens are provided with the product. Custom Tokens are generated to suit specific needs. Each field of the following entities can be referenced as a custom Token:

- Object Types
- Request Types and Request Header Types
- Report Types

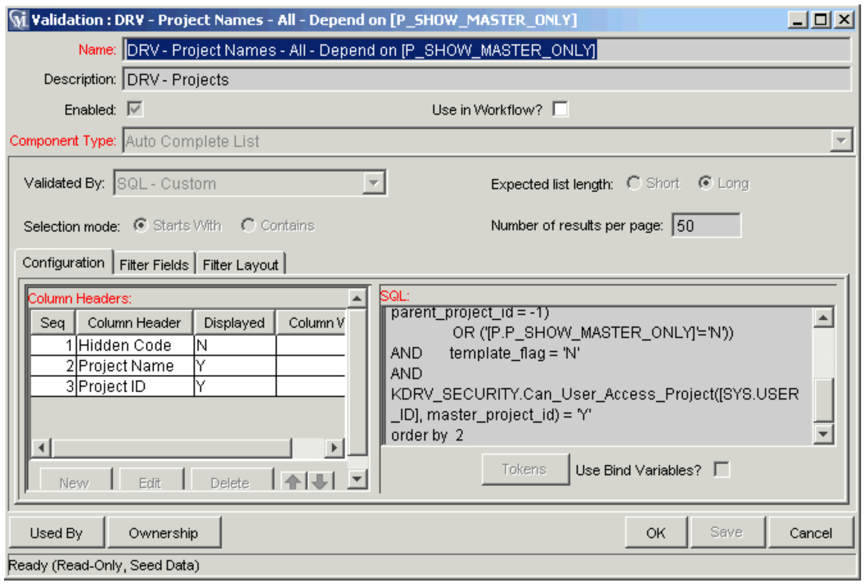
- User Data
- Workflow Parameters

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

Where Tokens Are Used

Tokens can be used 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 Submission
- Special Command commands
- Notifications for Tasks
- Notes for Requests Details



Name: DRV - Project Names - All - Depend on [P_SHOW_MASTER_ONLY]
Description: DRV - Projects
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

Column Headers:

Seq	Column Header	Displayed	Column W
1	Hidden Code	N	
2	Project Name	Y	
3	Project ID	Y	

SQL:

```
parent_project_id = -1)
          OR ('[P.P_SHOW_MASTER_ONLY]'='N'))
AND     template_flag = 'N'
AND
KDRV_SECURITY.Can_User_Access_Project([SYS.USER
_ID], master_project_id) = 'Y'
order by  2
```

Use Bind Variables? ☐

Ready (Read-Only, Seed Data)

Figure 4-1 Example of a Token Used in a SQL Statement

Token Builder Window Overview

In each of the entity windows listed in *"Where Tokens Are Used"* on page 38, a Token can be created by opening the Token Builder window.

To open the Token Builder window through the Request Types window:

1. Open a Request Type window, either by generating a new Request Type or by opening an existing one.
2. Click the **Commands** tab.
3. Click **New Cmd**.
4. Click **Tokens**.

   The Token Builder window opens, as shown in *Figure 4-2*.

5. Use the Token Builder window to help construct valid Tokens.

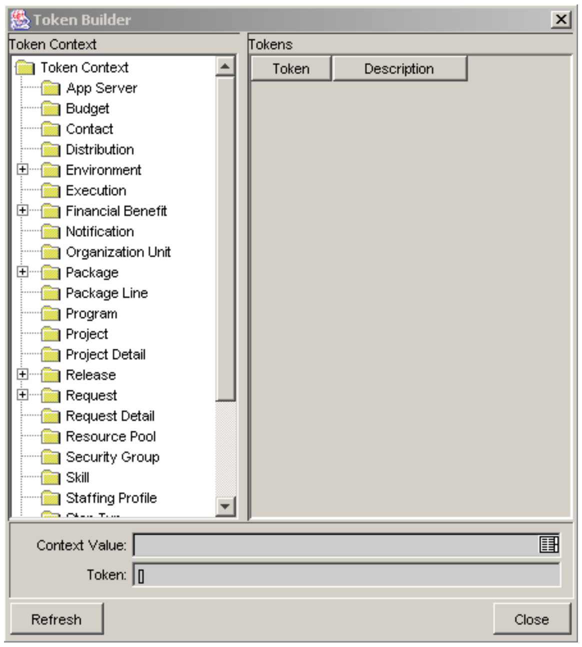


Figure 4-2 Token Builder Window

Folders are displayed in the left pane of the Token Builder window. These folders contain groups of Tokens that correspond to entities defined in Mercury ITG Center. For a list of entities and associated Tokens, see *"Tokens"* on page 95. 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.

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* - the Environment and Environment App entities evaluate differently than the other entities.

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

Prefix (Entity)	Entity and Description	User Data Format?	Parameter Format?
AS	App Server	N	N
BGT	Budget	Y	N
CON	Contact	Y	N
DEST_ENV	Destination Environment. If an App Code is specified, it will be 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're 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
NOTIF	Notification	N	N
ORG	Organization Unit	Y	N
PKG	Package	Y	N

Table 4-1. Entities

Prefix (Entity)	Entity and Description	User Data Format?	Parameter Format?
PKG.PKGL	Package (Package Line)	Y	N
PKG.PEND	Package (Pending Package)	Y	N
PKGL	Package Line	Y	Y
PRG	Program	Y	N
PRJ	Project	Y	N
PRJD	Project Details	N	Y
REL	Release	N	N
REL.DIST	Release (Distribution)	Y	N
REQ	Request	Y	Y
REQ.PEND	Request (Pending)	N	N
REQD	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're null.	Y	N
SOURCE_ENV.ENV	Source Environment. Ignores App Codes and only uses the ENV values.	Y	N
SYS	System	N	N
TSK	Task	Y	N
TSK.PEND	Task (Pending)	N	N
USR (User)	User	Y	N
VAL	Validation	N	N
VAL.VALUE	Validation (Value). Use this format to specify a specific Validation.	Y	N
VALUE	Validation (Value)	Y	N
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]
```

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 will resolve to information related to the Workflow, whereas WF.WFS Tokens will 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]
```

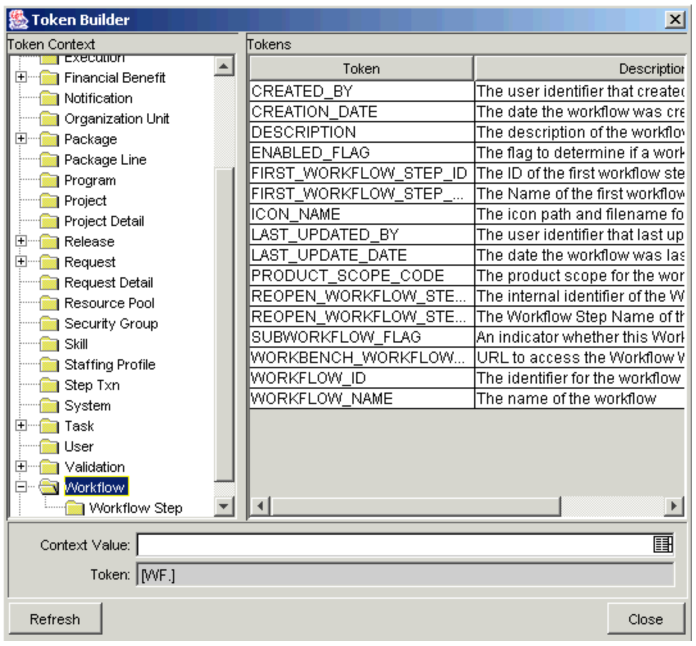
Tokens are evaluated according to the current context of Mercury ITG Center, which is derived based on information known at the time of evaluation. For more information, see *"Token Evaluation"* on page 54.

Explicit Entity Format

It is possible to provide a specific context value for an entity. This allows the default context to be overridden. Some Tokens can never be evaluated in the default context. In these cases, the context must be set using an explicit entity format:

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

The Token Builder helps generate Tokens in this format by providing a list of possible entity name values. When such a list is available, the Context Value auto-complete field at the bottom of the Token Builder becomes enabled. Like any other auto-complete field, either type into the field to reduce the list or click the auto-complete icon in the field to open the Validate window. Once a value is selected, it is inserted into the Token in the Token field, generating an explicit entity Token (see *Figure 4-3*).



Execution
Financial Benefit
Notification
Organization Unit
Package
Package Line
Program
Project
Project Detail
Release
Request
Request Detail
Resource Pool
Security Group
Skill
Staffing Profile
Step Txn
System
Task
User
Validation
Workflow
Workflow Step

Token	Descriptior
CREATED_BY	The user identifier that createc
CREATION_DATE	The date the workflow was cre
DESCRIPTION	The description of the workflo
ENABLED_FLAG	The flag to determine if a work
FIRST_WORKFLOW_STEP_ID	The ID of the first workflow ste
FIRST_WORKFLOW_STEP_...	The Name of the first workflow
ICON_NAME	The icon path and filename fo
LAST_UPDATED_BY	The user identifier that last up
LAST_UPDATE_DATE	The date the workflow was las
PRODUCT_SCOPE_CODE	The product scope for the wor
REOPEN_WORKFLOW_STE...	The internal identifier of the W
REOPEN_WORKFLOW_STE...	The Workflow Step Name of th
SUBWORKFLOW_FLAG	An indicator whether this Worl
WORKBENCH_WORKFLOW...	URL to access the Workflow V
WORKFLOW_ID	The identifier for the workflow
WORKFLOW_NAME	The name of the workflow

Token: [WF.]

Figure 4-3 Explicit Entity Format

Example

Suppose the Email Address for the user “jsmith” is to be referenced. The Token would be:

```
[USR=”jsmith”.EMAIL_ADDRESS]
```

To construct this Token in the Token Builder window:

1. Select the User folder.

   Available Tokens are displayed in the list on the right pane. The Context Value field at the bottom of the Token Builder is enabled. The string [USR.] appears in the Token field below the Context Value field.

2. Click the auto-complete icon in the Context Value field.

   A Validate window opens with a list of users.

3. Scroll through the list to find user “jsmith.” Select this user and click **OK**.

   The string [USR=”jsmith”] appears in the Token field.

4. In the list of Tokens, select EMAIL_ADDRESS.

   The string [USR=”jsmith”.EMAIL_ADDRESS] appears in the Token field. This is the complete Token. Since the Token is now complete, the Token field becomes enabled.

5. Select the Token.

6. Press Ctrl+C on the keyboard to copy the Token.

7. Press Ctrl+V on the keyboard to paste the Token into another field.

Using Tokens within Other Tokens

The explicit entity format can be used to put 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

Table 4-2 includes a list of the Tokens that support the explicit entity format.

Note

It is important to note that *entity_name* is case-sensitive and can contain spaces or other ASCII symbols.

Tokens for the User and Security Group entities can never be evaluated in the default format, and require the use of the explicit entity format. An example would be the Token [USR.EMAIL_ADDRESS]. This Token can never be evaluated because Mercury ITG Center cannot determine to which user it should refer.

Table 4-2. Tokens supporting explicit entity format

Token Prefix	Example	Acceptable Explicit Entry
BGT	[BGT=”Development Budget”.CREATED_BY]	Budget Name
CON	[CON=”Smith, John”.PHONE_NUMBER]	Last Name, First Name
ENV	[ENV=”ITG_SERVER".CLIENT_TRANSFER_PROTOCOL]	Environment Name
ORG	[ORG="Project Managers".MANAGER_ID]	Organization Unit Name
PKG	[PKG="30010".CREATED_BY]	Package Number
REQ	[REQ="30006".CREATED_BY]	Request Number
RSCP	[RSCP="Development Resources".CREATED_BY]	Resource Pool Name
SG	[SG=”Administrator”.LAST_UPDATED_BY]	Security Group Name
SKL	[SKL="Architect".AVERAGE_COST_RATE]	Skill Name
STFP	[STFP="ITG Pilot".CREATED_BY]	Staffing Profile Name
USR	[USR="jsmith".LAST_NAME]	User Name
VAL	[VAL="Date".CREATED_BY]	Validation Name
WF	[WF="Dev -> Test -> Prod".CREATED_BY]	Workflow Name
WF.WFS	[WF=“Workflow Name”.WFS=“1”.STEP_NAME]	Workflow Step Sequence Number

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 has been generated whose 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 being referenced, UD indicates that User Data is being 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-complete Lists may have different hidden and displayed values. For all other Validations, the hidden and displayed values are identical.

When context can be determined, User Data Tokens are displayed with the system-defined Tokens in the Token Builder.

Table 4-1 indicates which Tokens support the User Data format.

Parameter Format

Object Type custom fields, Request Type custom fields, Request Header Type fields, Project fields, and Workflow Parameters use the Parameter format for Tokens as shown below:

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

In this specific case, the Prefix is the name of the entity that uses a custom field. The modifier “P” indicates that Parameters for that entity are being 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) has been generated that is used on Package Lines. In the default format the Token would be:

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

In this context, PKGL is the prefix since the Package Lines entity has been referenced, “P” indicates that Parameters have been 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-complete Lists 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

This section provides some additional details and examples on using Request Tokens in the following context:

- *Request Token Prefixes*
- *Tokens in Request Table Components*

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>
```
Example: REQ.DEPARTMENT_CODE

```
REQ.P.<custom header field Token>
```
Example: REQ.P.BUSINESS_UNIT

```
REQ.P.<field group Token starting with KNTA_>
```
Example: REQ.P.KNTA_SKILL

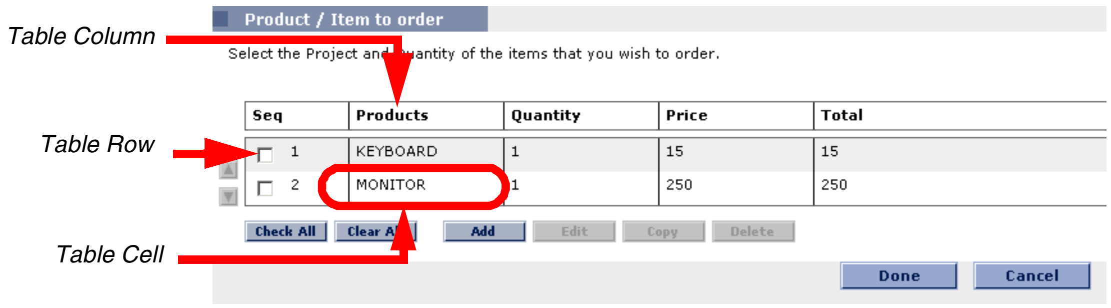
Fields defined in the Request Type use the REQD prefix. It is also possible to access standard header fields using the REQD prefix:

```
REQD.P.<custom detail field>
```

```
REQD.<standard header Token>
```

Tokens in Request Table Components

When referring to items in a Table Component, the Tokens need to follow specific formats. These formats differ depending on the item that is being referenced within the table. *Figure 4-4* illustrates the basic elements of the table. These elements will be referenced when discussing the different options for referencing data within the table using Tokens.



Product / Item to order

Select the Project and quantity of the items that you wish to order.

Seq	Products	Quantity	Price	Total
1	KEYBOARD	1	15	15
2	MONITOR	1	250	250

Figure 4-4 Table component formats

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 referencing 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 will be used. A table component named Employee with 4 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 shown.

```
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 4-3* shows other attributes of the Environments and Applications.

Table 4-3. Sample Environment and App Attributes

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

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

Table 4-4. 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

Token Evaluation

Tokens are evaluated at the point when Mercury ITG 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, Project, 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.

Appendix A System Special Commands

This appendix discusses the pre-defined Special Commands:

Special Commands

The following sections describe each Special Command in detail:

- *"ksc_copy_script Special Commands"* on page 79
- *"ksc_om_migrate"* on page 82
- *"ksc_capture_output"* on page 83
- *"ksc_gl_migrate"* on page 84
- *"ksc_parse_jcl"* on page 85
- *"ksc_submit_job"* on page 85
- *"ksc_set_exit_value"* on page 86
- *"ksc_clear_exit_value"* on page 86
- *"ksc_run_sql"* on page 86

ksc_connect Special Commands

The ksc_connect Special Commands instruct the execution engine to open a connection to a specified Environment. This command initiates a TELNET, SSH or SSH2 session with the server or client defined for the Environment. The command then sends all command steps that follow it directly to the machine, as though someone was actually typing the command on that machine. In this way, the execution engine is able to run virtually any command-line directive that the machine understands.

Note

All ksc_connect Special Commands must end with the 'ksc_exit' Special Command to exit the TELNET, SSH or SSH2 session.

The ksc_connect Special Commands include:

- *ksc_connect_dest_client*
- *ksc_connect_dest_server*
- *ksc_connect_source_client*
- *ksc_connect_source_server*

ksc_connect_dest_client

This command initiates a TELNET, SSH or SSH2 session with the client of the destination Environment. The destination Environment refers to the destination Environment of the Workflow Step initiating command execution.

Table A-1. ksc_connect_dest_client Parameters

Parameter	Default Token	Description
USERNAME	[DEST_ENV.CLIENT_ USERNAME	Username on [DEST_ENV].
PASSWORD	[DEST_ENV.CLIENT_ PASSWORD	Password on [DEST_ENV].
NT_DOMAIN	[DEST_ENV.CLIENT_NT_ DOMAIN	Windows NT Domain name of [DEST_ENV].
DEST_BASE_ PATH	[DEST_ENV.CLIENT_ BASE_PATH	Base Path of [DEST_ENV].
CONNECTION_ PROTOCOL	[DEST_ENV.CLIENT_CON _PROTOCOL_MEANING]	Specifies the connection protocol. Possible values are listed in Validation "CONNECTION_PROTOCOL".
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example Using ksc_connect_dest_client

```
# Make a remote connection to the client of the
# destination environment defined for the current
# workflow step.

ksc_connect_dest_client
<commands>
ksc_exit

# Make a remote connection to the client defined for
# the environment named 'STAGING'.

ksc_connect_dest_client DEST_ENV="STAGING"
<commands>
ksc_exit
```

ksc_connect_dest_server

This command initiates a TELNET, SSH or SSH2 session with the server of the destination Environment. The destination Environment refers to the destination Environment of the Workflow Step initiating command execution.

Table A-2. ksc_connect_dest_server Parameters

Parameter	Default Token	Description
USERNAME	[DEST_ENV.SERVER_ USERNAME	Username on [DEST_ENV].
PASSWORD	[DEST_ENV.SERVER_ PASSWORD	Password on [DEST_ENV].
NT_DOMAIN	[DEST_ENV.SERVER_ NT_DOMAIN	Windows NT Domain name of [DEST_ENV].
DEST_BASE_ PATH	[DEST_ENV.SERVER_ BASE_PATH	Base Path of [DEST_ENV].
CONNECTION_ PROTOCOL	[DEST_ENV.SERVER_ CON_PROTOCOL_ MEANING]	Specifies the connection protocol. Possible values are listed in Validation “CONNECTION_PROTOCOL”.
DEST_ENV	[DEST_ENV. ENVIRONMENT_NAME]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example using ksc_connect_dest_server

```
# Make a remote connection to the server of the
# destination environment defined for the current
# workflow step.

ksc_connect_dest_server
<commands>
ksc_exit

# Make a remote connection to the server defined for
# the environment named ‘Staging’.

ksc_connect_dest_server DEST_ENV="STAGING"
<commands>
ksc_exit
```

ksc_connect_source_client

This command initiates a TELNET, SSH or SSH2 session with the client of the source Environment. The source Environment refers to the source Environment of the Workflow Step initiating command execution.

Table A-3. ksc_connect_source_client Parameters

Parameter	Default Token	Description
USERNAME	[SOURCE_ENV.CLIENT_ USERNAME	Username on [SOURCE_ENV].
PASSWORD	[SOURCE_ENV.CLIENT_ PASSWORD	Password on [SOURCE_ENV].
NT_DOMAIN	[SOURCE_ENV.CLIENT_ NT_DOMAIN	Windows NT Domain name of [SOURCE_ENV].
SOURCE_BASE _PATH	[SOURCE_ENV.CLIENT_ BASE_PATH	Base Path of [SOURCE_ENV].
CONNECTION_ PROTOCOL	[SOURCE_ENV.CLIENT_ CON_PROTOCOL_ MEANING]	Specifies the connection protocol. Possible values are listed in Validation “CONNECTION_PROTOCOL”.
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.

Example using ksc_connect_source_client

```
# Make a remote connection to the client of the source
# environment defined for the current workflow step.

ksc_connect_source_client
<commands>
ksc_exit

# Make a remote connection to the client defined for
# the environment named ‘STAGING’.

ksc_connect_source_client SOURCE_ENV="STAGING"
<commands>
ksc_exit
```

ksc_connect_source_server

This command initiates a TELNET, SSH or SSH2 session with the server of the source Environment. The source Environment refers to the source Environment of the Workflow Step initiating command execution.

Table A-4. ksc_connect_source_server Parameters

Parameter	Default Token	Description
USERNAME	[SOURCE_ENV.SERVER_ USERNAME	Username on [SOURCE_ENV].
PASSWORD	[SOURCE_ENV.SERVER_ PASSWORD	Password on [SOURCE_ENV].
NT_DOMAIN	[SOURCE_ENV.SERVER_ NT_DOMAIN	Windows NT Domain name of [SOURCE_ENV].
SOURCE_BASE _PATH	[SOURCE_ENV.SERVER_ BASE_PATH	Base Path of [SOURCE_ENV].
CONNECTION_ PROTOCOL	[SOURCE_ENV.SERVER_ CON_PROTOCOL_ MEANING]	Specifies the connection protocol. Possible values are listed in Validation “CONNECTION_PROTOCOL”.
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.

Examples using ksc_connect_source_server

```
# Make a remote connection to the server of the source
# environment defined for the current workflow step.

ksc_connect_source_server
<commands>
ksc_exit

# Make a remote connection to the server defined for
# the environment named 'STAGING'.

ksc_connect_source_server SOURCE_ENV="STAGING"
<commands>
ksc_exit
```

ksc_exit

This command exits the TELNET, SSH or SSH2 session initiated by the *ksc_connect Special Commands*. For examples using ksc_exit, see *“ksc_connect Special Commands”* on page 58.

ksc_copy Special Commands

The ksc_copy Special Commands provide the mechanism for transferring files to and from the various Environments defined in Mercury ITG Center.

Note

The default use of these commands requires that the entity containing the command has three fields with the following Tokens defined:

1 `[P.P_FILENAME]`

2 `[P.P_FILE_TYPE]`

3 `[P.P_SUB_PATH]`

If these fields are not defined as part of the entity, they must be passed as parameters or the command will fail.

Files are copied using either FTP, SCP or SCP2, depending on the configuration of the Environment.

The ksc_copy Special Commands include:

- *ksc_copy_client_client*
- *ksc_copy_client_server*
- *ksc_copy_server_client*
- *ksc_copy_server_server*
- *ksc_copy_client_tmp*
- *ksc_copy_server_tmp*
- *ksc_copy_tmp_client*
- *ksc_copy_tmp_server*

ksc_copy_client_client

This command copies a file from the source client Environment to the destination client Environment.

Table A-5. ksc_copy_client_client Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
SOURCE_BASE _PATH	[SOURCE_ENV. CLIENT_BASE_PATH]	The base path of the source client Environment to be used instead of what is defined for the current source Environment.
DEST_BASE _PATH	[DEST_ENV. CLIENT_BASE_PATH]	The base path of the destination client Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example #1 using ksc_copy_client_client

```
# Copy a file between source and destination clients.

ksc_copy_client_client SUB_PATH="forms"
  FILENAME="[P.P_MODULE].fmb" FILE_TYPE="BINARY"

# Copy a file between the client defined in the ‘STAGING’
# environment and the destination client.

ksc_copy_client_client DEST_ENV=”STAGING”
```

Example #2 using ksc_copy_client_client

```
# Override the base path of the destination directory.

ksc_copy_client_client DEST_BASE_PATH=”/u1/datatree/ex1”
SUB_PATH=”.” FILENAME="[P.P_MODULE].fmb" FILE_TYPE="BINARY"
```

ksc_copy_client_server

This command copies a file from the source client Environment to the destination server Environment.

Table A-6. ksc_copy_client_server Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
SOURCE_BASE _PATH	[SOURCE_ENV.CLIENT_ BASE_PATH]	The base path of the source client Environment to be used instead of what is defined for the current source Environment.
DEST_BASE _PATH	[DEST_ENV. SERVER_BASE_PATH]	The base path of the destination server Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example using ksc_copy_client_server

```
# Copy a file between source client and
# destination server.

ksc_copy_client_server SUB_PATH="install/sql"
  FILENAME="[P.P_SQL_SCRIPT]" FILE_TYPE="ASCII"
```

ksc_copy_server_client

This command copies a file from the source server Environment to the destination client Environment.

Table A-7. ksc_copy_server_client Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
SOURCE_BASE _PATH	[SOURCE_ENV. SERVER_BASE_PATH]	The base path of the source server Environment to be used instead of what is defined for the current source Environment.
DEST_BASE _PATH	[DEST_ENV. CLIENT_BASE_PATH]	The base path of the destination client Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example using ksc_copy_server_client

```
# Copy a file between source server and
# destination client.

ksc_copy_server_client SUB_PATH="[P.P_SUB_DIRECTORY]"
 FILE_TYPE="[P.P_FILE_TYPE]"
```

ksc_copy_server_server

This command copies a file from the source server Environment to the destination server Environment.

Table A-8. ksc_copy_server_server Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
SOURCE_BASE _PATH	[SOURCE_ENV. SERVER_BASE_PATH]	The base path of the source server Environment to be used instead of what is defined for the current source Environment.
DEST_BASE _PATH	[DEST_ENV. SERVER_BASE_PATH]	The base path of the destination server Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

Example using ksc_copy_server_server

```
# Copy a file between source and destination servers.
ksc_copy_server_server FILENAME="[P.P_FILE]"

# Copy a file between the source server and the
# destination server overriding the base bath.

ksc_copy_server_server FILENAME=”install_driver.sh”
DEST_BASE_PATH=”/u2/app/drivers”

# Copy a form between the ‘STAGING’ and destination servers.

ksc_copy_server_server SOURCE_ENV="STAGING" SUB_PATH="forms"
  FILENAME="[P.P_MODULE].fmb" FILE_TYPE="BINARY"
```

ksc_copy_client_tmp

This command copies a file from the source client Environment to the temporary Package transfer directory on the application server. This temporary directory is automatically cleaned up after an execution completes and can be referenced using the [AS.PKG_TRANSFER_PATH] Token.

Table A-9. ksc_copy_server_tmp Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
SOURCE_BASE _PATH	[SOURCE_ENV. CLIENT_BASE_PATH]	The base path of the source client Environment to be used instead of what is defined for the current source Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.

ksc_copy_server_tmp

This command copies a file from the source server Environment to the temporary Package transfer directory on the application server. This temporary directory is automatically cleaned up after an execution completes and can be referenced using the [AS.PKG_TRANSFER_PATH] Token.

Table A-10. ksc_copy_server_tmp Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.

Table A-10. ksc_copy_server_tmp Parameters

Parameter	Default Token	Description
SOURCE_BASE _PATH	[SOURCE_ENV. SERVER_BASE_PATH]	The base path of the source server Environment to be used instead of what is defined for the current source Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.

ksc_copy_tmp_client

This command copies a file from the temporary Package transfer directory on the application server to the destination client Environment. This temporary directory is automatically cleaned up after an execution completes and can be referenced using the [AS.PKG_TRANSFER_PATH] Token.

Table A-11. ksc_copy_server_tmp Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
DEST_BASE _PATH	[DEST_ENV. CLIENT_BASE_PATH]	The base path of the destination server Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_copy_tmp_server

This command copies a file from the temporary Package transfer directory on the application server to the destination server Environment. This temporary directory is automatically cleaned up after an execution completes and can be referenced using the [AS.PKG_TRANSFER_PATH] Token.

Table A-12. ksc_copy_server_tmp Parameters

Parameter	Default Token	Description
SUB_PATH	[P.P_SUB_PATH]	The sub-directory that should be used to locate the file relative to the base path of each Environment.
DEST_BASE _PATH	[DEST_ENV. SERVER_BASE_PATH]	The base path of the destination server Environment to be used instead of what is defined for the current destination Environment.
FILENAME	[P.P_FILENAME]	Name of the file to be copied.
FILE_TYPE	[P.P_FILE_TYPE]	The file type associated with the file (ASCII or BINARY).
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_respond

This command is currently only used to support Patch*Applicator. This command is able to intelligently respond to interactive prompts generated by the Oracle “adpatch” and “adadmin” programs. General use of this Special Command for arbitrary programs is not yet supported. For simple interactive programs, see *“ksc_simple_respond”* on page 70.

ksc_simple_respond

This command executes an interactive UNIX command on a remote computer. This command is useful when the command to be executed will prompt for additional information (such as the UNIX ‘su’ command to switch user accounts) or may not return an exit code upon completion (such as starting up a new shell using ‘sh’).

Note

This command can only be used from within a remote execution session, such as between ‘ksc_connect’ and ‘ksc_exit’ commands.

The following syntax is supported:

```
ksc_simple_respond “command”
ksc_simple_respond “command” “prompt 1" “response 1" [“prompt
2" “response 2" … ]
ksc_simple_respond “command” -hide “prompt 1" “response 1"
[“prompt 2" “response 2" … ]
```

There can be as many prompt-response pairs as necessary. Each prompt must be matched with a response, even if the response is an empty string. The prompts must appear in the exact order they will be displayed as the command is run. All arguments must be enclosed in quotes. In addition, if the command or any of the arguments contains double quotes (“), any other character can be used as the quote character. The first character after the string ‘ksc_simple_respond’ will be interpreted as the quote character, and that character must appear at the beginning and end of each argument.

By using the -hide option, the value passed in for the response will not be displayed in the execution log. In the log, the value will be displayed as ****. This flag should be used for each prompt/response pair that needs this treatment.

Note

The execution engine will wait for each specified prompt. If a prompt does not appear for some reason, then the execution engine will continue to wait for it until the command times out.

Examples using ksc_simple_respond

If it becomes necessary to invoke a new shell while in a remote session, it would be ideal to simply use the command ‘sh’. However, this can cause the execution engine to wait indefinitely while waiting for an exit code. To avoid this problem, the ‘sh’ command can be encapsulated in a ksc_simple_respond command with no prompts as shown:

```
ksc_simple_respond “sh”
```

As another example, suppose it becomes necessary to switch to another user account while in a remote session using the ‘su’ command. This command always prompts for password, unless performed by a root user. By utilizing the -hide feature, the password will not be displayed in the execution logs. This interactivity can be handled using ksc_simple_respond as follows:

```
ksc_simple_respond "su <username>" -hide "word:" "<password>"
```

Note that “word:” was used as the prompt instead of the entire word “password:”. The execution engine will wait for the specified prompt string, whether it is all—or just a part—of the prompt text.

As one more example, consider the following Bourne shell command:

```
echo "Enter a string:\c"; read str; echo $str
```

Normally, this command line would cause the execution engine to hang while waiting for an exit code (the command will never exit because it is waiting for input), which would eventually timeout when the execution timeout time is reached. Use ksc_simple_respond to process this command as shown (this command should be entered on a single line):

```
ksc_simple_respond #echo "Enter a string:\c"; read str; echo
$str# #a string:# #my_value#
```

Since the command line contained double quotes, the pound sign (#) is used as the quote character. During execution, this command step will prompt “Enter a string:” and wait for input. The string “my_value” would be entered automatically, this value will then be echoed to the output device (in this case, the execution log), and execution will continue as normal with the next command step.

ksc_local_exec

This command invokes a local process on the machine running the Mercury ITG Server. It can be used to run any program that does not require interactive input. Each call using ‘ksc_local_exec’ is an independent process. It does not execute in the context of other commands that precede it. The starting directory for the processes generated using ‘ksc_local_exec’ is the home directory of the Mercury ITG Server. Full paths to the executable being called are necessary if the Mercury ITG Server does not have the correct system path information.

Note

The ksc_local_exec command does not open a TELNET, SSH or SSH2 connection to the Mercury ITG Server. It operates by creating a new child process on the machine that is running the Mercury ITG Server. Therefore, the user account and password for this process will be the same as the account and password used to start the Mercury ITG Server.

Example using ksc_local_exec

```
# Rename existing file ‘file.txt’ to ‘newfile.txt’
ksc_local_exec mv file.txt newfile.txt


# Run a DOS batch file
ksc_local_exec cmd /c runme.bat
```

System commands do not invoke either Unix shells or DOS shells. This means that the following code segment using ‘ksc_local_exec’ is not valid, because it cannot use the ‘pipe’ (|) or redirect commands (>):

```
ksc_local_exec cat names.txt | grep address > file.out
```

An effective way to use the ksc_local_exec command is to put a series of commands into a .sh file, and then execute the .sh file as shown:

```
ksc_begin_script + [AS.CR_TRANSFER_PATH] run.sh
..
<series of commands>
ksc_end_script

ksc_local_exe ksh run.sh
```

ksc_replace

This command is used to edit the contents of a file and place it into another file. This command works in a way similar to the ‘sed’ utility and supports the same substituting expressions.

The files must be located on the Mercury ITG Server in the [AS.PKG_TRANSFER_PATH] directory. This requires the use of the ksc_copy_tmp_* commands.

Table A-13. ksc_replace Parameters

Parameter	Default Token	Description
FILENAME	[P.P_FILENAME]	Name of the source file to be edited.
OUTFILE	[OUTFILE]	Name of the output file after applying the substitution expressions.
SUBST		The substitution expression.

Example using ksc_replace

```
ksc_copy_server_tmp FILENAME=”config.template”
FILE_TYPE=”ASCII”

ksc_replace FILENAME=”config.template” OUTFILE=”config.cfg”
SUBST=”s/NAME/[P.NAME]/g”

ksc_copy_tmp_server FILENAME=”config.cfg”
```

ksc_set

This command sets the value of a temporary variable which may be used to manage command conditions or aid in command processing.

The following syntax is supported:

```
ksc_set VARIABLE=”Value”
```

To reference the value of this variable, use the familiar Token syntax without any prefix. Unlike the ‘ksc_store’ command, ‘ksc_set’ does not write values to the database. The scope of the variable that is set is valid from when the variable is defined to the end of the command steps for the entity. This make using ‘ksc_set’ more attractive than shell variables because the values are retained between separate ‘ksc_connect’ sessions. Another advantage of using ‘ksc_set’ is that the Token values are visible in the logs, not just the variable names. This command may be nested within a ‘ksc_connect’ command (see the following example).

Example using ksc_set

```
# Set the value of a compile flag.
#
ksc_set COMPILE=”YES”
# ksc_set nested within a ksc_connect
```

```
ksc_connect_dest_server
ksc_set REBUILD="NO"
ksc_exit
```

Later, a temporary variable can be referenced in a command condition or in another command step. For example, the command condition may look like:

```
'[COMPILE]' = 'YES'
```

ksc_set_env

Use this command to set the correct Environment context of an execution in cases where the Workflow source and destination Environments are overridden using the DEST_ENV and SOURCE_ENV parameters. Normally it is not necessary to use this command since it is called internally from other Special Commands. If it is used on a stand-alone basis, it must come after any 'ksc_copy' commands.

Table A-14. ksc_set_env Parameters

Parameter	Default Token	Description
DEST_ENV_ID	[DEST_ENV. ENVIRONMENT_ID]	ID of the destination Environment to be used instead of the destination Environment on the current Workflow Step.
SOURCE_ENV_ID	[SOURCE_ENV. ENVIRONMENT_ID]	ID of the source Environment to be used instead of the source Environment on the current Workflow Step.
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_store

This command dynamically sets the values of fields defined for Object Types, Request Types, and Report Types. This command is useful to set or alter the value of fields based on the command output. This command may only be used on fields which have been custom configured. Custom configured fields are those with Tokens that are evaluated using the [P.<TOKEN>] or [VP.<TOKEN>] format. After altering a Token, future evaluations of the Token will use the new value. The new values are written to the database, so the changes are not temporary as in ‘ksc_set’.

This command may be nested within a ‘ksc_connect’ command (as seen in the following example) and its value can be referenced in command conditions.

The following syntax is supported:

```
ksc_store TOKEN=”Value”
ksc_store TOKEN=”Hidden Value”, “Visible Value”
```

In the first case, the hidden and visible values of the field will be set to the same value. In the second case, the hidden and visible values are set independently. “Hidden Value” refers to the [P.<TOKEN>] format. “Visible Value” refers to the [VP.<TOKEN>] format.

Example using ksc_store

In the following example, it is assumed that the entity in question has the following Tokens defined:

```
[P.DRIVER]
[P.REVISION]
[P.RESULT]

# Store the name of the driver file.

ksc_store DRIVER=”driver.sh”

# Capture the Revision number of a file.
#
ksc_connect_dest_server
cd “SourceCode/java”
grep ‘$Revision’ ServerAdmin.java
ksc_store REVISION=”[EXEC.OUTPUT]”
ksc_exit

# Set the hidden and visible result codes of a parameter.
#
ksc_store RESULT=”IN_PROG”,”In Progress”
```

ksc_comment

This command adds single line comments to the execution log. It can be used to indicate informational or error messages. HTML tags are supported.

The following syntax is supported:

```
ksc_comment <comment>
```

The comment text can be any text string.

ksc_concsub

This command submits Oracle Application concurrent requests from the operating system command line. It is treated as a Special Command because the command engine must capture the concurrent request ID, which is an output of successful submission. To work properly, this command must be called within a ‘ksc_connect - ksc_exit’ command block.

If ‘ksc_concsub’ is used to submit a concurrent request to an Oracle Applications database other than where Mercury ITG Center is currently installed, the ORA_APPS_DB_LINK parameter must be added to the ‘ksc_concsub’ command. Otherwise, the status of the concurrent request cannot be determined after submission.

The following syntax is supported:

```
ksc_concsub ORA_APPS_DB_LINK="DB_LINK" CONCSUB
```

DB_LINK corresponds to the database link from the Mercury ITG Center schema to the APPS schema of the database to which the concurrent request is submitted.

Example using ksc_concsub

```
ksc_concsub ORA_APPS_DB_LINK=[DEST_ENV.ORA_APPS_DB_LINK]
CONCSUB
[DEST_ENV.APP.DB_USERNAME]/[DEST_ENV.APP.DB_PASSWORD]@[DEST_ENV
.DB_CONNECT_STRING]  FND  'Application Developer' SYSADMIN
WAIT=N CONCURRENT FND FNDFMREG [DEST_ENV.APP_CODE]
[P.P_FILENAME]
```

Note

The Special Command ‘ksc_concsub’ is followed by the exact CONCSUB call that will be executed directly at the command line.

The complete syntax for Oracle's CONCSUB is shown below. Optional parameters are in square brackets.

```
CONCSUB
<ORACLE ID>
<Responsibility Application Short Name>
<Responsibility Name>
<User Name>
[WAIT=N]
CONCURRENT
<Concurrent Program Application Short Name>
<Concurrent Program Name>
[START=<Requested Start Date>]
[REPEAT_DAYS=<Repeat Interval>]
[REPEAT_END=<Request Resubmission End Date>]
<Concurrent Program Arguments...>
```

For additional information on using the CONCSUB command, see the Oracle documentation.

Note

It is not possible to retrieve the concurrent request logs from a 'ksc_concsub' submission submitted against a remote database.

ksc_begin_script / ksc_end_script

The object command structure of Mercury ITG Center lends itself nicely to standard, step-by-step processes. In most cases, these commands are fully capable of automating the migration of an object. However, in some circumstances, it is necessary to add additional logic to the commands for an object. For example, perhaps a loop must be generated to repeat a command several times. This is where scripts-on-the-fly are best applied.

Scripts-on-the-fly are designed to leverage the architecture, tools, and knowledge already present in an organization. By using a script-on-the-fly, administrators can define migration logic in their preferred scripting language (such as Bourne Shell, C Shell or Perl). The scripts only need to be defined once. The execution engine copies the script wherever it needs to be executed. The execution engine can also be instructed to clean up the script after it has been executed, leaving no traces behind.

The following syntax is supported:

```
ksc_begin_script <full_path_to_file_to_be_generated>
<directives from any scripting language>
ksc_end_script
```

It is commonly used in the following format:

```
ksc_begin_script [AS.PKG_TRANSFER_PATH][P.P_SCRIPT_FILENAME]
```

Since the script will be generated into a temporary directory by use of the [AS.PKG_TRANSFER_PATH] Token, this Token will reference a unique temporary directory per execution and end with the proper directory slash '/' or '\'. After generation, the script can be transferred to another machine for execution using the 'ksc_copy_script' commands described in *ksc_copy_script Special Commands*.

Example using ksc_begin_script and ksc_end_script

```
ksc_begin_script [AS.PKG_TRANSFER_PATH][P.P_SCRIPT_FILENAME]
#!/usr/bin/csh
#
# Script to lock, check in, and re-checkout the original
# file using RCS commands.
#
# Print a warning if the file does not exist.
#

if ($#argv != 2) then
        echo "$0 : wrong number of arguments"
        echo "Usage: $0 sub_path filename"
        exit 1
endif

set sub_path = $argv[1]
set filename = $argv[2]

if (-e "$sub_path/RCS/$filename,v") then
        rcs -l $sub_path/$filename
        ci -m"Before Copy." $sub_path/$filename
        co -l $sub_path/$filename
else
       echo "Warning: File $sub_path/$filename not found in
RCS repository"
endif

exit 0
ksc_end_script

# Copy the script to the destination server and excute it.
ksc_copy_script_dest_server
ksc_connect_dest_server
csh [P.P_SCRIPT_FILENAME]
rm [P.P_SCRIPT_FILENAME]
ksc_exit
```

ksc_copy_script Special Commands

Use these Special Commands to transfer files from the temporary file transfer directory (defined by Token [AS.PKG_TRANSFER_PATH]) to other machines. These commands are typically used in conjuction with the 'ksc_begin_script' and 'ksc_end_script' commands, but can also be used in other ways.

The ksc_copy_script Special Commands include:

- *ksc_copy_script_dest_client*
- *ksc_copy_script_dest_server*
- *ksc_copy_script_source_client*
- *ksc_copy_script_source_server*

ksc_copy_script_dest_client

This command copies a script contained in [AS.PKG_TRANSFER_PATH], a temporary directory located on the Mercury ITG Server, to the base path of the destination client Environment.

Table A-15. ksc_copy_script_dest_client Parameters

Parameters	Default Token	Description
SCRIPT _FILENAME	[P.P_SCRIPT_FILENAME]	The name of the script file to transfer.
DEST_BASE _PATH	[DEST_ENV. CLIENT_BASE_PATH]	The base path of the destination client Environment to be used instead of what is defined for the current destination Environment.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_copy_script_dest_server

This command copies a script contained in [AS.PKG_TRANSFER_PATH], a temporary directory located on the Mercury ITG Server, to the base path of the destination server Environment.

Table A-16. ksc_copy_script_dest_server Parameters

Parameters	Default Token	Description
SCRIPT _FILENAME	[P.P_SCRIPT_FILENAME]	The name of the script file to transfer.

Table A-16. ksc_copy_script_dest_server Parameters

Parameters	Default Token	Description
DEST_BASE _PATH	[DEST_ENV. SERVER_BASE_PATH]	The base path of the destination server Environment to be used instead of what is defined for the current destination Environment.
DEST_ENV	[DEST_ENV]	Name of the destination Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_copy_script_source_client

This command copies a script contained in [AS.PKG_TRANSFER_PATH], a temporary directory located on the Mercury ITG Server, to the base path of the source client Environment.

Table A-17. ksc_copy_script_source_client Parameters

Parameters	Default Token	Description
SCRIPT _FILENAME	[P.P_SCRIPT_FILENAME]	The name of the script file to transfer.
DEST_BASE _PATH	[SOURCE_ENV. CLIENT_BASE_PATH]	The base path of the source client Environment to be used instead of what is defined for the current source Environment.
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the destination Environment on the current Workflow Step.

ksc_copy_script_source_server

This command copies a script contained in [AS.PKG_TRANSFER_PATH], a temporary directory located on the Mercury ITG Server, to the base path of the source server Environment.

Table A-18. ksc_copy_script_source_client Parameters

Parameters	Default Token	Description
SCRIPT _FILENAME	[P.P_SCRIPT_FILENAME]	The name of the script file to transfer.
SOURCE_BASE _PATH	[SOURCE_ENV. SERVER_BASE_PATH]	The base path of the source server Environment to be used instead of what is defined for the current source Environment.
SOURCE_ENV	[SOURCE_ENV]	Name of the source Environment to be used instead of the source Environment on the current Workflow Step.

ksc_om_migrate

Use this command to launch migrations supported by the Object*Migrator.

The following syntax is supported:

```
ksc_om_migrate CONC_PROGRAM=<conc_program_name>
APP_SHORT_NAME=<APP_SHORT_NAME> OM_ARCHIVE_FLAG=<Y/N>
```

The parameters CONC_PROGRAM and APP_SHORT_NAME are required. All other parameters are optional and are used to override the default behavior.

Table A-19. ksc_om_migrate Parameters

Parameter	Default Token	Description
CONC _PROGRAM	None. This is a mandatory parameter.	The concurrent program name. This has been pre-configured and will not need to be modified.
OM_ARCHIVE _FLAG	[WFS. OM_ARCHIVE_FLAG]	Specifies whether the migration will store to the archive rather than using what has been specified for the current Workflow Step.
APP_SHORT _NAME	None. This is a required parameter.	This value is normally “CLM” but can be modified if the Object*Migrator has been installed into a custom account.

Table A-19. ksc_om_migrate Parameters

Parameter	Default Token	Description
SOURCE _ENV	[SOURCE_ENV]	The Environment to migrate from rather than the one defined on the Workflow Step.
DEST _ENV	[DEST_ENV]	The Environment to migrate to rather than the one defined on the Workflow Step.

Example using ksc_om_migrate

```
#
#Launch an AOL Concurrent Program Migration
#
ksc_om_migrate CONC_PROGRAM="CLMRMCP1" APP_SHORT_NAME="CLM"
```

ksc_capture_output

The ‘ksc_capture_output’ Special Command is only used in Validations. It is used to get data from an alternate source, and use that data to populate an auto-complete field. This functionality provides additional flexibility when designing auto-complete lists.

Many enterprises have found that they need to use alternate sources of data within their applications. Examples of these sources might be a flat file, an alternate database source, or output from a command line execution. The ‘ksc_capture_output’ command may be used in conjunction with these alternate data sources, in the context of a Validation, to provide a list of values on the fly.

The syntax for the ‘ksc_capture_output’ is:

```
ksc_capture_output <command>
```

In the Validation Workbench, under Validated By, choose either **Command With Delimited Output** or **Command With Fixed Width Output** and input the delimiting character or field length information. Then, under New Command, enter the steps. The example below would put the Validations into the address.txt file, then run the ‘ksc_capture_output’ against the address.txt file:

```
ksc_begin_script[AS.PKG_TRANSFER_PATH]address.txt
street
city
state
```

```
zipcode
ksc_end_script
ksc_capture_output cat[AS.PKG_TRANSFER_PATH]address.txt
```

In this case, the entire sequence of commands would be executed on the local machine where the Mercury ITG Server is running. This is the preferred method of invoking 'ksc_capture_output'. The 'ksc_capture_output' command may be embedded between 'ksc_connect' and 'ksc_exit' commands, but the time delay is significant depending on network load (because the Validation actually requires an entire TELNET, SSH or SSH2 session to be generated to the remote machine). It is recommended that 'ksc_capture_output' only be used in a local execution scenario.

'ksc_capture_output' may be called more than once. Each call will append the results to the previous call.

ksc_gl_migrate

Use this command to launch migrations supported by the GL *Migrator.

The following syntax is supported:

```
ksc_gl_migrate CONC_PROGRAM=<conc_program_name>
APP_SHORT_NAME=<APP_SHORT_NAME> GL_ARCHIVE_FLAG=<Y/N>
```

The parameters CONC_PROGRAM and APP_SHORT_NAME are required. All other parameters are optional and are used to override the default behavior.

Table A-20. ksc_gl_migrate Parameters

Parameter	Default Token	Description
CONC _PROGRAM	None. This is a mandatory parameter.	The concurrent program name. This has been pre-configured and will not need to be modified.
GL_ARCHIVE _FLAG	[WFS. OM_ARCHIVE_FLAG]	Specify whether the migration will store to the archive rather than using what has been specified for the current Workflow Step.
APP_SHORT _NAME	None. This is a required parameter.	This value is normally "CLGM" but can be modified if the GL*Migrator has been installed into a custom account.
SOURCE _ENV	[SOURCE_ENV]	The Environment to migrate from rather than the one defined on the Workflow Step.

Table A-20. ksc_gl_migrate Parameters

Parameter	Default Token	Description
DEST _ENV	[DEST_ENV]	The Environment to migrate to rather than the one defined on the Workflow Step.

Example ksc_gl_migrate

```
#
# Launch a Budget Organization migration
#
ksc_gl_migrate CONC_PROGRAM="CLGMRBO1" APP_SHORT_NAME="CLGM"
```

ksc_parse_jcl

This command is only used by the ‘OS/390 JCL Migration’ Object Type to parse a JCL script using the Mainframe parameters for the specified Environment.

Table A-21. ksc_parse_jcl Parameters

Parameter	Default Token	Description
FILENAME	[P.P_FILENAME]	Name of the JCL source file to be edited.
OUTFILE	[OUTFILE]	Name of the output JCL file after applying the substitution expressions.
ENV_ID	[DEST_ENV. ENVIRONMENT_ID]	The ID of the Environment containing the mainframe substitution expressions.

ksc_submit_job

This command is only used by the ‘OS/390 JCL Migration’ Object Type to submit JCL to the Mainframe JES.

Table A-22. ksc_submit_job Parameters

Parameter	Default Token	Description
PATH	[AS.PKG_TRANSFER_ PATH]	Path to the JCL file.

Table A-22. ksc_submit_job Parameters

Parameter	Default Token	Description
FILENAME	[P.P_FILENAME]	Name of the JCL source file to be edited.

ksc_set_exit_value

Use this command to set the exit value of the command execution to any value. When not used, the command execution engine returns standard execution results, such as FAILURE, SUCCESS, and ERROR (if an internal error occurred) that the Workflow engine can transition on. Using 'ksc_set_exit_value' allows for the flexibility to set any exit value and enables custom Workflow transitions.

The following formats are supported:

```
# Sets the hidden and visible value to <value>.
ksc_set_exit_value "<value>"

# Sets both the hidden and visible values independently.
ksc_set_exit_value "<hidden_value>", "<visible_value">
```

The Workflow engine will key off of the hidden value to determine if a transition should be made. The visible value is for display purposes.

'ksc_set_exit_value' is ideal for situations where there could be a number of different execution results, not just Success or Failure. Using 'ksc_set_exit_value' allows the Workflow engine to transition on any number of execution outcomes.

ksc_clear_exit_value

Use this command to clear the exit value set by 'ksc_set_exit_value'. When cleared, the execution engine will return its standard results, SUCCESS, FAILURE, or ERROR.

ksc_run_sql

This command runs a SQL query against the chosen Environment. The result of the last row queried is returned in the [SQL_OUTPUT] Token. The result of the entire query is placed in the [AS.PKG_TRANSFER_PATH][PKGL.SEQ].txt file in *ITG_Home*.

To run this Special Command, any Execution Steps in the Change Management Workflow must have their Source Environments defined in the Workflow Step window.

Table A-23 lists this Special Command's parameters.

Table A-23. ksc_run_sql Parameters

Parameter	Default Token	Description
QUERY_STRING	[QUERY_STRING]	A SQL select statement.
ENV_NAME	[ENV_NAME]	The name of the Environment where data will be queried. The JDBC connection should be checked in the Environment checker.
EXCEPTION_OPTION	[EXCEPTION_OPTION]	If no data is returned, determines if an exception should be thrown. The only available option is '-no_data_exception.'

Example ksc_run_sql

```
ksc_run_sql QUERY_STRING="select sysdate from sys.dual"
ENV_NAME="[SOURCE_ENV.ENVIRONMENT_NAME]"
```

Note

The 'ksc_run_sql' Special Command can be used to populate a Validation. This is appropriate when the Validation is validated by a Command with Delimited Output. In this case, the Data Delimiter should be set to "#@#".

The following code is an example of ksc_run_sql in a Validation.

```
ksc_run_sql QUERY_STRING="select id, name from some_table"
ENV_NAME="[SOURCE_ENV.ENVIRONMENT_NAME]"
ksc_capture_output cat [AS.PKG_TRANSFER_PATH][PKGL.SEQ].txt
```

Summary of All Special Command Parameters

Table A-24 provides the parameters for all predefined Special Commands.

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_begin_script		
ksc_comment		
ksc_concsub		
ksc_connect_dest_client	USERNAME	[DEST_ENV.CLIENT_USERNAME]
	PASSWORD	[DEST_ENV.CLIENT_PASSWORD]
	NT_DOMAIN	[DEST_ENV.CLIENT_NT_DOMAIN]
	DEST_BASE_PATH	[DEST_ENV.CLIENT_BASE_PATH]
	CONNECTION_ PROTOCOL	[DEST_ENV.CLIENT_CON_PROTOCOL_ MEANING]
	DEST_ENV	[DEST_ENV]
ksc_connect_dest_server	USERNAME	[DEST_ENV.SERVER_USERNAME]
	PASSWORD	[DEST_ENV.SERVER_PASSWORD]
	NT_DOMAIN	[DEST_ENV.SERVER_NT_DOMAIN]
	DEST_BASE_PATH	[DEST_ENV.SERVER_BASE_PATH]
	CONNECTION_ PROTOCOL	[DEST_ENV.SERVER_CON_PROTOCOL _MEANING]
	DEST_ENV	[DEST_ENV]
ksc_connect_source_client	USERNAME	[SOURCE_ENV.CLIENT_USERNAME]
	PASSWORD	[SOURCE_ENV.CLIENT_PASSWORD]
	NT_DOMAIN	[SOURCE_ENV.CLIENT_NT_DOMAIN]
	SOURCE_BASE_ PATH	[SOURCE_ENV.CLIENT_BASE_PATH]
	CONNECTION_ PROTOCOL	[SOURCE_ENV.CLIENT_CON_PROTOC OL_MEANING]
	SOURCE_ENV	[SOURCE_ENV]

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_connect_source_server	USERNAME	[SOURCE_ENV.SERVER_USERNAME]
	PASSWORD	[SOURCE_ENV.SERVER_PASSWORD]
	NT_DOMAIN	[SOURCE_ENV.SERVER_NT_DOMAIN]
	SOURCE_BASE_ PATH	[SOURCE_ENV.SERVER_BASE_PATH]
	CONNECTION_ PROTOCOL	[SOURCE_ENV.SERVER_CON_PROTO COL_MEANING]
	SOURCE_ENV	[SOURCE_ENV]
ksc_copy_client_client	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_ PATH	[SOURCE_ENV.CLIENT_BASE_PATH]
	DEST_BASE_PATH	[DEST_ENV.CLIENT_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
ksc_copy_client_server	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_ PATH	[SOURCE_ENV.CLIENT_BASE_PATH]
	DEST_BASE_PATH	[DEST_ENV.SERVER_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_copy_server_client	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_PATH	[SOURCE_ENV.SERVER_BASE_PATH]
	DEST_BASE_PATH	[DEST_ENV.CLIENT_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
ksc_copy_server_server	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_ PATH	[SOURCE_ENV.SERVER_BASE_PATH]
	DEST_BASE_PATH	[DEST_ENV.SERVER_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
ksc_copy_client_tmp	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_ PATH	[SOURCE_ENV.CLIENT_BASE_PATH]
	DEST_BASE_PATH	[DEST_ENV.CLIENT_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]
ksc_copy_server_tmp	SUB_PATH	[P.P_SUB_PATH]
	SOURCE_BASE_ PATH	[SOURCE_ENV.SERVER_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	SOURCE_ENV	[SOURCE_ENV]

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_copy_tmp_client	SUB_PATH	[P.P_SUB_PATH]
	DEST_BASE_ PATH	[DEST_ENV.CLIENT_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	DEST_ENV	[DEST_ENV]
ksc_copy_tmp_server	SUB_PATH	[P.P_SUB_PATH]
	DEST_BASE_ PATH	[DEST_ENV.SERVER_BASE_PATH]
	FILENAME	[P.P_FILENAME]
	FILE_TYPE	[P.P_FILE_TYPE]
	DEST_ENV	[DEST_ENV]
ksc_copy_script_dest_client	SCRIPT_FILENAME	[P.P_SCRIPT_FILENAME]
	DEST_BASE_PATH	[DEST_ENV.CLIENT.BASE_PATH]
	DEST_ENV	[DEST_ENV]
ksc_copy_script_dest_server	SCRIPT_FILENAME	[P.P_SCRIPT_FILENAME]
	DEST_BASE_PATH	[DEST_ENV.SERVER_BASE_PATH]
	DEST_ENV	[DEST_ENV]
ksc_copy_script_source_client	SCRIPT_FILENAME	[P.P_SCRIPT_FILENAME]
	SOURCE_BASE_ PATH	[SOURCE_ENV.CLIENT_BASE_PATH]
	SOURCE_ENV	[SOURCE_ENV]
ksc_copy_script_source_ server	SCRIPT_FILENAME	[P.P_SCRIPT_FILENAME]
	SOURCE_BASE_ PATH	[SOURCE_ENV.SERVER_BASE_PATH]
	SOURCE_ENV	[SOURCE_ENV]
ksc_clear_exit_value		
ksc_end_script		
ksc_exit		

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_gl_migrate	CONC_PROGRAM	
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
	GL_ARCHIVE_FLAG	[WFS.GL_ARCHIVE_FLAG]
	APP_SHORT_NAME	
ksc_local_exec		
ksc_om_migrate	CONC_PROGRAM	
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
	OM_ARCHIVE_FLAG	[WFS.OM_ARCHIVE_FLAG]
	APP_SHORT_NAME	
ksc_parse_jcl	FILENAME	[P.P_FILENAME]
	OUTFILE	
	ENV_ID	[DEST_ENV.ENVIRONMENT_ID]
ksc_replace	FILENAME	[P.P_FILENAME]
	OUTFILE	
	SUBST	
ksc_respond		
ksc_run_sql		
ksc_set	Custom Token	
ksc_set_env	DEST_ENV_ID	[DEST_ENV.ENVIRONMENT_ID]
	SOURCE_ENV_ID	[SOURCE_ENV.ENVIRONMENT_ID]
	SOURCE_ENV	[SOURCE_ENV]
	DEST_ENV	[DEST_ENV]
ksc_set_exit_value		
ksc_simple_respond		
ksc_store	Custom Token	

Table A-24. Special Command Parameters

Special Command	Parameters	Defaults
ksc_submit_job	PATH	[AS.PKG_TRANSFER_PATH]
	FILENAME	[P.P_FILENAME]

Appendix B Tokens

This appendix provides a list of all entity Tokens. Use *Table B-1* as a quick reference guide to jump to the desired location.

Table B-1. Token Tables

Table	Page
Table B-2, App Server Properties	*96*
Table B-3, Budget	*97*
Table B-4, Contacts	*97*
Table B-5, Distribution	*98*
Table B-6, Environments	*99*
Table B-7, Environment Applications	*101*
Table B-8, Command Execution	*103*
Table B-9, Notifications	*104*
Table B-10, Organization Unit	*104*
Table B-11, Packages	*105*
Table B-12, Package Lines	*107*
Table B-13, Package Pending	*108*
Table B-14, Program	*109*
Table B-15, Projects	*109*
Table B-16, Project Details	*112*
Table B-17, Releases	*112*
Table B-18, Requests	*113*
Table B-1, Request Details	*116*

Table B-1. Token Tables

Table	Page
Table B-19, Request Pending	*116*
Table B-20, Report Submissions	*117*
Table B-21, Resource Pools	*118*
Table B-22, Security Groups	*119*
Table B-23, Skill	*119*
Table B-24, Staffing Profile	*120*
Table B-25, System	*121*
Table B-26, Tasks	*121*
Table B-27, Tasks Pending	*123*
Table B-28, Users	*124*
Table B-29, Validations	*126*
Table B-30, Validation Values	*126*
Table B-31, Workflows	*127*
Table B-32, Workflow Steps	*128*
Table B-33, Workflow Step Transaction	*130*

For a list of the Tokens that are associated with Field Groups, see *“Field Group Tokens”* on page 131.

System Tokens

Table B-2. App Server Properties

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

Note

Other App Server Properties Tokens are generated from the parameters in the server.conf file. For a description of each server parameter, see *System Administration Guide*.

Table B-3. Budget

Prefix	Token	Description
BGT	ACTIVE_FLAG	The active flag of 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 (Project, Program, or Org Unit) to which the Budget is linked.
BGT	BUDGET_IS_FOR_ID	The ID of the Project/Program/Org Unit to which the Budget is linked.
BGT	BUDGET_IS_FOR_NAME	The name of the Project/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 to which this Budget rolls up to.
BGT	BUDGET_ROLLS_UP_TO_NAME	The name of the Budget to which this Budget rolls up to.
BGT	BUDGET_URL	The URL to view this Budget.
BGT	CREATED_BY	The username of the user who created the Budget.
BGT	CREATION_DATE	The date when the Budget was created.
BGT	DESCRIPTION	The description of the Budget.
BGT	END_PERIOD	The end period of the Budget.
BGT	INITIATION_REQ	The initiation Request ID of the Budget.
BGT	PERIOD_SIZE	The period size of the Budget.
BGT	START_PERIOD	The start period of the Budget.
BGT	STATUS_CODE	The status code of the Budget.
BGT	STATUS_NAME	The status name of the Budget.

Table B-4. Contacts

Prefix	Token	Description
CON	COMPANY	The company the Contact works for.
CON	COMPANY_NAME	The name of the company the Contact works for.
CON	CONTACT_ID	The ID of the Contact (defined in the table KCRT_CONTACTS).
CON	CREATED_BY	The ID of the User that created the Contact.

Table B-4. Contacts

Prefix	Token	Description
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 that 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	USER_ID	The UserID of the Contact, if the Contact is a Mercury ITG Center user.
CON	USERNAME	The username of the Contact (if applicable). This may be a username for an external system, not necessarily Mercury ITG Center.

Table B-5. Distribution

Prefix	Tokens	Description
DIST	CREATED_BY	The ID of the user that created the Distribution.
DIST	CREATED_BY_USERNAME	The Mercury ITG Center Username of the user that created the Distribution.
DIST	DESCRIPTION	The description of the Release.
DIST	DISTRIBUTION_ID	The ID of the Distribution (defined in table KREL_DIESTRIBUTION).
DIST	DISTRIBUTION_NAME	The name of the Distribution.
DIST	DISTRIBUTION_STATUS	The Workflow status of the Distribution Workflow.
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 that last updated the Distribution.
DIST	LAST_UPDATED_BY_ USERNAME	The Mercury ITG Center username of the user that last updated the Distribution.
DIST	LAST_UPDATE_DATE	The date the Distribution was last updated.

Table B-5. Distribution

Prefix	Tokens	Description
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.

Table B-6. Environments

Prefix	Token	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 type is Windows.
ENV	CLIENT_ENABLED_FLAG	The flag indicating whether the client portion of the Environment is enabled.
ENV	CLIENT_PASSWORD	The password Mercury ITG Center uses to log onto or access the client. This value is encrypted.
ENV	CLIENT_TYPE_CODE	The Validation value code of the client machine type.
ENV	CLIENT_USERNAME	The username Mercury ITG Center uses to log onto 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 that created the Environment.
ENV	CREATION_DATE	The date the Environment was created.
ENV	DATABASE_ENABLED_FLAG	The flag indicating whether the database portion of the Environment is enabled.
ENV	DATABASE_TYPE	The Validation value code of the database type.
ENV	DB_CONNECT_STRING	For Oracle database type, the connect string used to access the database from the command line.

Table B-6. Environments [continued]

ENV	DB_LINK	For Oracle database type, the database link from the Mercury ITG 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 ITG Center uses to log onto 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 ITG Center uses to log onto or access the database.
ENV	DB_VERSION	The version of the database (such as 8.1.7).
ENV	DESCRIPTION	The description of the Environment.
ENV	ENABLED_FLAG	The flag indicating 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 name of the Environment.
ENV	LAST_UPDATED_BY	The ID of the User that last updated the Environment.
ENV	LAST_UPDATE_DATE	The date the Environment was last updated.
ENV	LOCATION	The location of the Environment.
ENV	MSSQL_DB_NAME	For MS 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_TRANSFER_PROTOCOL	The protocol used to transfer files to or from this server.

Table B-6. Environments [continued]

ENV	SERVER_TRANSFER_PROTOCOL_ MEANING	The visible value of the server transfer protocol.
ENV	SERVER_ENABLED_FLAG	The flag indicating 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 ITG Center uses to log onto 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 ITG Center uses to log onto or access the server.

Note

If any Mercury ITG Extensions have been installed, there will be more Environment Tokens with the prefix 'AC.' For more detailed information on these Tokens, see the Mercury ITG Extensions documentation.

Table B-7. Environment Applications

Prefix	Token	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 ITG Center uses to log onto or access the client. This value is encrypted.
ENV.APP	CLIENT_USERNAME	The Application-specific username Mercury ITG Center uses to log onto 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.

Table B-7. Environment Applications

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 that 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 ITG Center schema to the Environment's database schema.
ENV.APP	DB_NAME	For MS SQL Server database type, the Application-specific database name used to access the database from the command line.
ENV.APP	DB_PASSWORD	The Application-specific password Mercury ITG Center uses to log onto or access the database. This value is encrypted.
ENV.APP	DB_USERNAME	The Application-specific username or schema name Mercury ITG Center uses to log onto or access the database.
ENV.APP	DESCRIPTION	The description of the Application.
ENV.APP	ENABLED_FLAG	The flag indicating 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 the Application is associated with.
ENV.APP	ENVIRONMENT_NAME	The name of the Environment the application is associated with.
ENV.APP	LAST_UPDATED_BY	The ID of the User that 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.

Table B-7. Environment Applications

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 ITG Center uses to log onto or access the server. This value is encrypted.
ENV.APP	SERVER_USERNAME	The Application-specific username Mercury ITG Center uses to log onto or access the server.
ENV.APP	WORKBENCH_ENVIRONMENT_ URL	The URL of the Environment window in the Workbench.

Table B-8. Command Execution

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

Note

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 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]'
```

Table B-9. Notifications

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.

Table B-10. Organization Unit

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 manager of the Org unit.

Table B-10. Organization Unit

Prefix	Tokens	Description
ORG	MANAGER_USERNAME	The name of the manager of the Org unit.
ORG	ORG_UNIT_ID	The ID of the Org unit (defined in table KRSC_ORG_UNITS).
ORG	ORG_UNIT_NAME	The name of the Org unit.
ORG	PARENT_ORG_UNIT_ID	The ID of the parent Org unit.
ORG	PARENT_ORG_UNIT_NAME	The name of the parent Org unit.

Table B-11. Packages

Prefix	Token	Description
PKG	ASSIGNED_TO_EMAIL	The email address of the User that the Package is assigned to.
PKG	ASSIGNED_TO_GROUP_ID	The ID of the Security Group that the Package has been assigned to.
PKG	ASSIGNED_TO_GROUP_NAME	The Security Group that the Package has been assigned to.
PKG	ASSIGNED_TO_USERNAME	The name of the User that the Package has been assigned to.
PKG	ASSIGNED_TO_USER_ID	The ID of the user that the Package has been assigned to.
PKG	CREATED_BY	The ID of the user that created the Package.
PKG	CREATED_BY_EMAIL	The email address of the User that created the Package.
PKG	CREATED_BY_USERNAME	The Mercury ITG Center username of the User that created the Package.
PKG	CREATION_DATE	The date the Package was created.
PKG	DESCRIPTION	The description of the Package.
PKG	ID	The ID of the Package in the table KDLV_PACKAGES.
PKG	LAST_UPDATED_BY	The ID of the User that last updated the Package.
PKG	LAST_UPDATED_BY_EMAIL	The email address of the User that last updated the Package.
PKG	LAST_UPDATED_BY_ USERNAME	The Mercury ITG Center username of the User that last updated the Package.
PKG	LAST_UPDATE_DATE	The date the Package was last updated.

Table B-11. Packages

Prefix	Token	Description
PKG	MOST_RECENT_NOTE_ AUTHOR_FULL_NAME	First and last name of the author of the most recent (chronological) note.
PKG	MOST_RECENT_NOTE_ AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
PKG	MOST_RECENT_NOTE_ AUTHORED_DATE	Date of the most recent (chronological) note.
PKG	MOST_RECENT_NOTE_TEXT	Text of the most recent (chronological) note.
PKG	NOTES	All notes for the Package.
PKG	NUMBER	The name/number of the Package.
PKG	PACKAGE_GROUP_CODE	The Package Group code.
PKG	PACKAGE_GROUP_NAME	The name of the Package Group.
PKG	PARENT_REQUEST_ID	The ID of the Request that created this Package (if applicable).
PKG	PRIORITY	The priority of the Package.
PKG	PRIORITY_CODE	The Validation value code of the Package priority.
PKG	PRIORITY_NAME	The Validation value meaning of the Package priority.
PKG	PRIORITY_SEQ	The priority sequence of the Package.
PKG	PROJECT_CODE	The Validation value code of the Project the Package belongs to.
PKG	PROJECT_NAME	The Validation value meaning of the Project the Package belongs to.
PKG	SUBMIT_DATE	The date that the Package was submitted.
PKG	REQUESTED_BY_EMAIL	The email address of the User who requested the Package.
PKG	REQUESTED_BY_USERNAME	The Mercury ITG Center username of the User who requested the Package.
PKG	REQUESTED_BY_USER_ID	The ID of the user that requested the Package.
PKG	PACKAGE_ID	The ID of the Package in the table KDLV_PACKAGES.
PKG	PACKAGE_NO_LINK	Shows up as a standard hyperlink to the Package in HTML-format Notifications.
PKG	PACKAGE_TYPE	The Validation value meaning of the Package type.
PKG	PACKAGE_TYPE_CODE	The Validation value code of the Package type.

Table B-11. Packages

Prefix	Token	Description
PKG	PACKAGE_URL	The URL of the Package in the standard interface.
PKG	PERCENT_COMPLETE	Percent complete of the Package.
PKG	RUN_GROUP	The run group of the Package.
PKG	STATUS	The Validation value meaning for the status of the Package.
PKG	STATUS_CODE	The Validation value code for the status of the Package.
PKG	WORKBENCH_PACKAGE_NO_ LINK	The URL of the Package in the Workbench.
PKG	WORKBENCH_PACKAGE_URL	The URL of the Package screen in the Workbench.
PKG	WORKFLOW_ID	The ID of the Workflow used by the Package.
PKG	WORKFLOW_NAME	The name of the Workflow used by the Package.

Table B-12. Package Lines

Prefix	Token	Description
PKGL	APP_CODE	The App 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).

Table B-12. Package Lines

Prefix	Token	Description
PKGL	WORKBENCH_OBJECT_ TYPE_URL	URL to access the Object Type window for this Object Type in the Workbench.

Table B-13. Package Pending

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.
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.

Table B-13. Package Pending

Prefix	Tokens	Description
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.

Table B-14. Program

Prefix	Token	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.

Table B-15. Projects

Prefix	Tokens	Description
PRJ	ACTUAL_DURATION	The actual duration of the Project.
PRJ	ACTUAL_EFFORT	The actual effort associated with the Project.
PRJ	ACTUAL_FINISH_DATE	The actual finish date of the Project.
PRJ	ACTUAL_START_DATE	The actual start date of the Project.

Table B-15. Projects

Prefix	Tokens	Description
PRJ	BUDGET_ID	The ID of the Budget linked to the Project.
PRJ	BUDGET_NAME	The name of the Budget linked to the Project.
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 Project.
PRJ	CREATED_BY_EMAIL	The email address of the user who created the Project.
PRJ	CREATED_BY_USERNAME	The username of the person who created the Project.
PRJ	CREATION_DATE	The creation date of the Project.
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 Project.
PRJ	ESTIMATED_REMAINING_ DURATION	The estimated remaining duration of the Project.
PRJ	ESTIMATED_REMAINING_ EFFORT	The estimated remaining effort involved in the Project.
PRJ	ESTIMATED_FINISH_DATE	The estimated finish date of the Project.
PRJ	LAST_UPDATE_DATE	The date the Project was last updated.
PRJ	LAST_UPDATED_BY	The last person to update the Project.
PRJ	LAST_UPDATED_BY_EMAIL	The email address of the last person to update the Project.
PRJ	LAST_UPDATED_BY_ USERNAME	The username of the last person to update the Project.
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 (chronological) note.
PRJ	MOST_RECENT_NOTE_ AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
PRJ	MOST_RECENT_NOTE_ AUTHORED_DATE	Date of the most recent (chronological) note.
PRJ	MOST_RECENT_NOTE_TEXT	Text of the most recent (chronological) note.

Table B-15. Projects

Prefix	Tokens	Description
PRJ	MOST_RECENT_NOTE_TYPE	Type of the most recent (chronological) 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	MOST_RECENT_USER_NOTE _TYPE	Type of the most recent user note (USER or FIELD CHANGE).
PRJ	PARENT_PROJECT_ID	The ID of the parent Project.
PRJ	PARENT_PROJECT_NAME	The name of the parent Project.
PRJ	PERCENT_COMPLETE	The Project's completed percentage.
PRJ	PRIORITY	The priority of the Project.
PRJ	PROJECT_ID	The number that uniquely identifies the Project (same as PROJECT_NUMBER) in the table KDRV_PROJECTS.
PRJ	PROJECT_MANAGER	The Manager of the Project.
PRJ	PROJECT_MANAGER_EMAIL	The email address of the Project Manager.
PRJ	PROJECT_MANAGER_ USERNAME	The username of the Project Manager.
PRJ	PROJECT_NAME	The name of the Project.
PRJ	PROJECT_NAME_LINK	Shows up as a standard hyperlink to the Project in HTML-format Notifications.
PRJ	PROJECT_NUMBER	The number that uniquely identifies the Project (same as PROJECT_ID).
PRJ	PROJECT_PATH	The Project Path. This is a hierarchy of parent Projects that contain this Project.
PRJ	PROJECT_STATE	The Project State.
PRJ	PROJECT_TEMPLATE	The name of the Project Template used to create the Project.
PRJ	PROJECT_TYPE_CODE	Returns TASK for Tasks and PROJECT for Projects.

Table B-15. Projects

Prefix	Tokens	Description
PRJ	PROJECT_URL	The URL for the Project Overview.
PRJ	SCHEDULED_EFFORT	The scheduled effort defined in the Project.
PRJ	SCHEDULED_DURATION	The Project's scheduled duration.
PRJ	SCHEDULED_FINISH_DATE	The Project's scheduled finish date.
PRJ	SCHEDULED_START_DATE	The Project's scheduled start date.
PRJ	SUMMARY_CONDITION	The Project's Summary Condition.
PRJ	WORKBENCH_PROJECT_URL	The URL to access this Project in the Workbench.

Table B-16. Project Details

Prefix	Tokens*	Description
PRJD	PROJECT_DETAIL_ID	The ID of Project Detail in the table KDRV_PROJECT_DETAILS.
PRJD	PROJECT_ID	The ID of the Project in the table KDRV_PROJECT_DETAILS.

* Parameters are accessible with this prefix (similar to Request Detail): [PRJD.P.CUSOM_TOKEN].

Table B-17. Releases

Prefix	Tokens	Description
REL	RELEASE_ID	The ID of the Release in the table KREL_RELEASES.
REL	RELEASE_NAME	The name of the Release.
REL	RELEASE_STATUS	The status of the Release.
REL	CREATED_BY	The ID of the user who created the Release.
REL	CREATED_BY_USERNAME	The Mercury ITG 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 ITG Center username of the user who last updated the Release.
REL	LAST_UPDATE_DATE	The date that the Release was last updated.
REL	MOST_RECENT_NOTE_ AUTHOR_FULL_NAME	First and last name of the author of the most recent (chronological) note.

Table B-17. Releases

Prefix	Tokens	Description
REL	MOST_RECENT_NOTE_ AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
REL	MOST_RECENT_NOTE_ AUTHORED_DATE	Date of the most recent (chronological) note.
REL	MOST_RECENT_NOTE_ TEXT	Text of the most recent (chronological) note.
REL	RELEASE_MANAGER	The Mercury ITG Center user who is designated the Release Manager.
REL	RELEASE_TEAM	The group of Mercury ITG Center users associated with the Release.
REL	RELEASE_GROUP	The high level categorization of the Release.
REL	DESCRIPTION	The description of the Release.
REL	NOTES	The notes contained within the Release.

Table B-18. Requests

Prefix	Token	Description
REQ	APPLICATION_CODE	The Validation value code for the application that the Request is assigned to.
REQ	APPLICATION_NAME	The Validation value meaning of the application that the Request is assigned to.
REQ	ASSIGNED_TO_EMAIL	The email address of the user the Request has been assigned to.
REQ	ASSIGNED_TO_GROUP_ID	The ID of the Security Group that the Request has been assigned to.
REQ	ASSIGNED_TO_GROUP_NAME	The name of the Security Group that the Request has been assigned to.
REQ	ASSIGNED_TO_USERNAME	The Mercury ITG Center username of the user that the Request has been assigned to.
REQ	ASSIGNED_TO_USER_ID	The ID of the user that the Request has been assigned to.
REQ	COMPANY	The Company employing the user that created the Request.
REQ	COMPANY_NAME	The name of the Company employing the user that created the Request.

Table B-18. Requests

Prefix	Token	Description
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 that created the Request.
REQ	CREATED_BY_EMAIL	The email address of the user that created the Request.
REQ	CREATED_BY_USERNAME	The Mercury ITG Center username of the user that created the Request.
REQ	CREATION_DATE	The date 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 description of the Request.
REQ	LAST_UPDATED_BY	The ID of the user that last updated the Request.
REQ	LAST_UPDATED_BY_EMAIL	The email address of the user that last updated the Request.
REQ	LAST_UPDATED_BY_ USERNAME	The Mercury ITG Center username of the user that last updated the Request.
REQ	LAST_UPDATE_DATE	The date the Request was last updated.
REQ	MOST_RECENT_NOTE_ AUTHOR_FULL_NAME	First and last name of the author of the most recent (chronological) note.
REQ	MOST_RECENT_NOTE_ AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
REQ	MOST_RECENT_NOTE_ AUTHORED_DATE	Date of the most recent (chronological) note.
REQ	MOST_RECENT_NOTE_TEXT	Text of the most recent (chronological) note.
REQ	MOST_RECENT_NOTE_TYPE	Type of the most recent (chronological) 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 name of the author of the most recent user note.

Table B-18. Requests

Prefix	Token	Description
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	Status of the Request.
REQ	NOTES	All notes for the Request.
REQ	PERCENT_COMPLETE	The percent complete of the Request.
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 Project the Request belongs to.
REQ	PROJECT_NAME	The Validation value meaning of the Project the Request belongs to.
REQ	SUBMIT_DATE	The date that the Request was submitted.
REQ	REQUEST_GROUP_CODE	The code for the Request Group.
REQ	REQUEST_GROUP_NAME	The name of the Request Group.
REQ	REQUEST_ID	The ID of the Request in the table KCRT_REQUESTS.
REQ	REQUEST_ID_LINK	Shows up as a standard hyperlink to the Request in HTML-format 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 of the Request.
REQ	REQUEST_URL	URL of the Request in standard interface.
REQ	STATUS_ID	The ID of the status of the Request.
REQ	STATUS_NAME	The status of the Request.

Table B-18. Requests

Prefix	Token	Description
REQ	WORKBENCH_REQUEST_ TYPE_URL	The URL of the Request Type in the Workbench.
REQ	WORKFLOW_ID	The ID of the Workflow used by the Request.
REQ	WORKFLOW_NAME	The name of the Workflow used by the Request.

Figure B-1 Request Details

Prefix*	Tokens	Description
REQD	CREATED_BY	The ID of the User who created the Request Detail.
REQD	CREATION_DATE	The date the Request Detail was created.
REQD	LAST_UPDATED_BY	The ID of the User that last updated the Request Detail.
REQD	LAST_UPDATE_DATE	The date the Request Detail was last updated.
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.

* Prefix is mainly used for accessing custom fields: `[REQD.P.CUSTOM_TOKEN]`

Table B-19. Request Pending

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

Table B-19. Request Pending

Prefix	Tokens	Description
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 is being blocked by the Request.
REQ.PEND	ASSIGNED_TO_USER_ID	The username of the assigned user (or resource) of the entity that is being blocked by the Request.
REQ.PEND	ASSIGNED_TO_GROUP_NAME	The name of the assigned group (or resource group) of the entity that is being blocked by the Request.
REQ.PEND	ASSIGNED_TO_GROUP_ID	The ID of the assigned group (or resource group) of the entity that is being blocked by the Request.
REQ.PEND	RESOURCE_USERNAME	The name of the resource associated with the entity that is being blocked by the Request.
REQ.PEND	RESOURCE_ID	The username of the assigned user (or resource) associated with the entity that is being blocked by the Request.
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.
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 is being blocked by the Request.
REQ.PEND	ENTITY_TYPE_ID	The ID of the type of entity that is being blocked by the Request.
REQ.PEND	ENTITY_TYPE_NAME	The name of the type of entity that is being blocked by the Request.

Table B-20. Report Submissions

Prefix	Tokens	Description
RP	CREATED_BY	The ID of the User who submitted the Report.
RP	CREATION_DATE	The date the Report was submitted.
RP	FILENAME	The filename for the Report. This file name is found in the REPORT_URL.

Table B-20. Report Submissions

RP	LAST_UPDATED_BY	The ID of the User that last updated the Report submission.
RP	LAST_UPDATE_DATE	The date the Report submission was last updated.
RP	NEW_STATUS	The visible value for the Report’s new Status.
RP	NEW_STATUS_CODE	The code for the Report’s new Status.
RP	OLD_STATUS	The visible value for the Report’s old status.
RP	OLD_STATUS_CODE	The code for the Report’s old status.
RP	REPORT_LOG_URL	The Web address where the Report log is located.
RP	REPORT_SUBMISSION_ID	The ID of the Report submission in the table KNTA_REPORT_SUBMISSIONS.
RP	REPORT_TYPE_NAME	The name of the Report Type of the Report submission.
RP	REPORT_TYPE_ID	The ID of the Report Type of the Report submission.
RP	REPORT_URL	The Web address where the Report output is located.
RP	STATUS	The status of the Report submission.
RP	STATUS_CODE	The Validation value code for the status of the Report submission.
RP	WORKBENCH_REPORT_TYPE_URL	The URL of the Report Type in the Workbench.

Table B-21. Resource Pools

Prefix	Tokens	Description
RSCP	ACTIVE_FLAG	The active flag of the Resource Pool.
RSCP	CREATED_BY	The username of the user who created the Resource Pool.
RSCP	CREATION_DATE	The date that the Resource Pool was created.
RSCP	DESCRIPTION	The description of the Resource Pool.
RSCP	END_PERIOD	The end period of the Resource Pool.
RSCP	INITIATION_REQ	The initiation Request ID of the Resource Pool.
RSCP	PERIOD_SIZE	The period size of the Resource Pool.
RSCP	RESOURCE_POOL_URL	The URL to view this Resource Pool.

Table B-21. Resource Pools

Prefix	Tokens	Description
RSCP	RSC_POOL_ID	The ID of the Resource Pool in table KRSC_RSC_POOLS.
RSCP	RSC_POOL_IS_FOR_ENTITY_NAME	The entity name to which the Resource Pool is linked (Program or Org Unit).
RSCP	RSC_POOL_IS_FOR_ID	The ID of the Program or Org unit to which the Resource Pool is linked.
RSCP	RSC_POOL_IS_FOR_NAME	The name of the Program or ORg unit to which the Resource Pool is linked.
RSCP	RSC_POOL_NAME	The name of the Resource Pool.
RSCP	START_PERIOD	The start period of the Resource Pool.
RSCP	STATUS_CODE	The status code of the Resource Pool.
RSCP	STATUS_NAME	The status name of the Resource Pool.

Table B-22. Security Groups

Prefix	Tokens	Description
SG	CREATED_BY	The ID of the User who created the Security Group.
SG	CREATION_DATE	The date the Security Group was created.
SG	DESCRIPTION	The description for the Security Group.
SG	LAST_UPDATED_BY	The ID of the User that last updated the Security Group.
SG	LAST_UPDATE_DATE	The date 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 name of the Security Group.

Table B-23. Skill

Prefix	Tokens	Description
SKL	AVERAGE_COST_RATE	The average cost rate associated with the skill.
SKL	CREATED_BY	The user ID that created the Skill.
SKL	CREATED_BY_USERNAME	The name of the user that created the Skill.
SKL	CREATION_DATE	The date that the Skill was created.

Table B-23. Skill

Prefix	Tokens	Description
SKL	SKILL_CATEGORY_CODE	The lookup code of 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 name of the Skill.

Table B-24. Staffing Profile

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 that the Staffing Profile was created.
STFP	DESCRIPTION	The description of the Staffing Profile.
STFP	END_PERIOD	The end period of the Staffing Profile.
STFP	INITIATION_REQ	The initiation Request ID 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.
STFP	STAFF_PROF_IS_FOR_ID	The ID of the Project, Program or Org unit to which the Staffing Profile is linked.
STFP	STAFF_PROFL_IS_FOR_ NAME	The name of the Project, Program or Org unit to which the Staffing Profile is linked (Project, Program, or Org Unit).
STFP	STAFF_PROF_NAME	The name of the Staffing Profile.
STFP	START_PERIOD	The start period of the Staffing Profile.
STFP	STATUS_CODE	The status code of the Staffing Profile.
STFP	STATUS_NAME	The status name of the Staffing Profile.

Table B-25. System

Prefix	Tokens	Description
SYS	DATE	The date at the time the Token is parsed.
SYS	NEWLINE	A new line character.
SYS	TIME_STAMP	A date and time stamp at the time the Token is parsed.
SYS	UNIQUE_IDENTIFIER	Used to obtain a unique number from the database. It can be used to generate unique filenames, etc. It is often necessary to use with the 'ksc_set' Special Command.
SYS	UNIX_NEWLINE	The UNIX new line character.
SYS	USERNAME	The Mercury ITG Center username of the User currently logged onto Mercury ITG Center.
SYS	USER_ID	The ID of the User currently logged onto Mercury ITG Center.

Table B-26. Tasks

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.
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.

Table B-26. Tasks

Prefix	Tokens	Description
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 the Task was last updated.
TSK	LAST_UPDATED_BY	The last person 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 (chronological) note.
TSK	MOST_RECENT_NOTE_ AUTHOR_USERNAME	Username of the author of the most recent (chronological) note.
TSK	MOST_RECENT_NOTE_ AUTHORED_DATE	Date of the most recent (chronological) note.
TSK	MOST_RECENT_NOTE_TEXT	Text of the most recent (chronological) note.
TSK	PARENT_PROJECT_ID	The ID of the parent Project.
TSK	PARENT_PROJECT_NAME	The name of the parent Project.
TSK	PERCENT_COMPLETE	The Task's completed percentage.
TSK	PRIORITY	The priority of the Task.
TSK	PROJECT_PATH	The Project Path. Hierarchy of parent Projects that contain this Task.
TSK	PROJECT_TEMPLATE	The name of the Project Template used to create the Project containing the Task.
TSK	PROJECT_TYPE_CODE	Returns TASK for Tasks and PROJECT for Projects.
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.

Table B-26. Tasks

Prefix	Tokens	Description
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 Task's scheduled duration.
TSK	SCHEDULED_FINISH_DATE	The Task's scheduled finish date.
TSK	SCHEDULED_START_DATE	The Task's scheduled start date.
TSK	SCHEDULING CONSTRAINT	The Task's scheduling constraint.
TSK	TASK_CATEGORY	The predefined category the Task belongs to.
TSK	TASK_ID	The number that uniquely identifies the Task (same as TASK_NUMBER). This corresponds to the PROJECT_ID column in table KDRV_PROJECTS.
TSK	TASK_NAME	The name of the Task.
TSK	TASK_NAME_LINK	Standard hyperlink to the Task in HTML-format 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.

Table B-27. Tasks Pending

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 panel.
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.

Table B-27. Tasks Pending

Prefix	Tokens	Description
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.
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.

Table B-28. Users

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 employing the user.

Table B-28. Users

USR	COMPANY_NAME	The name of the Company employing the user.
USR	COST_RATE	The cost rate of the user ($/hour - subject to security of user evaluating the Token).
USR	CREATED_BY	The ID of the user that created the user.
USR	CREATED_BY_USERNAME	The Mercury ITG Center username of the user that created the user.
USR	CREATION_DATE	The date 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 that the user belongs to.
USR	EMAIL_ADDRESS	The email address of the user.
USR	END_DATE	The date 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_UPDATED_BY_USERNAME	The Mercury ITG 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 onto Mercury ITG 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_SKILL_ID	The ID of the primary skill associated with the user.
USR	PRIMARY_SKILL_NAME	The name of the primary skill associated with the user.
USR	RESOURCE_CATEGORY_CODE	The lookup code of Resource Category (lookup type = RSC - Category) to which the user belongs.

Table B-28. Users

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 onto Mercury ITG 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)

Table B-29. Validations

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.
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.

Table B-30. Validation Values

Prefix	Tokens	Description
VALUE	CREATED_BY	The ID of the user that created the value.
VALUE	CREATION_DATE	The date the value was created.

Table B-30. Validation Values

VALUE	DEFAULT_FLAG	The flag to indicate whether the value is the default value for the associated lookup type.
VALUE	DESCRIPTION	The description of the value.
VALUE	ENABLED_FLAG	The flag to indicate whether the value is enabled for selection in a drop-down list.
VALUE	LAST_UPDATED_BY	The ID of the user that last updated the value.
VALUE	LAST_UPDATE_DATE	The date the value was last updated.
VALUE	LOOKUP_CODE	The code associated with the value.
VALUE	LOOKUP_TYPE	The lookup type the value belongs to.
VALUE	MEANING	The meaning associated with the value.
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.

Table B-31. Workflows

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.

Table B-31. Workflows

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.

Table B-32. Workflow Steps

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.
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 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.

Table B-32. Workflow Steps

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).
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 that the result of the Workflow Step is written to.
WFS	RESULT_WORKFLOW_PARAMETER_ NAME	The name of the Workflow parameter that the result of the Workflow Step is written to.
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 name of the Workflow Step.
WFS	STEP_NO	The display sequence of the Workflow Step relative to all other Steps in the Workflow.

Table B-32. Workflow Steps

WFS	STEP_SOURCE_NAME	The name of the Workflow Step Source.
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.

Table B-33. Workflow Step Transaction

Prefix	Tokens	Description
WST	CONCURRENT_REQUEST_ID	The ID of the concurrent request that was launched in Oracle Applications.
WST	CREATED_BY	The ID of the user that created the Step transaction.
WST	CREATION_DATE	The date the Step transaction was created.
WST	ERROR_MESSAGE	The error message for the Step transaction.
WST	EXECUTION_BATCH_ID	The ID of the Execution Batch for the Workflow Step.
WST	HIDDEN_STATUS	The hidden value for the status of the Step transaction.
WST	LAST_UPDATED_BY	The ID of the user that last updated the Step transaction.
WST	LAST_UPDATED_BY_EMAIL	The email address of the user that last updated the Step transaction.
WST	LAST_UPDATED_BY_USERNAME	The Mercury ITG Center username of the user that last updated the Step transaction.
WST	LAST_UPDATE_DATE	The date the Step transaction was last updated.
WST	NEW_HIDDEN_STATUS	The new hidden value for the status of the Step transaction.
WST	NEW_STATUS	The new status of the Step transaction.
WST	OLD_HIDDEN_STATUS	The old hidden value for the status of the Step transaction.
WST	OLD_STATUS	The old status of the Step transaction.
WST	STATUS	The status of the Step transaction.

Table B-33. Workflow Step Transaction

WST	STEP_TRANSACTION_ID	The ID of the Step transaction in the table KWFL_STEP_TRANSACTIONS.
WST	TIMEOUT_DATE	The date that the Step transaction times out.
WST	USER_COMMENT	The user comment for the Step transaction.
WST	WORKFLOW_ID	The ID of the Workflow for the Step transaction.
WST	WORKFLOW_STEP_ID	The ID of the Workflow Step for the Step transaction.

Field Group Tokens

Field Groups can be attached to Request Header Types to enable additional pre-configured fields on Requests. Field Groups are often delivered as a part of Mercury ITG Center Best Practice functionality. You will only have access to Field Groups associated with products that are licensed at your site.

Use *Table B-34* as a quick reference guide to jump to the desired location.

Table B-34. Field Group Token Tables

Table	Page
Table B-35, Demand Management Field Group Tokens	*132*
Table B-36, Master Project Reference on Request Field Group Tokens	*132*
Table B-37, PFM Asset Field Group Tokens	*132*
Table B-38, PFM Project Field Group Tokens	*133*
Table B-39, PFM Proposal Field Group Tokens	*134*
Table B-40, PMO Field Group Tokens	*134*
Table B-41, Program Reference on Request Field Group Tokens	*135*
Table B-42, Work Item Field Group Tokens	*135*

Table B-35. Demand Management Field Group Tokens

Field	Token
SLA Level	KNTA_SLA_LEVEL
SLA Violation Data	KNTA_SLA_VIOLATION_DATE
Service Request Date	KNTA_SLA_SERV_REQUESTED_ON
Service Satisfied Date	KNTA_SLA_SERV_SATISFIED_ON
Estimated Start Date	KNTA_EST_START_DATE
Estimated Effort	KNTA_EFFORT
Reject Date	KNTA_REJECTED_DATE
Demand Satisfied Date	KNTA_DEMAND_SATISFIED_DATE

Table B-36. Master Project Reference on Request Field Group Tokens

Field	Token
Master Project	KNTA_MASTER_PROJ_REF

Table B-37. PFM Asset Field Group Tokens

Field	Token
Business Unit	KNTA_BUSINESS_UNIT
Asset Name	KNTA_PROJECT_NAME
Asset Health	KNTA_PROJECT_HEALTH
Project Class	KNTA_PROJECT_CLASS
Asset Class	KNTA_ASSET_CLASS
Business Objective	KNTA_BUSINESS_OBJECTIVE
Project Plan	KNTA_PROJECT_PLAN
Budget	KNTA_BUDGET
Financial Benefit	KNTA_FINANCIAL_BENEFIT
Staffing Profile	KNTA_STAFFING_PROFILE
Net Present Value	KNTA_NET_PRESENT_VALUE
Value Rating	KNTA_VALUE_RATING

Table B-37. PFM Asset Field Group Tokens

Field	Token
Risk Rating	KNTA_RISK_RATING
Return on Investment	KNTA_RETURN_ON_INVESTMENT
Custom Field Value	KNTA_CUSTOM_FIELD_VALUE
Total Score	KNTA_TOTAL_SCORE
Discount Rate	KNTA_DISCOUNT_RATE

Table B-38. PFM Project Field Group Tokens

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

Table B-39. PFM Proposal Field Group Tokens

Field	Token
Business Unit	KNTA_BUSINESS_UNIT
Project Name	KNTA_PROJECT_NAME
Project Class	KNTA_PROJECT_CLASS
Asset Class	KNTA_ASSET_CLASS
Business Objective	KNTA_BUSINESS_OBJECTIVE
Project Template	KNTA_PROJECT_TEMPLATE
Project Manager	KNTA_PROJECT_MANAGER
Budget	KNTA_BUDGET
Expected Benefit	KNTA_FINANCIAL_BENEFIT
Staffing Profile	KNTA_STAFFING_PROFILE
Net Present Value	KNTA_NET_PRESENT_VALUE
Value Rating	KNTA_VALUE_RATING
Risk Rating	KNTA_RISK_RATING
Return on Investment	KNTA_RETURN_ON_INVESTMENT
Custom Field Value	KNTA_CUSTOM_FIELD_VALUE
Total Score	KNTA_TOTAL_SCORE
Discount Rate	KNTA_DISCOUNT_RATE

Table B-40. PMO Field Group Tokens

Field	Token
Escalation Level	KNTA_ESCALATION_LEVEL
Role Description	KNTA_ROLE_DESCRIPTION
Risk Impact Level	KNTA_RISK_IMPACT_LEVEL
Probability	KNTA_PROBABILITY
CR Level	KNTA_CR_LEVEL
Business Impact Severity	KNTA_IMPACT_SEVERITY

Table B-41. Program Reference on Request Field Group Tokens

Field	Token
Program	KNTA_PROGRAM_REFERENCE

Table B-42. Work Item Field Group Tokens

Field	Token
Scheduled Start Date	KNTA_USR_SCHED_START_DATE
Actual Start Date	KNTA_USR_ACTUAL_START_DATE
Scheduled Finish Date	KNTA_USR_SCHED_FINISH_DATE
Actual Finish Date	KNTA_USR_ACTUAL_FINISH_DATE
Scheduled Duration	KNTA_SCHED_DURATION
Actual Duration	KNTA_ACTUAL_DURATION
Scheduled Effort	KNTA_SCHED_EFFORT
Actual Effort	KNTA_ACTUAL_EFFORT
Workload?	KNTA_WORKLOAD
Workload Category	KNTA_WORKLOAD_CATEGORY
Skill	KNTA_SKILL
Allow External Update of Actual Effort	KNTA_ALLOW_EXTERNAL_UPDATE
_Scheduled Start Date	KNTA_SCHED_START_DATE
_Actual Start Date	KNTA_ACTUAL_START_DATE
_Scheduled Finish Date	KNTA_SCHED_FINISH_DATE
_Actual Finish Date	KNTA_ACTUAL_FINISH_DATE
_Scheduled Effort Over Duration	KNTA_SCHED_EFF_OVER_DUR

Index