Mercury IT Governance Center™

Service-Oriented Architecture: Web Services Guide

Version: 6.0

The information in this document, while based on release 6.0, is generally applicable to release 7.0 of Mercury IT Governance Center. However, the Mercury Time Management Web services described in this version of the document are not supported. An update of this document for release 7.0 is planned for the near future.

MERCURY™

Mercury
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

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

Publication Number: ITG60SOA0605B

Table of Contents

List of Tables and Figures

Tables

Figures

Chapter 1 Introduction

In This Chapter:

About This Document

Mercury IT Governance Center™ has been enhanced to enable a service-oriented architecture (SOA). The implementation utilizes Web services and is limited to the following:

- Mercury Demand Management™ and its corresponding workflow engine
- Mercury Time Management™
- Mercury Financial Management™

This document describes the Mercury IT Governance Center Web services interface and implementation. The chapters address the following topics:

- Chapter 1, *Introduction,* on page 9

  Describes the organization and use of this document, plus provides a brief overview of service-oriented architecture.

- Chapter 2, *Mercury IT Governance Center Implementations,* on page 13

  Provides overviews of the various Mercury IT Governance Center SOA implementations.

- Chapter 3, *Installation and Configuration,* on page 21

  Describes the installation and configuration necessary to take advantage of this new functionality.

- Chapter 4, *Understanding the Web Services Implementation,* on page 27

  Describes the relationship between Web services and Mercury IT Governance Center as well as introducing new terminology.

- Chapter 5, *Examples,* on page 41

  Provides sample scenarios using Web services with Mercury IT Governance Center.

- Chapter 6, *Reference,* on page 49

  Describes the technical implementation details including the Mercury IT Governance Center special commands and the Web services calls.

Who Should Read This Document

This document is for the following audience types and assumes that the reader has some basic understanding of Web services concepts and implementations.

- Application developers and configurators
- User administrators
- System or instance administrators

For More Information

For information about audience types, see the *Guide to Documentation*.

Prerequisite Documents

Prerequisite documents include:

- *Getting Started*
- *Key Concepts*
- *Mercury Demand Management User's Guide*
- *Mercury Demand Management: Configuring a Request Resolution System*
- *Mercury Time Management User's Guide*
- *Mercury Time Management Configuration Guide*
- *Mercury Financial Management User's Guide*

For More Information

For information about these documents and how to access them, see the *Guide to Documentation*.

Related Documents

Supplemental documentation includes:

- *System Administration Guide and Reference*
- *Commands, Tokens, and Validation Guide and Reference*

For More Information

For information about these documents and how to access them, see the *Guide to Documentation*.

Commercially available documentation, such as *Java Web Services* by Dave Chappell and Tyler Jewell and published by O'Reilly and Associates, Inc., can also be helpful.

Overview of Service-Oriented Architecture

A service-oriented architecture allows business processes to interact in a platform- and language-neutral mode. These business processes can range from feature-rich, enterprise applications to discrete, stand-alone functions. A service-oriented architecture also enables distribution of the workload and functionality (provided by these business processes) across environments. The distributed processing can be highly localized, conducted throughout an enterprise, or global through the use of the Web.

Currently, implementation of a service-oriented architecture in Mercury IT Governance Center is based on Simple Object Access Protocol (SOAP) and Web services, both widely adopted and standardized interfaces.

Use of Web services provides a consistent interface for any non-Mercury IT Governance Center system calling into or out of Mercury IT Governance Center. This facilitates interoperability with business processes that are provided by third-party vendors or installed on non-supported platforms. SOAP is utilized for transactions between Mercury IT Governance Center instances.

Chapter

2

Mercury IT Governance Center Implementations

In This Chapter:

Mercury Demand Management

Web services are supported in Mercury Demand Management and its corresponding workflow engine. The following services are supported:

- Incoming
  - **Delete requests.** Deletes the specified Mercury IT Governance Center requests.
  - **Get request types by field group.** Returns a list of request types that contain the specified field group.
  - **Get request type field.** Returns the description of all the fields in the specified request type.
  - **Import request.** Creates a new Mercury IT Governance Center request.
  - **Update remote reference.** Receives status and field updates, and applies them on the specified Mercury IT Governance Center request.
- Outgoing
  - **Export request.** Special command `ksc_export_request`
  - **Export request status change.** Special command `ksc_export_status_change`, or the background PENDING_STATUS_CHANGE service
  - **Export request fields.** Special command `ksc_export_fields`

Exporting Requests

Request export is initiated through the Mercury IT Governance Center command-execution engine, by means of the `ksc_export_request` special command. When this command is executed, the following occurs:

1. All visible field values on the local request are gathered into a SOAP request.
2. The remote "import request" Web service is called with this SOAP request.

The request type name specified in the `ksc_export_request` command will be used in place of the local request's type and is used to determine which request type to use on the remote system when creating the imported request.

Only "visible" field values are exported. "Hidden" values (such as the internal keys behind the visible values displayed in the user interface) are not included in the exported SOAP request, since they typically represent information applicable only to the local environment. See *Request Fields* on page 32 for additional details regarding field processing during request import.

Notes on the local request can be included in the export, if specified by the special command. All notes are included, including both user notes and field-change notes. In the remote system, user notes will be appended with the message "(Originally added by *<author's full name>*)."

References on the local request can be included in the export, if specified by the special command. All references are included as URL links, with the following exceptions:

- Release references are not exported, since no corresponding URL link is available for a Mercury IT Governance Center release.
- Existing remote references are not exported.

On the remote system, if the exported SOAP request is successfully imported, a remote reference will be created that links the new request with its source. Remote reference creation can be disabled if specified by the special command.

On the local system, if the request was successfully exported, then a remote reference will also be created on the local request that links it to the corresponding remote request. Remote reference creation on the local request can be disabled if specified by the special command.

In the typical use case, the result of request export will be a remote request that has a remote reference back to the local request. The local request will also have a remote reference to the remote request.

Exporting Status Changes

Once a request is linked to a remote request through a remote reference, then the local request can send status updates to the remote request. This can occur in two ways:

- `ksc_export_status_change` special command

  The local request can send a status change to any remote references that it knows about. The special command takes a string argument that represents the status to be sent. When the command is invoked, the "update remote reference" Web service is called on all existing remote references.

  If the remote system is a Mercury IT Governance Center system, then it receives the status update and loads the request specified by the "update remote reference" Web service request. If this request has a remote reference back to the request that made the Web service call, then the status of that remote reference record is updated with the new status.

  In addition, if any eligible workflow step transition is found with a transition value that matches the new status, that transition is taken thereby progressing the workflow to the next step.

  Field updates can also be specified in the `ksc_export_status_change` command. These are specified as token-value pairs and will be included in the "update remote reference" SOAP request. Note that only "visible" value tokens are supported. When specifying the field values to be exported, remember to use the "visible" token syntax (as in "`REQD.VP.FIELD_X=[REQD.VP.FIELD_Y]`"). See *Request Fields* on page 32 for additional details regarding Web service field processing.

- PENDING_STATUS_CHANGE background service

  Requests in separate Mercury IT Governance Center instances that are linked to each other through remote references keep their statuses synchronized with the periodic PENDING_STATUS_CHANGE service. When this service is enabled, each change to the status field of a request is logged into the KNTA_PENDING_API_CHANGES database table. (Enabling the PENDING_STATUS_CHANGE service is discussed in Chapter 3, *Installation and Configuration,* on page 21.)

For each periodic run of the service, the following steps are completed:

- All records in KNTA_PENDING_API_CHANGES are queried for requests that have one or more remote references.
- For each request, all status changes (since the last run of the service) are loaded.
- The status changes are reviewed in chronological order, and for each status change, a `ksc_export_status_change` Web service call is sent to all remote references on that request.
- Each record in KNTA_PENDING_API_CHANGES is deleted as it is successfully processed.

Exporting Field Updates

As previously mentioned, token-value pairs can be exported along with a status change as part of the `ksc_export_status_change` command. If you want to send field updates with no status information, then the `ksc_export_fields` special command exists for this purpose.

Field updates are specified as token-value pairs on the special command. Note that only “visible” value tokens are supported. When specifying the field values to be exported, remember to use the “visible” token syntax (as in "`REQD.VP.FIELD_X=[REQD.VP.FIELD_Y]`"). See *Request Fields* on page 32 for additional details regarding Web service field processing.

The local request can send field updates to any remote references that it knows about. When the command is invoked, the “update remote reference” Web service is called on all existing remote references.

If the remote system is a Mercury IT Governance Center system, then it receives the field updates and loads the request specified by the “update remote reference” Web service request. Fields are processed and, if applicable, applied on the remote request.

Mercury Time Management

Web services are supported in Mercury Time Management. The following services are supported:

- Incoming
  - **Export time data.** Receives a filter containing the search criteria and returns the specified data from Mercury IT Governance Center.
- Outgoing
  - **Export time data.** Special command `ksc_export_actual_time`
  - **Export time data as XML.** Special command `ksc_export_actual_time_as_xml`

Exporting Time

Time export is initiated through the Mercury IT Governance Center command-execution engine by means of the `ksc_export_actual_time` or `ksc_export_actual_time_as_xml` special commands. When the command is executed, the following occurs:

1. All hours logged in different time sheets are gathered into a SOAP request.

   The start date, end date, work item type, and work item ID specified for the `ksc_export_actual_time` (or `ksc_export_actual_time_as_xml`) are used to restrict the time sheets. The SOAP request contains aggregated information by work item type, work item ID/name, resource, charge code, rate, and activity.

   For time logged against projects and tasks, the hours are rolled up to the master project. Also, resource and master user data is included in the SOAP request.

2. For `ksc_export_actual_time`, the remote "update" Web service is called.

   For `ksc_export_actual_time_as_xml`, the SOAP body is saved as an XML-formated file.

Mercury Financial Management

Web services are supported in Mercury Financial Management. The following services are supported:

- Incoming
  - **Create budget.** Creates a new budget on Mercury IT Governance Center.
  - **Export budget.** Receives a filter containing the search criteria and returns the specified data from Mercury IT Governance Center.
  - **Import budget.** Updates the budget data on Mercury IT Governance Center.
- Outgoing
  - **Export budget.** Special command `ksc_export_budget`
  - **Export budget as XML.** Special command `ksc_export_budget_as_xml`

Exporting Budgets

Budget export is initiated through the Mercury IT Governance Center command-execution engine by means of the `ksc_export_budget` or `ksc_export_budget_as_xml` special commands. When the command is executed, the following occurs:

1. Budget information is gathered into a SOAP request.

   The budget name specified for the `ksc_export_budget` (or `ksc_export_budget_as_xml`) is used to find the budget. The SOAP request contains the budget, budget line, and budget line detail information.

2. For `ksc_export_budget` the remote "update" Web service is called.

   For `ksc_export_budget_as_xml`, the SOAP body is saved as a XML-formated file.

Chapter

3

Installation and Configuration

In This Chapter:

Release 6.0 SP4

Installing Web Services

Software in support of Web services is included in the Service Pack 4 download bundle. To install SP4, see the README that accompanies the download bundle.

Both the download bundle and README file can be obtained from the Mercury IT Governance Download Center at the following URL:

http://itg.merc-int.com/support/download/login.jsp

Configuring Web Services

To configure Mercury IT Governance Server for Web services support, complete the following steps:

1. (Optional, although highly recommended) Create a backup copy of the Mercury IT Governance Center `<itg_home>/server.conf` file.
2. Open the `server.conf` file for edit.
3. Verify that the following base address is specified:

   ```
   com.kintana.core.server.BASE_URL=
                                  http://<hostname>:<port>/<itg>
   ```

   where the placeholders represent the following:

`<hostname>`	the hostname of the Mercury IT Governance Server
`<port>`	the port number for Web access
`<itg>`	the path where Mercury IT Governance Center is accessed

4. Enable Web services calls by adding (or modifying) the following setting:

   ```
   com.kintana.core.server.ENABLE_WEB_SERVICES=true
   ```

5. Enable the PENDING_STATUS_CHANGE service by modifying (or adding) the following settings:

```
com.kintana.core.server.PENDING_STATUS_CHANGE_SERVICE_
                                                    ENABLED=true

com.kintana.core.server.PENDING_STATUS_CHANGE_SERVICE_
                                                  DELAY=<seconds>

com.kintana.core.server.PENDING_STATUS_CHANGE_SERVICE_
                                                POOL_SIZE=<size>
```

   where the placeholders represent the following:

`<seconds>`	number of seconds before the next initiation of this service. Mercury recommends starting with 300 seconds (five minutes) and adjusting as necessary to ensure the desired system performance.
`<size>`	maximum number of threads for this service. Mercury recommends starting with five and adjusting as necessary to ensure the desired system performance.

6. Save the changes to the `server.conf` file.

7. Open the `<itg_home>/conf/webservices.conf` file for edit.

   For each remote server that will be called from the Mercury IT Governance Server, add the following setting in the credentials section:

```
<remoteServer baseURL="<protocol>://<hostname>:<port>/<path>
              username="<username>" password="<password>">
```

   where the placeholders represent the following:

`<protocol>`	either `http` or `https` (note case)
`<hostname>`	the hostname of the remote server
`<port>`	the port number for Web access
`<path>`	the path where remote server is accessed
`<username>`	the username for the remote server
`<password>`	the password for the remote server

   Make sure that the `baseURL` attributes match the `BASE_URL` key in the `server.conf` file of the Mercury IT Governance Centers instances you

will be calling, as well as the `URL` parameter for the `ksc_export_request` special command (see *ksc_export_request* on page 63)

8. Save the changes to the `webservices.conf` file.

9. Stop, then restart the Mercury IT Governance Server.

Using Web Services with HTTPS

To enable access to Mercury IT Governance Center Web services using HTTPS:

1. Enable Web services as described in *Configuring Web Services*.
2. Configure an external Web server to use SSL as described in the *System Administration Guide and Reference*.

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

You will need to do this if the certification authority (CA) is not one of the known certificate authorities that ship with the Java Runtime Environment (such as Verisign). If you use another CA, such as an authority internal to your organization, you will need to perform the following additional steps:

1. Use the following command to import the certificate into the keystore:

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

   where the placeholders represent the following:

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

   If the jssecacerts keystore does not already exist, a new one will be created with the password you specified.

   See http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html and http://www.churchillobjects.com/c/11201e.html for more information about the `keytool` utility.

2. Open the `server.conf` file for edit.

3. Add the following setting to the file:

   ```
   com.kintana.core.server.WEB_SERVICES_SSL_KEYSTORE_
                                                PASSWORD=<password>
   ```

   where the placeholder represents the following:

<password>	the password entered in step 1

4. Save the changes to the `server.conf` file.

If you call Web services on a remote server using HTTPS, both the baseURL attribute in the `webservices.conf` file on the local system and the BASE_URL key in the `server.conf` on the remote system should use the https:// prefix instead of http://.

Chapter

4

Understanding the Web Services Implementation

In This Chapter:

Mercury Web Services Concepts

This section introduces several concepts that are unique to, or are important for understanding Mercury's Web services implementation. Unless otherwise specified, the following information pertains to Mercury Demand Management, Mercury Time Management, and Mercury Financial Management.

Supervisor and Subordinate Relationships

The term "supervisor" is used to refer to the host that is governing the overall workflow processes.

The word "subordinate" denotes the system where the supporting business processes are being conducted.

Your business processes and workflows define whether Mercury IT Governance Center instances are supervisors, subordinates, both, or neither.

Local versus Remote

Terminology associated with the supervisor are generally discussed using their normal form (for example, workflow or request). Occasionally for clarity and distinction, these terms may be preceded by "local," further indicating that these are supervisor-centric (for example, local workflow).

When terms are associated with a subordinate, they are preceded by "remote" (for example, remote workflow or remote request).

Remote Reference

To facilitate interoperability with Mercury IT Governance Center, a new subentity has been created and termed "remote reference." This entity exists within the standard reference section of a Mercury Demand Management request in a "remote reference" subsection.

This remote reference provides a reference to an entity or object on a subordinate and provides a mechanism for status messaging.

Remote references include: identifier, description, URL, and status attributes. Each of these attributes are described in more detail later in this chapter.

Architecture Elements

Field-Mapping Requirements

For Mercury Demand Management, integration between Mercury IT Governance Center instances requires no middleware. Data is transmitted as SOAP requests. An optional XSL transformation file can be provided.

For integrations between Mercury IT Governance Center instances and non-Mercury IT Governance Center systems, the development of middleware may be required. Where XSL transformations are not sufficient, the middleware solution must handle field mappings and translations.

Authentication

The implementation of Web services security relies solely on secure connections via HTTPS. All Web services interactions are performed under a unique Web services account specified in the `server.conf` file as described in *Configuring Web Services* on page 22.

The Mercury IT Governance Server needs to have its own SSL server certificate (also described in *Configuring Web Services* on page 22). Additionally, client-side authentication is required and should be implemented using SSL server certificates.

Logging

Whenever a Web service is invoked on an Mercury IT Governance Center system, Mercury IT Governance Center verifies that the specified transaction name has not already been used by the calling system. If a previously-used name is provided, the Web service call is rejected and an error is returned.

If the transaction name has not already been used, the following information is added to the KNTA_OPENAPI_TXN_AUDIT table.

- Timestamp
- Request Header object including:
  - Caller name
  - Transaction name
  - Audit note
- Operation name (Web service call)

Security

For Mercury Time Management and Mercury Financial Management, all features described in this document (both Web services and Mercury IT Governance Center special commands) bypass budget and time sheet security. It is assumed that the external system accessing Mercury IT Governance Center has sufficient privileges. When budgets are exported, all budgets that match the criteria are exported regardless of access. Similarly, for time sheet information, all hours that match the criteria are exported regardless of access.

Also, budgets are created or updated regardless of security and budget status.

For Mercury Time Management, an update service is not provided.

Non-updateable Fields

For Mercury Time Management and Mercury Financial Management, the non-updateable fields are detailed in *Table 4-1*.

Table 4-1. Non-updateable fields

Field	Description
Budget name	Any change of value for the budget name in an incoming SOAP request generates an error.
Budget ID	Any change of value for the budget ID in an incoming SOAP request generates an error.
Associated with type	Any change of value for the type association in an incoming SOAP request generates an error.
Associated with name	Any change of value for the name association in an incoming SOAP request generates an error.
Start period name	Any value for the start period name in an incoming SOAP request is ignored.
Start period start date	Any change of value for the start period start date in an incoming SOAP request generates an error.
End period name	Any value for the end period name in an incoming SOAP request is ignored.
End period start date	Any change of value for the end period start date in an incoming SOAP request generates an error.
Period type	Any change of value for the period type in an incoming SOAP request generates an error.
Region	Any change of value for the region in an incoming SOAP request generates an error.
Base currency	Any change of value for the base curency in an incoming SOAP request generates an error.

Field-Level Security

All features described in this document (both Web services and related Mercury IT Governance Center special commands) bypass field-level security.

When a request is exported, all fields are exported. Conversely, when a request is imported, all fields are imported regardless of the account being used to access the Web service. However, these may be subject to the limitations described in *Request Fields* on page 32.

Remote References

For Mercury Demand Management, a remote reference represents a remote entity that is linked to a Mercury IT Governance Center request via Web services. Remote references appear in the References section in the user interface of a Mercury IT Governance Center request. They can only be created through Web services, by exporting or importing a Mercury IT Governance Center request. There is no facility in the user interface to create a remote reference.

Request Copy Functionality

For Mercury Demand Management, a special note is warranted here about copying requests. Because remote references are only created through an explicit Web service export/import transaction, they are not included in the standard request copy functionality. When a user copies a Mercury IT Governance Center request through the user interface, remote references are not copied.

Since exporting a request is essentially an extension of copying a request, any existing remote references on the request being exported are not included in the resulting SOAP request.

Remote references are created only through Web services, during Mercury IT Governance Center request export or import.

Request Fields

For Mercury Demand Management, when a request is imported or new field values are provided by either the `ksc_export_status_change` or `ksc_export_fields` special commands, the fields are processed according to the rules described in the following sections.

Ignored Fields

For Mercury Demand Management, the non-updateable request fields detailed in *Table 4-2* will not be imported or are handled specially.

Table 4-2. Ignored request fields

Field	Description
REQ.CREATED_BY (and derivatives such as CREATED_BY_USERNAME and so forth)	Any value for this field in an incoming SOAP request is ignored. When a request is updated as a result of a Web service call, the last-updated-by user is set to the user defined by the WEB_SERVICES_USER server parameter.
REQ.CREATION_DATE	Any value for this field in an incoming SOAP request is ignored. It is set by the system during request creation, in the same manner as when a request is created through the user interface.
REQ.LAST_UPDATED_BY (and derivatives such as LAST_UPDATED_BY_USERNAME and so forth)	Any value for this field in an incoming SOAP request is ignored. When a request is updated as a result of a Web service call, the last-updated-by user is set to the user defined by the WEB_SERVICES_USER server parameter.
REQ.LAST_UPDATE_DATE	Any value for this field in an incoming SOAP request is ignored. It is set by the system during request creation and update, in the same manner as when a request is updated through the user interface.
REQ.PERCENT_COMPLETE	The percent complete of a request cannot be directly influenced by the Web service interface. Request percent complete is controlled by the underlying workflow process, and this field will be ignored if present in Web service request import and subsequent field-update calls. The percent complete can be indirectly influenced by the `ksc_export_status_change` command or the PENDING_STATUS_CHANGE background service, driving workflow transitions which may then change the value of the request percent complete field.
REQ.REQUEST_ID REQD.REQUEST_DETAIL_ID	The request ID is uniquely determined by the system in which the request is created, and cannot be specified or altered through the Web services interface.
REQ.REQUEST_ID_LINK, REQ.REQUEST_URL REQ.WORKBENCH_REQUEST_TYPE_URL	These are tokens provided for convenience, and do not correspond to fields on the request. As such, their presence in a SOAP request will be ignored.
REQ.REQUEST_TYPE_ID and REQ.REQUEST_TYPE_NAME	When a SOAP request is imported by the Web services interface, the value of the REQUEST_TYPE_NAME field is used to specify the request type of the new request that will be created as a result of the import. This field is ignored in any subsequent Web service requests to update fields, as changing the request type of an existing request can have significant side effects and should be done in a controlled fashion by a user with appropriate access rights.

Table 4-2. Ignored request fields

Field	Description
REQ.STATUS_ID REQ.STATUS_NAME	The status field of a request cannot be directly influenced by the Web service interface. Request status is controlled by the underlying workflow process, and this field will be ignored if present in Web service request import and subsequent field-update calls. The status can be indirectly influenced by the `ksc_export_status_change` command or the PENDING_STATUS_CHANGE background service, driving workflow transitions which may then change the value of the request status field.
REQ.SUBMIT_DATE	Any value for this field in an incoming SOAP request is ignored. It is set by the system when the request is submitted.
REQ.WORKFLOW_ID and REQ.WORKFLOW_NAME	This field is ignored in any incoming SOAP requests. Changing the workflow of an existing request can have significant side effects and should be done in a controlled fashion by a user with appropriate access rights.
Notes	When a request is exported, notes are included in the exported SOAP request (if the special command specifies to include them), and will be imported in the resulting remote system. However, notes are currently not supported in subsequent Web service field updates. As such, the following tokens will be ignored if they are present in an incoming “update remote reference” Web service call: • REQ.NOTES • REQ.MOST_RECENT_NOTE_AUTHOR_FULL_NAME • REQ.MOST_RECENT_NOTE_AUTHOR_USERNAME • REQ.MOST_RECENT_NOTE_AUTHORED_DATE • REQ.MOST_RECENT_NOTE_CONTEXT • REQ.MOST_RECENT_NOTE_TEXT • REQ.MOST_RECENT_NOTE_TYPE • REQ.MOST_RECENT_USER_NOTE_AUTHOR_FULL_NAME • REQ.MOST_RECENT_USER_NOTE_AUTHOR_USERNAME • REQ.MOST_RECENT_USER_NOTE_AUTHORED_DATE • REQ.MOST_RECENT_USER_NOTE_CONTEXT • REQ.MOST_RECENT_USER_NOTE_TEXT

Visible and Hidden Parameters

For Mercury Demand Management, the Web services interface handles "visible" values for all supported fields. "Hidden" values (such as the internal keys behind the visible values displayed in the user interface) are not processed by the inbound Web services interface, since they typically represent information applicable only to the local environment, and meaningless in the remote environment.

This is handled automatically during request export, but care must be taken when specifying individual field values using the `ksc_export_fields` or `ksc_export_status_change` commands. If you are using tokens to refer to values that you want to export on the local request, be sure to use the "visible" parameter syntax. For example:

- ```
  ksc_export_fields
  "REQD.VP.FIELD_X=[REQD.VP.FIELD_Y]"
  ksc_end_of_parameters
  ```

  This is valid. The visible value of local field FIELD_Y will be sent and will be populated into FIELD_X on the remote request (if the value is valid in the remote environment).

- ```
  ksc_export_fields
  "REQD.P.FIELD_X=[REQD.P.FIELD_Y]"
  ksc_end_of_parameters
  ```

  This may not be valid. The hidden value of local field FIELD_Y will be sent and this most likely will not be a valid value for FIELD_X on the remote system. However, there may be cases where you do explicitly want to send the hidden value of a field to the remote system. If this is required, it is important to remember that the value will be validated as a visible value on FIELD_X in the remote system. For example, if FIELD_X is an auto-complete, then the value will be entered as cull text in the standard auto-complete processing, as if the user had typed that value directly into the auto-complete field in the user interface.

Multi-value Auto-complete Fields

For Mercury Demand Management, auto-complete fields with multi-select capability are fully supported by the `ksc_export_request` command, and will be correctly validated by the request import process in the remote system. However, multiple values cannot be sent with individual token-value pairs in the `ksc_export_fields` and `ksc_export_status_change` commands. In these cases, all values are processed as single values.

Processing Order

For Mercury Demand Management, all incoming SOAP fields that are found to have a corresponding field defined on the request (onto which they are being applied) are then processed to apply the new value. The fields are ordered according to their position in the user interface layout (left-to-right, top-to-bottom) and processing occurs in this order. This provides some level of predictability in field processing, thereby facilitating field-change rule processing and other potential dependencies.

Field-Change Rules

For Mercury Demand Management, rules can be defined on request types and in table component validations to specify dependencies between fields in the respective request or table. These rules are triggered when a field value changes. In the Web services interface, when a field value is updated, any existing field-change rules are run. The result should be the same as if a user in the user interface manually entered the new field value.

Field Validation

Each time a new value is applied into a field, the value is validated to ensure that it is allowed in that field. If a value is found to be invalid, then an error is generated. Validation is attempted on all incoming SOAP fields (with valid tokens) and errors are collected. The transaction fails if, after attempting all field validations, there exists one or more errors. All errors collected during validation are summarized in the corresponding exception that is then returned to the Mercury IT Governance Center instance that made the Web service call.

Each type of field component has different validation logic and some are not supported, as shown in *Table 4-3*.

.

Table 4-3. Validation logic for fields

Type	Validation Logic
Attachment	Not supported. Field is ignored and no error is generated.
Auto Complete List	If the incoming value is just a single string value, then the value is validated according to the following logic: • If the value is empty (NULL or “”), then apply an empty value to the request field and don’t do any more work. • If the value is not empty, then run the auto-complete using the incoming value as the cull text. • If there are no matches, an error is generated • If there is one match, then apply the matching value to the request field. • If there are many matches, then search for an exact match. If an exact match is found, then apply that value to the request field. If no exact match is found, an error is generated. If the incoming value is a multiple-value SOAP field with more than one value, then the request auto-complete field is tested to see whether it supports multiple values. If not, an error is generated. If so, then each single value is validated according to the following logic: • If the value is empty (NULL or “”), then an error is generated. • If the value is not empty, then run the auto-complete using the incoming value as the cull text. • If there are no matches, an error is generated. • If there is one match, then apply the matching value to the request field. • If there are many matches, then search for an exact match. If an exact match is found, then apply that value to the request field. If no exact match is found, an error is generated. If all values are validated with no errors, then they are concatenated back together and applied to the request field.
Budget	Not supported. Field is ignored and no error is generated.
Date Field	Incoming value is tested to ensure that it is a valid date. If the value is not a valid date, an error is generated.

Table 4-3. Validation logic for fields

Type	Validation Logic
Directory Chooser	Incoming value is set on the request field with no further validation.
Drop Down List	Incoming value is tested to ensure that it exists within the available options in the drop down list. If the value is not present in the list, an error is generated.
File Chooser	Incoming value is set on the request field with no further validation.
Financial Benefit	Not supported. Field is ignored and no error is generated.
Password Field	Not supported. Field is ignored and no error is generated.
Radio Buttons (Yes / No)	If the incoming value is “Y” or “Yes” (or internationalized equivalent), then the “Yes” option is chosen. Otherwise the “No” option is chosen.
Resource Pool	Not supported. Field is ignored and no error is generated.
Staffing Profile	Not supported. Field is ignored and no error is generated.
Table Component	If the incoming value is not a SOAP table field, an error is generated. Otherwise, each cell value within the SOAP table field is processed as an individual field within the table component on the request, following the same validation logic as described in this table.
Text Area	Incoming value is set on the request field with no further validation.
Text Field	Incoming value is set on the request field with no further validation.
Web Address (URL)	Incoming value is set on the request field with no further validation.

Error Handling

If a Web service call succeeds, no additional information is returned in the response If it fails, an error is returned. Errors generated by a Web service call will contain a code and corresponding message. The error codes are shown in *Table 4-4*.

Table 4-4. Error codes

Error Code	Description
0	An unexpected error.
1	Error in initializing the Web service receiver objects (such as deriving a request type ID in the subordinate and so forth).
2	Error during field validation. This is expected to be the most common error to occur when importing a SOAP Mercury IT Governance Center Request since specific values on the incoming fields may not match with valid values in corresponding fields on the receiving system. To facilitate debugging, all field validations are attempted, and if any fail, their error messages are “collected” and sent back together in one response to the caller.
3	Error during handling of references.
4	Error during request save.
5	Error indicating that the Web service API is disabled, so the requested operation could not be attempted.
6	Error in the Mercury IT Governance Center Web service framework, indicating that the Web service call in question has already been processed, and will not be processed again. Each Web service call has a unique identifier property, and these identifiers are tracked on the receiving system. If a call is received with the same identifier as a previous call, it is rejected.

Chapter 5 Examples

In This Chapter:

Scenario: Defect Resolution

Description

The research and development (R&D) team develops and implements corrections to programming errors (defects). Once corrective action is completed, the defect is tested by the quality assurance (QA) team and R&D notified if the test passed or failed.

The R&D process is tracked on a Mercury IT Governance Center system whereas the QA process is monitored elsewhere. Web services are used by the R&D team to request defect validation and obtain the status of the testing.

It is important to keep in mind that Web services interactions can also take place between multiple Mercury IT Governance instances or, if Web services are provisioned, between mutiple third-party systems, none of which are Mercury IT Governance Center instances.

Figure 5-1 shows the following Web services interactions between the two systems.

- **Software defect.** The supervisor workflow (QA handoff) creates a remote request to the subordinate for testing.

  The supervisor task awaits the results of the testing.

  The remote workflow (Start QA testing) is initiated on the subordinate.

- **Test status.** Once the testing is complete (Test complete), a status update is sent from the subordinate to the supervisor (Wait for QA).

  The supervisor task now takes the appropriate path based on the status of the testing.

  The subordinate workflow is completed and exited.

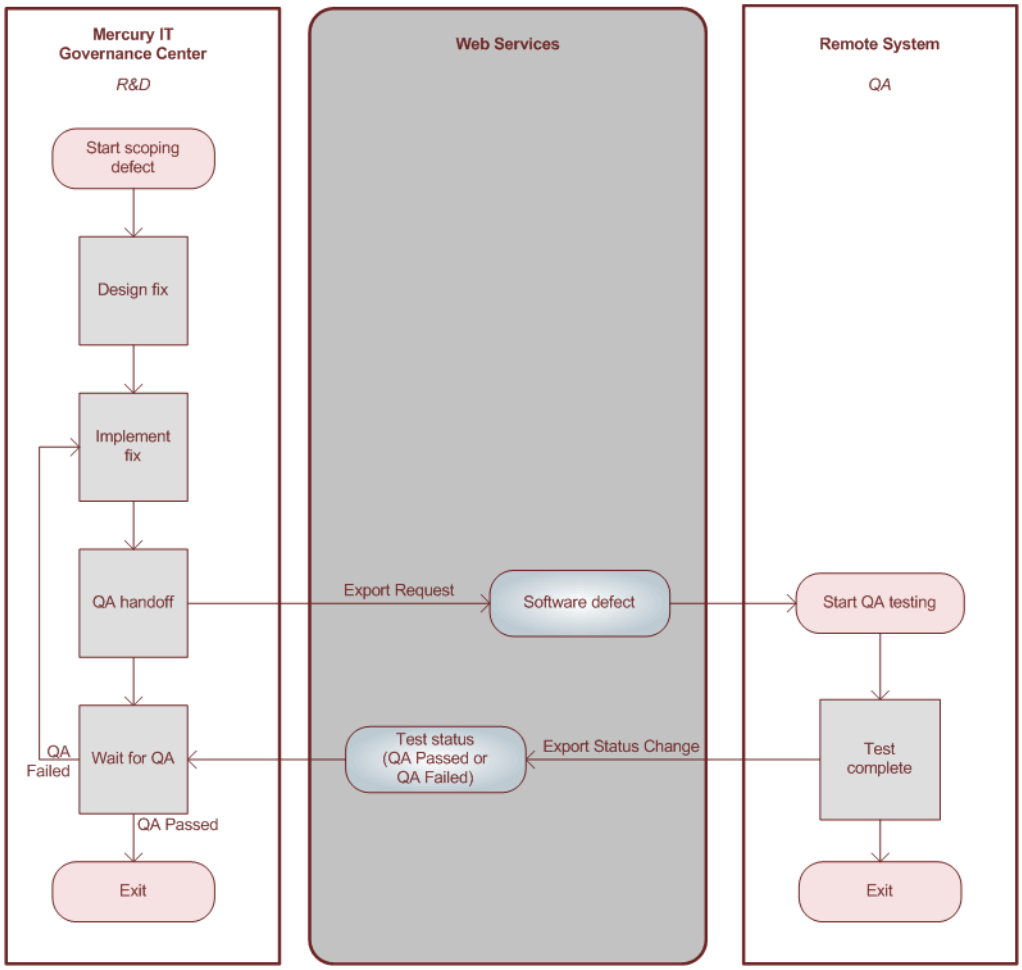

Figure 5-1. Sample scenario

Workflows

The Mercury IT Governance Center workflow is shown in *Figure 5-2*.

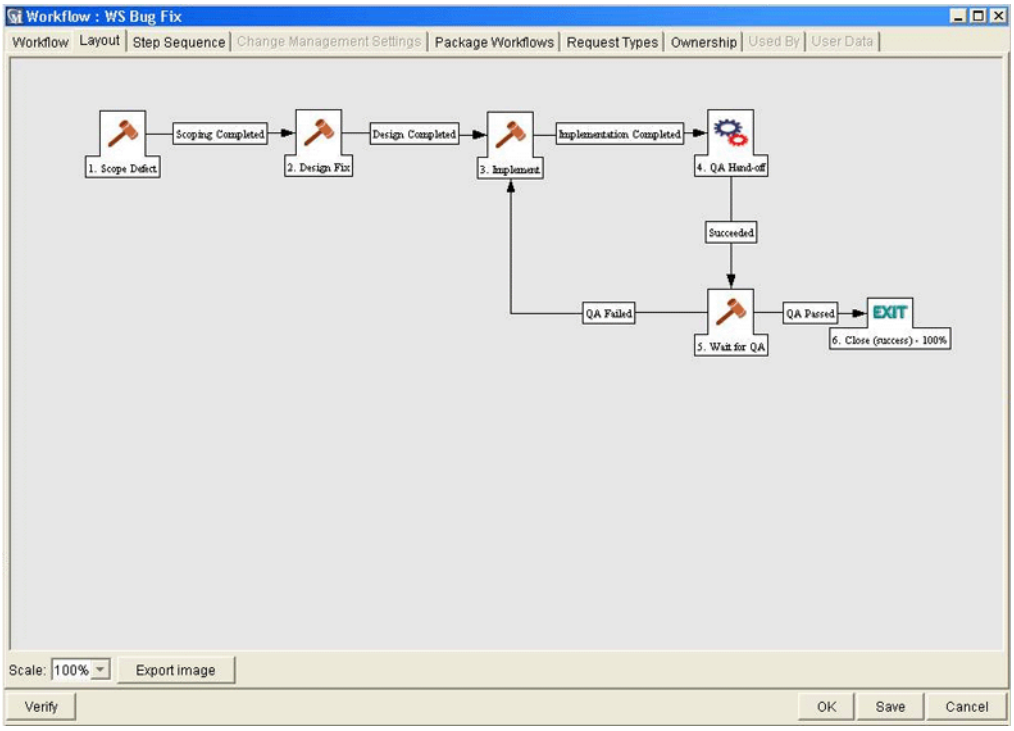

Figure 5-2. Mercury IT Governance Center workflow

Web Services

Export Request

For this scenario, the following special command is used by the supervisor to request testing of the corrective action. *Figure 5-3* shows the command from the Workbench.

```
ksc_export_request "http://qa/itg" "Software Defect" true true
true true
```

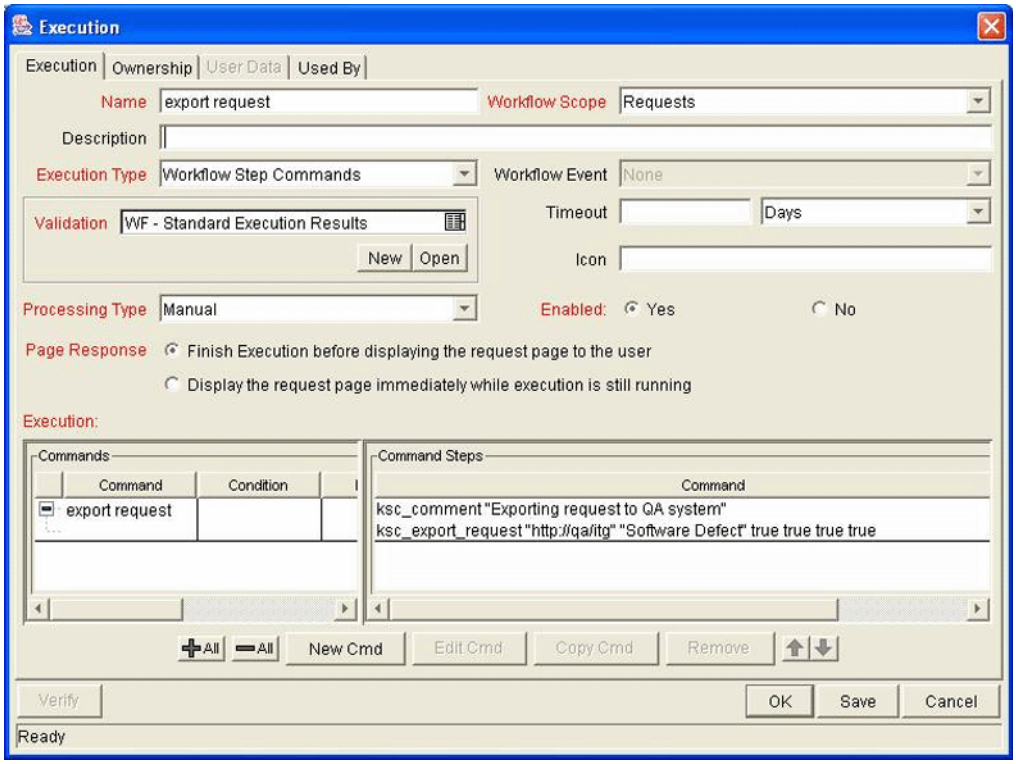

Figure 5-3. Export request

Export Request Status Change

For this scenario, a special command similar to the following is used by the subordinate (if the application is hosted on a Mercury IT Governance Center instance) to update the status of the request. *Figure 5-4* shows the command from the Workbench.

```
ksc_export_status_change "QA Passed"
REQD.VP.COMPLETION_DATE="[REQD.VP.COMPLETION_DATE]"
REQD.VP.COMPLETED_BY="[REQD.VP.COMPLETED_BY]"
ksc_end_parameters
```

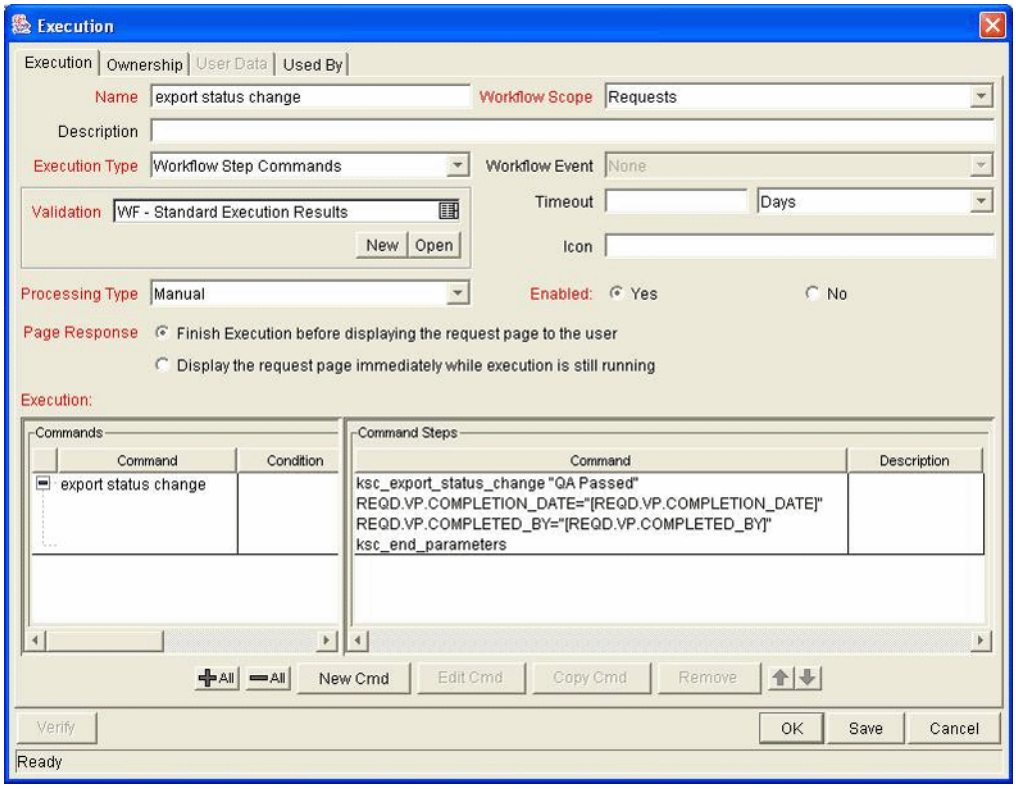

Figure 5-4. Export request status change

Export Request Fields

A special command similar to the following is used by the subordinate (if the application were hosted on a Mercury IT Governance Center instance) to update a field value for the request. *Figure 5-5* shows the commands from the Workbench.

```
ksc_export_fields
REQD.VP.COMPLETION_DATE="[REQD.VP.COMPLETION_DATE]"
REQD.VP.COMPLETED_BY="[REQD.VP.COMPLETED_BY]"
ksc_end_parameters
```

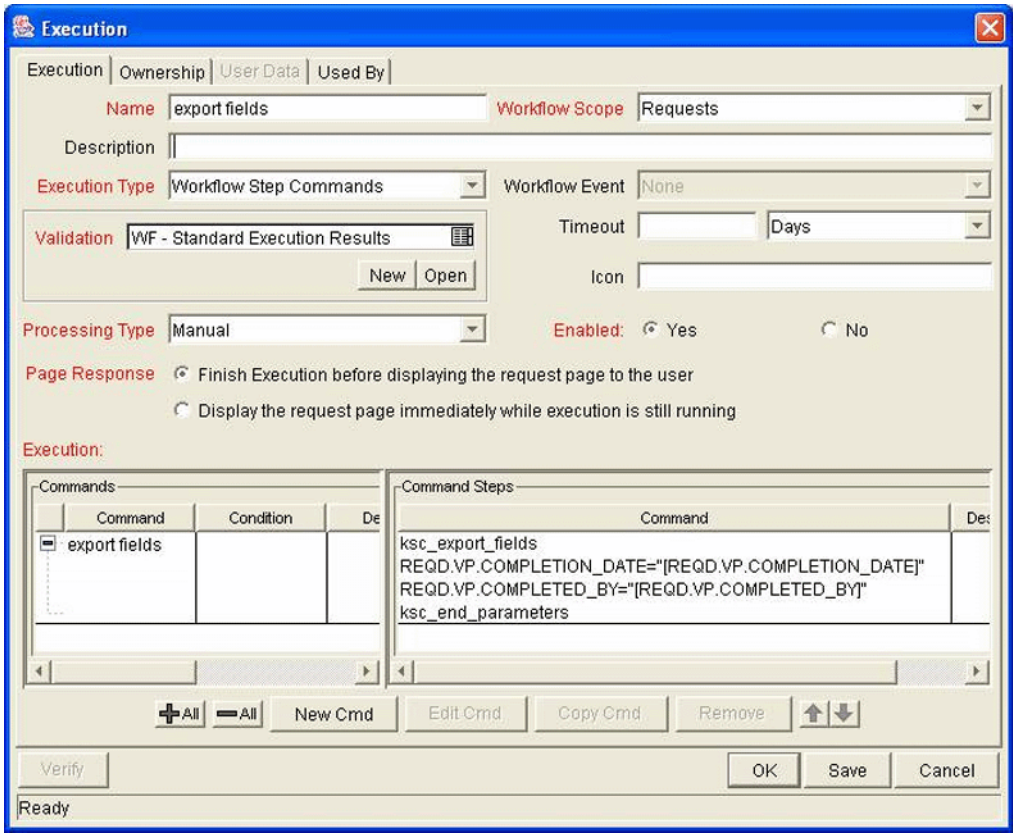

Figure 5-5. Export request fields

Chapter 6 Reference

In This Chapter:

Web Services Calls

Web Service Definition Language Specification

The Web Service Definition Language (WSDL) specification used in conjunction with the Mercury IT Governance Server can be downloaded from your Mercury IT Governance Server. Individual WSDL files exist for each Mercury product and can be found at the following URLs:

- Mercury Demand Management

  `<base_url>/services/Request?wsdl`

- Mercury Time Management

  `<base_url>/services/TM?wsdl`

- Mercury Financial Management

  `<base_url>/services/Finance?wsdl`

where the `<base_url>` placeholder represents the base URL for Web access to Mercury IT Governance Center.

For clarity in this document, Mercury utilizes Java notation to describe service call syntax. Please refer to the WSDL specification for an accurate description of the data types and schema used with Mercury IT Governance Center.

All Mercury IT Governance Center Web services may return a SOAP fault to the caller. If a valid SOAP request is received by a Mercury IT Governance Server and an error occurs while processing the request, a ProcessingError object is returned and contains a list of error messages.

Mercury Demand Management

deleteRequests

Description

`deleteRequests` allows a remote system to delete one or more requests on Mercury IT Governance Center. Each of the specified request IDs will be deleted, if it exists on the Mercury IT Governance Center instance. Requests that do not exist on the instance are ignored and no error is generated.

The number of requests that were deleted is returned.

Syntax

```
int deleteRequests (RequestHeader <header>, String[],
<requestIDs>)
```

Parameters

Parameter	Description
header	Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center.
requestIDs	Specifies the array of request IDs to be deleted.

getRequestTypesByFieldGroup

Description

`getRequestTypesByFieldGroup` allows a remote system to call Mercury IT Governance Center and retrieve a list of request types that contain the specified field group.

Syntax

```
String[] getRequestTypesByFieldGroup (RequestHedader <header>,
String <fieldGroup>)
```

Parameters

Parameter	Description
header	Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center.
fieldGroup	Specifies the field group to be used as a filter for the request types. If no field group is specified, all request types on the Mercury IT Governance Center instance are returned.

getRequestTypeFields

Description

`getRequestTypeFields` allows a remote system to call Mercury IT Governance Center and retrieve the description of all the fields in the specified request type.

An array of `FieldMetaData` objects are returned, each describing a single field in the specified request type.

Syntax

```
FieldMetaData[] getRequestTypeFields (RequestHeader <header>,
String <requestType>)
```

Parameters

Parameter	Description
header	Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center.
requestType	Specifies a request type name (must be valid). An error will be generated if no request type is provided.

importRequest

Description

`importRequest` allows a remote system to call Mercury IT Governance Center and instructs it to create a request within Mercury Demand Management.

The `request` parameter must include the `REQ.REQUEST_TYPE_NAME` field. This field will determine the type of the new request.

The new request will use the default workflow.

The service returns a Response object that contains a single RemoteReference pointing to the new request.

Refer to Chapter 4, *Understanding the Web Services Implementation,* on page 27 for a discussion of how fields and validation are treated when importing requests into Mercury IT Governance Center.

Syntax

```
Response importRequest (RequestHeader <header>, Request
<request>)
```

Parameters

Parameter	Description
header	Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center.
request	Provides the data that will be used when creating the request.

updateRemoteReference

Description

`updateRemoteReference` allows a subordinate to notify Mercury IT Governance Center of a change in status for a particular remote reference.

These updates can result in the following actions:

- If data is provided, the corresponding fields in the request will be updated with the new values. Keep in mind that transactions must be unique and that the receiver and source parameters are crucial.
- If the workflow is at a step that has an outbound transition that matches the new status, the workflow transition will be followed.
- The "status" display of the remote reference on the Request page in Demand Management will be updated.
- Returns an integer value which is always set to 0.

Syntax

```
int updateRemoteReference (RequestHeader <header>, Identifier
<receiver>, Identifier <source>, String <status>, Field[]
<field>)
```

Parameters

Parameter	Description
header	Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center.
receiver	Identifies the Mercury IT Governance Center instance where the status is to be updated.
source	Identifies the external system providing the update.
status	Specifies the new status.
field	Specifies the new data.

Mercury Time Management

read

Description

`read` allows a remote system to call Mercury IT Governance Center and retrieve the time data for the specified period.

A `ReadResponse` is returned and contains the header, a response message, and the data.

Syntax

`ReadResponse read (ReadMessage <readMessage>)`

Parameters

Parameter	Description
readMessage	The header and time filter as follows: **Header:** Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center. **Time filter:** Specifies the TMFilters object. Valid filters include: • Work item name • Work item type • Work item ID • Resource name • Start date • End date • Time sheet status Required filters include: • Start date • End date Time sheet statuses are an array.

Parameter	Description
readMessage (continued)	For work item names, specify the master project name, or meaning for Misc only. For work item ID, specify the master project ID, request ID, package ID, or lookup code for Misc only.

Mercury Financial Management

create

Description

`create` allows a remote system to call Mercury IT Governance Center and create a new budget. If a budget with the specified name already exists, an error is generated.

A `CreateResponse` is returned and contains the header, a response message, and the budget ID.

Syntax

`CreateResponse create (CreateMessage <createMessage>)`

Parameters

Parameter	Description
createMessage	The header and budget data as follows: **Header:** Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center. **Budget data:** Specifies the Budget object. (This is the same as the object returned by the read Web service.) The following budget fields are required: • Budget name • Start date • End date • Region • Period type (only Fiscal Month is currently supported)

read

Description

`read` allows a remote system to call Mercury IT Governance Center and retrieve the budget data.

A `ReadResponse` is returned and contains the header, a response message, and the data.

Syntax

`ReadResponse read (ReadMessage <readMessage>)`

Parameters

Parameter	Description
readMessage	The header and budget filter as follows: **Header:** Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center. **Budget filter:** Specifies the BudgetFilters object. Valid filters include: • Budget name • Budget ID • Budget status • Active • Associated with type • Associated with name • Budget user data There are no required filters. Budget statuses are an array. Budget user data are token-value pairs. For details on token-value pairs, see *Commands, Tokens, and Validations Guide and Reference*.

update

Description

`update` allows a remote system to call Mercury IT Governance Center and modify an existing budget. If a budget with the specified name does not exist, an error is generated.

An `UpdateMessage` is returned and contains the header, a response message, and the budget ID.

Syntax

`UpdateMessage update (UpdateMessage <updateMessage>)`

Parameters

Parameter	Description
updateMessage	The header and budget data as follows: **Header:** Provides the Request Header object with a unique transaction name, as well as username and password to access Mercury IT Governance Center. **Budget data:** Specifies the Budget object. (This is the same as the object returned by the read Web service.) The following budget fields cannot be updated: • Budget name • Budget ID • Associated with type • Associated with name • Start period name • Start period start date • End period name • End period start date • Period type • Region • Base currency

Mercury IT Governance Center Special Commands

Selected Web services calls have been incorporated in Mercury IT Governance Center as special commands. For details on token-value pairs, see *Commands, Tokens, and Validations Guide and Reference*.

Mercury Demand Management

ksc_export_fields

Description

`ksc_export_fields` causes the Mercury IT Governance Center workflow engine to invoke the *updateRemoteReference* Web service and update the data of the request on all the remote references associated with the request.

The Command Builder in the Workbench will show `ksc_export_fields` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

```
ksc_export_fields
<token1>="<value1>"
<token2>="<value2>"
...
<tokenN>="<valueN>"
ksc_end_of_parameters
```

Parameters

Name	Type	Description
token	String	Specifies the fully qualified token.
value	String	Specifies the new data value.

You cannot export multiple selection fields or tables using `ksc_export_fields`.

Do not include square brackets around the token.

ksc_export_request

Description

`ksc_export_request` causes the Mercury IT Governance Center workflow engine to invoke the *importRequest* Web service.

The Command Builder in the Workbench will show `ksc_export_request` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

`ksc_export_request <remoteServerURL>, <remoteRequestType>, <exportNotes>, <exportReferences>, <exportRemoteReference>, <createLocalReference>, [<XSLFile>]`

Parameters

Name	Type	Description
remoteServerURL	String	Provides the base URL of the "import request" server that will be called by this special command. For example, `http://<hostname>:<port>/itg/`
remoteRequestType	String	Specifies the name of a valid request type in the subordinate. This is particularly useful for integrations between Mercury IT Governance Center instances.
exportNotes	Boolean	Indicates whether you wish to include the current request's notes in the export.

Name	Type	Description
exportReferences	Boolean	Indicates whether you wish to include the current request references in the export. References are exported as URL links.
exportRemoteReference	Boolean	Indicates whether you wish to include a remote references back to the source request. If TRUE, then the remote server that is importing the request will create a remote reference back to the source request.
createLocalReference	Boolean	Indicates whether you wish to create a remote reference on the source request to the new request that was imported into the subordinate.
XSLFile	String	Optional. Specifies the file containing an XSL transformation to be performed on the SOAP message before sending.

ksc_export_status_change

Description

`ksc_export_status_change` causes the Mercury IT Governance Center Workflow Engine to invoke the *updateRemoteReference* Web service and update the current status of the request on all the remote references associated with the request, as well as update the specified data.

The Command Builder in the Workbench will show `ksc_export_status_change` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

```
ksc_export_status_change <newStatus>
<token1>="<value1>"
<token2>="<value2>"
...
<tokenN>="<valueN>"
ksc_end_of_parameters
```

Parameters

Name	Type	Description
newStatus	String	Specifies the new status.
token	String	Specifies the fully qualified token.
value	String	Specifies the new data value.

You cannot export multiple selection fields or tables using `ksc_export_status_change`.

Do not include square brackets around the token.

Mercury Time Management

ksc_export_actual_time

Description

`ksc_export_actual_time` allows Mercury IT Governance Center to update the time data for the specified period on the remote system with the actual data from the Mercury IT Governance Center instance.

The Command Builder in the Workbench will show `ksc_export_actual_time` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

`ksc_export_actual_time <remoteServerURL> <workItemType> <workItemID> <startDate> <endDate>`

Parameters

Name	Type	Description
remoteServerURL	String	Provides the base URL of the "update" request that will be called by this special command. For example, `http://<hostname>:<port>/itg/`
workItemType	String	Specifies the type of work item for which the time data should be updated. These include the following: • Projects • Tasks • Requests • Packages • Misc

Name	Type	Description
workItemID	String	Specifies the work item for which the time data should be updated. For requests or packages, specify the corresponding ID. For projects or tasks, specify the master project ID. For Misc, specify the lookup code.
startDate	Date	Specifies a start date for which the time data should be collected. The date format should be: `YYYY:MM:DD HH:MM:SS` where `HH` uses 24-hour notation.
endDate	Date	Specifies the end date for which the time data should be collected. The date format should be: `YYYY:MM:DD HH:MM:SS` where `HH` uses 24-hour notation.

ksc_export_actual_time_as_xml

Description

`ksc_export_actual_time_as_xml` allows Mercury IT Governance Center to create an XML-formatted file containing the time data for the specified period on the remote system with the actual data from the Mercury IT Governance Center instance.

The specified file will be created in the following directory:

`<base_url>/transfers/xml`

where the `<base_url>` placeholder represents the base URL for Web access to Mercury IT Governance Center.

The Command Builder in the Workbench will show `ksc_export_actual_time_as_xml` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

`ksc_export_actual_time_as_xml <filename <workItemType> <workItemID> <startDate> <endDate>`

Parameters

Name	Type	Description
filename	String	Specifies the name of the file that should contain the time data.
workItemType	String	Specifies the type of work item for which the time data should be updated. These include the following: • Projects • Tasks • Requests • Packages • Misc

Name	Type	Description
workItemID	String	Specifies the work item for which the time data should be updated. For requests or packages, specify the corresponding ID. For projects or tasks, specify the master project ID. For Misc, specify the lookup code.
startDate	Date	Specifies a start date for which the time data should be collected. The date format should be: `YYYY:MM:DD HH:MM:SS` where `HH` uses 24-hour notation.
endDate	Date	Specifies the end date for which the time data should be collected. The date format should be: `YYYY:MM:DD HH:MM:SS` where `HH` uses 24-hour notation.

Mercury Financial Management

ksc_export_budget

Description

`ksc_export_budget` allows Mercury IT Governance Center to update the budget data on the remote system with the data from the Mercury IT Governance Center instance.

The Command Builder in the Workbench will show `ksc_export_budget` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

`ksc_export_budget <remoteServerURL> <budgetName>`

Parameters

Name	Type	Description
remoteServerURL	String	Provides the base URL of the "update" request that will be called by this special command. For example, `http://<hostname>:<port>/itg/`
budgetName	String	Specifies the name of the budget to be updated.

ksc_export_budget_as_xml

Description

`ksc_export_budget_as_xml` allows Mercury IT Governance Center to create an XML-formatted file containing the budget data on the remote system with the data from the Mercury IT Governance Center instance.

The specified file will be created in the following directory:

`<base_url>/transfers/xml`

where the `<base_url>` placeholder represents the base URL for Web access to Mercury IT Governance Center.

The Command Builder in the Workbench will show `ksc_export_budget_as_xml` in the list of commands but you will not be able to use the Command Builder to construct complete commands.

You should manually fill in the necessary parameters in the execution step editing window.

Syntax

`ksc_export_budget_as_xml <filename> <budgetName>`

Parameters

Name	Type	Description
filename	String	Specifies the name of the file that should contain the budget data.
budgetName	String	Specifies the name of the budget to be included in the file.

Index

L

P

R

S

T

U

W