HP Service Manager Integration Suite (SMIS)

for the Windows and Unix operating systems

Software Version: 1.10

Developer Guide

Document Release Date: June 2010
Software Release Date: June 2010

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Documentation Updates

The title page of this document contains the following identifying information:

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

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

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

Support

Visit the HP Software Support Online web site at:

www.hp.com/go/hpsoftwaresupport

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

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

1 Overview 7
SMIS SPI 8
Manager SPI 8
Adapter SPI 9
Mapping SPI 10
TaskManager SPI 10
Development procedure 12
Schedule-based integrations 12
UI-based integrations 12
2 Developing an Integration Template 13
Initializing global variables 13
Developing a source adapter and a destination adapter 13
Developing a manager 14
Developing a custom controller 15
Implementing UI controls 17
Registering an integration template in SMIS 19
Register an integration template 19
View or edit a registered integration template 20
Exporting an integration template into an unload file 21
A Processing Logic of the Mapping Functionality 23

1 Overview

Service Manager Integration Suite (SMIS) is a platform that provides centralized management of integration instances, which fall into two categories:

- **Schedule-based**: runs as a schedule in the background.
- **UI-based**: Can be only invoked in the user interface (UI).

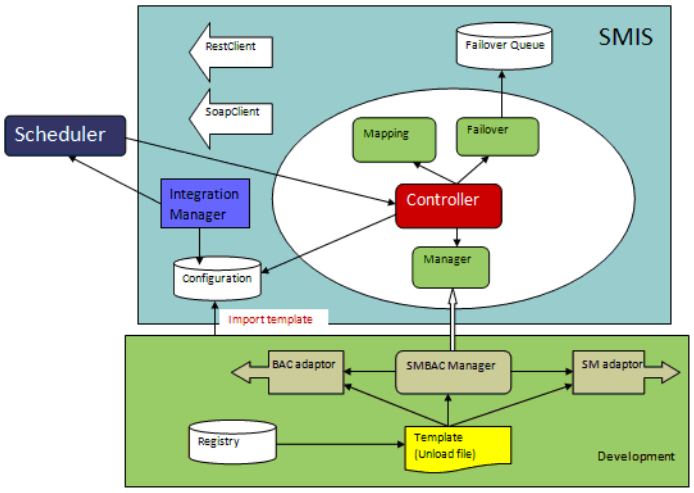
SMIS is also a plug-in-based development platform that:

- Reuses common functions and resolves conflicts across integrations.
- Supports customizing a default controller (for workflow) for integrations.
- Uses JavaScript to develop integration templates.

An integration must be registered as a template so that it can be added to SMIS. To register an integration as a template, the integration must be developed by following the Service Provider Interface (SPI) provided by SMIS.

The functional block diagram of SMIS and a sample integration are shown in Figure 1.

Figure 1 Functional Block Diagram of SMIS and a Sample Integration



SMIS SPI

SMIS provides the following SPI components:

- Manager SPI
- Adapter SPI
- Mapping SPI
- TaskManager SPI

Manager SPI

The following table describes the functions used to develop a manager.

Table 1 Manager SPI

Function	Parameter	Return	Description
appendTasks			Directs the source adapter to retrieve records, converts these records into tasks, and calls the task manager to push these tasks into the task queue.
preProcess	task	Boolean	Prepares the destObj and actions. If it returns false, the task will be ignored and removed.
process	task	Boolean	Returns true or false when the task is or is not processed successfully.
postProcess	task		Performs postprocessing after the task is processed.
getDestObj		Object	SCFile or Object.
getAction		String	Returns a customized action defined by the manager. Can be any value (for example: Insert/ Update/Delete).

Table 1 Manager SPI (cont'd)

Function	Parameter	Return	Description
isScheduleBased		Boolean	Indicates whether the integration is schedule-based or not.
finalize			Performs finalization actions.
initParams			Initializes parameters when the instance is first added. Use `this.configItem.setConfigParameterValue` (<parameter_name>, <value>) to set parameter values.

Adapter SPI

The following table describes the functions used to develop source and destination adapters.

Table 2 Adapter SPI

Function	Parameter	Return	Description
getRecords		Array	Retrieves records from external sources, and returns them to the manager.
sendRecord			Sends out the record according to the action. For example, `sendRecord (data, "delete").`
	Record	Object	JavaScript object that contains the result data to send.
	Action	String	Customized action. For example, "insert", "update", or "delete".
getFields		Array	Returns an array of field names, types, and descriptions. The field information will be used for mapping.

Mapping SPI

The following table describes the functions provided by the mapping function in SMIS.

Table 3 Mapping SPI

Function	Parameter	Return	Description
validate		boolean	Validates the input values.
	inRecord	Object	Input field values.
	direction	String	Mapping direction.
getOutRecord		Object	Gets outRecord by inRecord according to field mapping and value mapping (not including callback).
	inRecord	Object	Input field values.
	direction	String	Mapping direction.
setFieldValues			The final processing step of mapping, which sets the mapped value to destObject.
	inRecord	Object	Input field values.
	outRecord	Object	The object of getOutRecord.
	destObject	Object	The final object of the mapping result.
	smisContext	Object	The container that contains context values.
	direction	String	Mapping direction.

TaskManager SPI

The following table describes the functions provided by the task manager in SMIS.

Table 4 TaskManager SPI

Function	Parameter	Return	description
readTasks		Array	Reads tasks by instance ID.
	intId	String	Instance ID.
removeTask+		Object	Removes the task if it is processed successfully.
	task	Object	Task object.

Table 4 TaskManager SPI (cont'd)

Function	Parameter	Return	description
setFieldValues			The final processing step of mapping, which sets the mapped value to destObject.
updateTask		Object	Updates the task if it is processed unsuccessfully.
	task	Object	Task object.

Development procedure

Developing integrations with the SMIS SPI involves different steps for schedule-based and UI-based integrations.

Schedule-based integrations

To develop a schedule-based integration:

1. Initialize global variables. See Initializing global variables on page 13.
2. Develop an endpoint adapter and a Service Manager adapter. See Developing a source adapter and a destination adapter on page 13.
3. Develop a manager. See Developing a manager on page 14.
4. Develop a custom controller. See Developing a custom controller on page 15.

   SMIS provides a default controller. Determine if the default workflow suits the integration. If not, develop a custom controller.
5. Register the integration as a template. See Registering an Integration Template on page 21.
6. Export the integration template into an unload file. See Exporting an integration template into an unload file on page 21.

UI-based integrations

To develop a UI-based integration:

1. Initialize global variables. See Initializing global variables on page 13.
2. Develop a manager. See Developing a manager on page 14.
3. Implement UI controls. See Implementing UI controls on page 17.
4. Register the integration as a template. See Registering an Integration Template on page 21.
5. Export the integration template into an unload file. See Exporting an integration template into an unload file on page 21.

2 Developing an Integration Template

This chapter describes the tasks to develop an integration template:

- Initializing global variables
- Developing a source adapter and a destination adapter
- Developing a manager
- Developing a custom controller
- Implementing UI controls
- Registering an integration template in SMIS
- Exporting an integration template into an unload file

Initializing global variables

Each integration has global variables. To initialize the global variables of an integration:

1. Add a field to the `info` table, using the integration template name as its name.

   If the integration has global parameters, the field must be a structure field.

2. If the integration has global parameters, add all these global parameter names as fields to the newly added structure.

For example, an integration template named `SMBSM` has two global parameters: `PI` and `BIR`. You need to add a structure named `SMBSM` under the SMIS structure, and then add two fields, `PI` and `BIR`, to the `SMBSM` structure. Note: If the template `SMBSM` has no global parameters, only one field named `SMBSM` needs to be added under the SMIS structure.

Developing a source adapter and a destination adapter

To develop a source adapter, use `readRecords` to return records to the manager.

To develop a destination adapter, use `sendRecord` to process records from the manager.

The `getFields` function defines the fields used in an integration.

See `smis_TestSrcAdapter` and `smis_TestDestAdapter` in the Service Manager script library for examples.

Developing a manager

A manager is required for both schedule-based and UI-based integrations.

To develop a manager:

1. Develop `appendTasks` to prepare tasks in the task queue.
2. Put the logic in either the `preprocess` or `process` method.
3. Do cleaning in the `postProcess` or `finalize` method.
4. Prepare the destination object in the `getDestObj` method for the mapping function to set mapped values.
5. Prepare the action in the `getAction` method to use in the mapping callback(s).
6. Implement `isScheduleBased` to indicate whether the integration is schedule-based or not.

See `smis_TestManager` in the Service Manager script library for an example.

For a UI-based integration, the manager only needs to implement `isScheduleBased` and add the integration specific methods to it. See the following script for an example.

Figure 2 An example manager for a UI-based integration

```
var Class = lib.smis_Prototype.getClass();
var PIManagerClass = Class.create(lib.smis_Manager.getClass(),
{
   getUrl: function(vFile) {
   var baseUrl = this.configItem.getConfigParameterValue("baseurl");
   var url = baseUrl + "&IsmEntityId="+vFile["number"];
   url += "&IsmSubject=";
   var device = new SCFile("device");
   var RC = device.doSelect("logical.name=\"" + vFile["logical.name"] +
   "\"");
   if ( RC == RC_SUCCESS && device["ucmdb.id"] != null){
     url += device["ucmdb.id"];
   } else {
    url += vFile["logical.name"];
   }
    return url;
},
    isScheduleBased: function() {
      return false;
  }
});
   function getClass() {return PIManagerClass;}
```

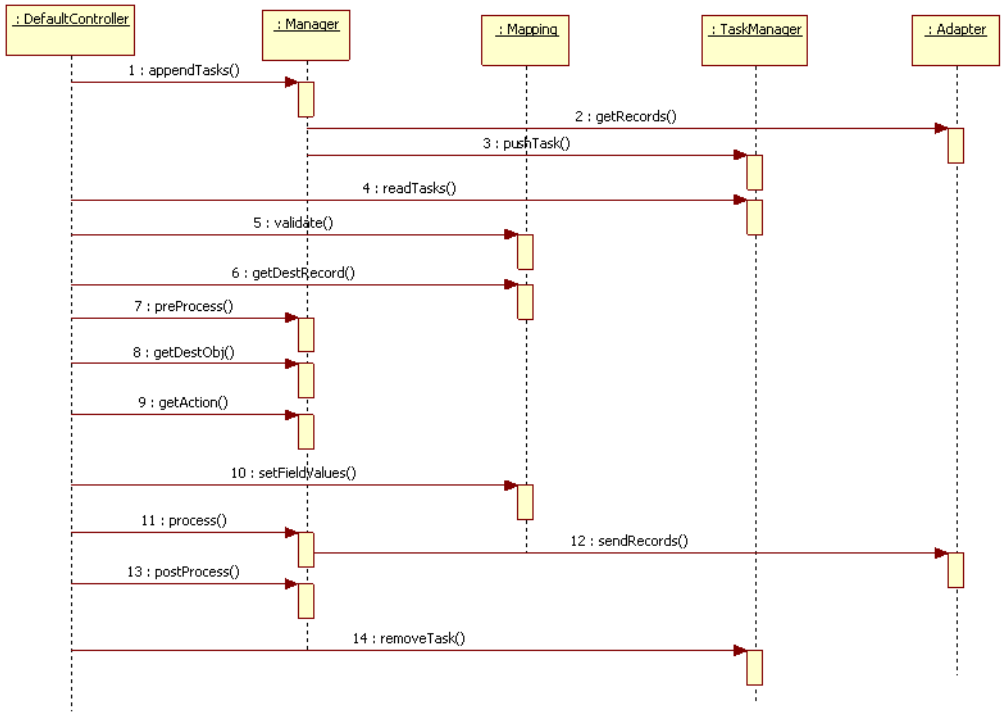
Developing a custom controller

A controller controls interactions between the manager, source and destination adapters, mapping function, and task manager of a schedule-based integration.

UI-based integrations do not need a controller.

SMIS provides a default controller. Figure 3 shows the workflow of the default controller. If this default workflow does not suit your integration, you need to develop a custom controller.

Figure 3 Workflow of the Default Controller



When developing a custom controller, you can refer to `smis_Controller` in the Service Manager script library.

The typical workflow of a controller is as follows:

1. Call the manager to do the following to prepare tasks:
   a. Delegate its source adapter to retrieve data from the endpoint;
   b. Wrap the data as tasks;
   c. Pass the tasks to the taskManager to save to the failover queue.
2. Read all the tasks from the failover queue.
3. Pass the data in the task to the mapping function for validation.
4. If the data passes validation, get the destRecord (which is the result of value mapping) from the mapping function.
5. Call the manager to preprocess and prepare the following:
   - The destObject (which is the target to save or update) ;
   - The action (for example, add/save/delete) that the manager should perform to process the destObject.
6. Call the mapping function to process the destObject. All the final mapped values by the final mapping are set to the destObject.
7. Call the manager to process the final destObject, and to return the result (success or failure).
8. If the result is success, remove the task from the failover queue; If the result is failure, increase the retry count and update it to the failover queue.
9. Call the manager's `postprocess` and `finalize` functions to do cleaning.

Normally, you do not need to create a custom controller. You can leave some processes empty if you do nothing in them. In some cases, you may not need a mapping function and failover queue, so you can create a simple controller.

Implementing UI controls

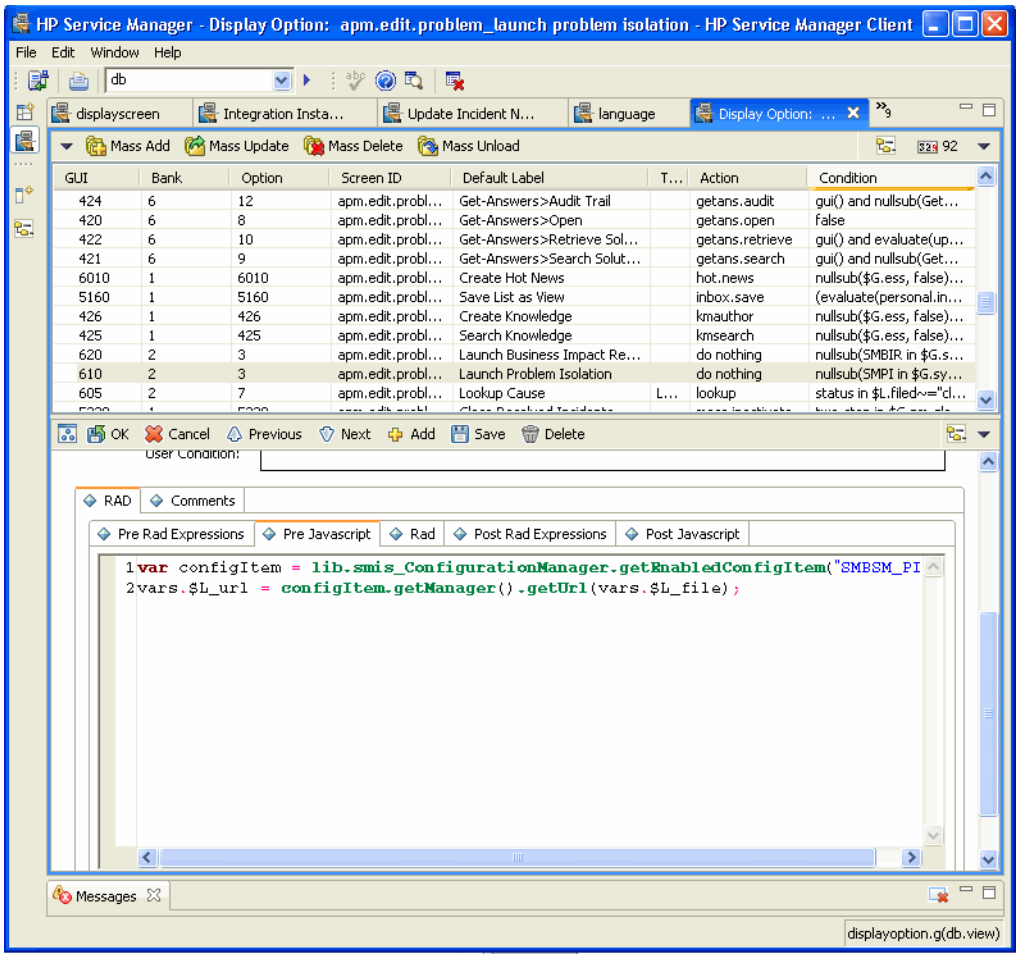
An integration normally comes with new UI controls, such as new menu options or buttons. To control the behavior of these UI controls, the integration needs to interact with SMIS. The following are examples:

- Getting parameter values specified in SMIS;
- Calling some business logic in the integration's manager (for example, to prepare a URL);
- Deciding if a UI element should be visible/enabled by checking the integration's global variables specified in SMIS.

To illustrate new UI controls, consider the Service Manager to Business Availabity Center (BAC) Problem Isolation integration.

This integration includes a new menu option to the Incident form: Launch Problem Isolation. When you select this menu option, a new browser window opens. The URL of this window is based on two things: the value of "baseurl" specified in SMIS and the affected CI field of the incident.

To implement this feature, you can use the Display Options tailoring tool to create a display option record, in which the script on the **Pre Javascript** tab controls the above described behavior. See the following screenshot.



The following examples show you some of the values you can retrieve or set. This is not an exhaustive list:

- To get the enabled instance of the integration:

  ```
  var configItem
  =lib.smis_ConfigurationManager.getEnabledConfigItem(<templateName>);
  ```

  For example: `var configItem =lib.smis_ConfigurationManager.getEnabledConfigItem(SMPI)`

- To get a URL from the manager:

  ```
  vars.$L_url = configItem.getManager().getUrl(vars.$L_file);
  ```

- To get a parameter value:

  ```
  var paramValue = configItem.getConfParamaterValue(paramName)
  ```

- To set the property of a UI element:

  Each integration has global variables, which can be used in RAD expressions like `SMBSM in vars.$G.system.info`. If an integration has global parameters, they can be used the same way, for example: `PI in vars.$G.system.info`. You can use these global variables to set the property of a UI element (for example, to set a menu item to be visible/invisible, or to set a button to be enabled/disabled).

Registering an integration template in SMIS

You need to register an integration as a template in SMIS before it is available in the template list in SMIS. You can view or edit an integration template after it is registered in SMIS.

Register an integration template

To register an integration as a template:

1 Register the general information of the integration template.

a Log on to Service Manager as a system administrator.

b From **Database Manager,** open the `SMISRegistry.g` form.

c On the **General** tab, enter the following information of the integration:

— **Name**: Name of the integration template.

— **Version**: Version of the integration template.

— **Manager Class Name**: Script name of the manager of the integration.

— **Controller Class Name**: Script name of the controller of the integration. If this field is left blank, a default controller is used.

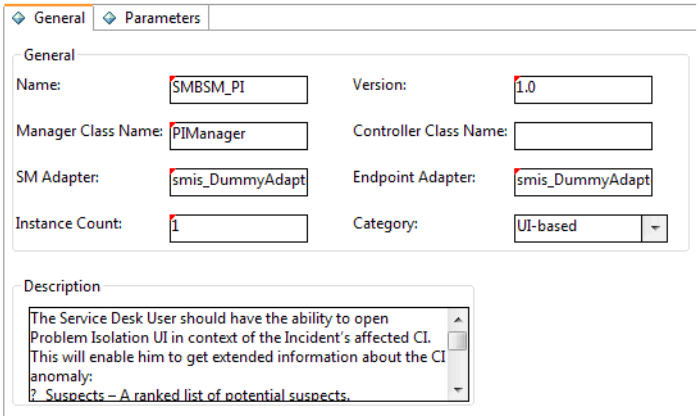
— **SM Adapter**: Name of the Service Manager adapter.

— **Endpoint Adapter**: Name of the endpoint adapter.

— **Instance Count**: Maximum allowed number of instances of the integration.

— **Category**: Category of the integration (Schedule-based or UI-based).

See the following screenshot for an example.

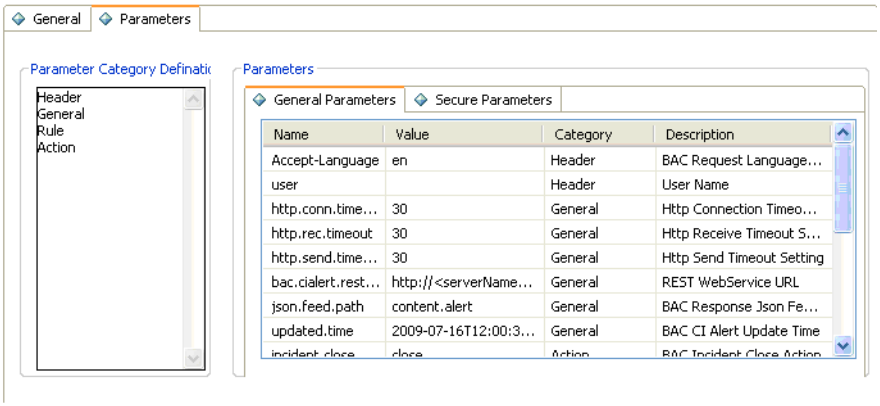


2 Enter parameters required for the integration.

a Select the **Parameters** tab.

b If necessary, enter parameter categories for the integration.

c On the **General Parameters** and **Secure Parameters** tabs, enter parameters of the integration.



Newly added parameter categories are not available in the `Category` list until the form is saved and then reopened.

There is a default `Global` category. SMIS will initialize all parameters of this category to global variables, which can be used in Service Manager. These parameters can only be set to true or false.

3 Edit the out-of-box mappings.

a On the **General** tab, click the **Go to Configure Field Mapping** link.

The Registry Field Mapping page opens.

b On the **Field Mapping** tab, add or edit field mappings.

c On the **Field Mapping** tab, click **Edit Callback** to edit callbacks, or click **Clear Callback** to delete callbacks.

d On the **Value Mapping** tab, add or edit value mappings.

e Click **Finish**.

The default fields in the drop-down list are defined in the `getFields` method of the Service Manager adapter and endpoint adapter.

Each mapping value can be calculated by combining the direct mapping value, default value, value mapping table, and callback value.

For information about the processing logic of mapping, see Appendix A, Processing Logic of the Mapping Functionality.

View or edit a registered integration template

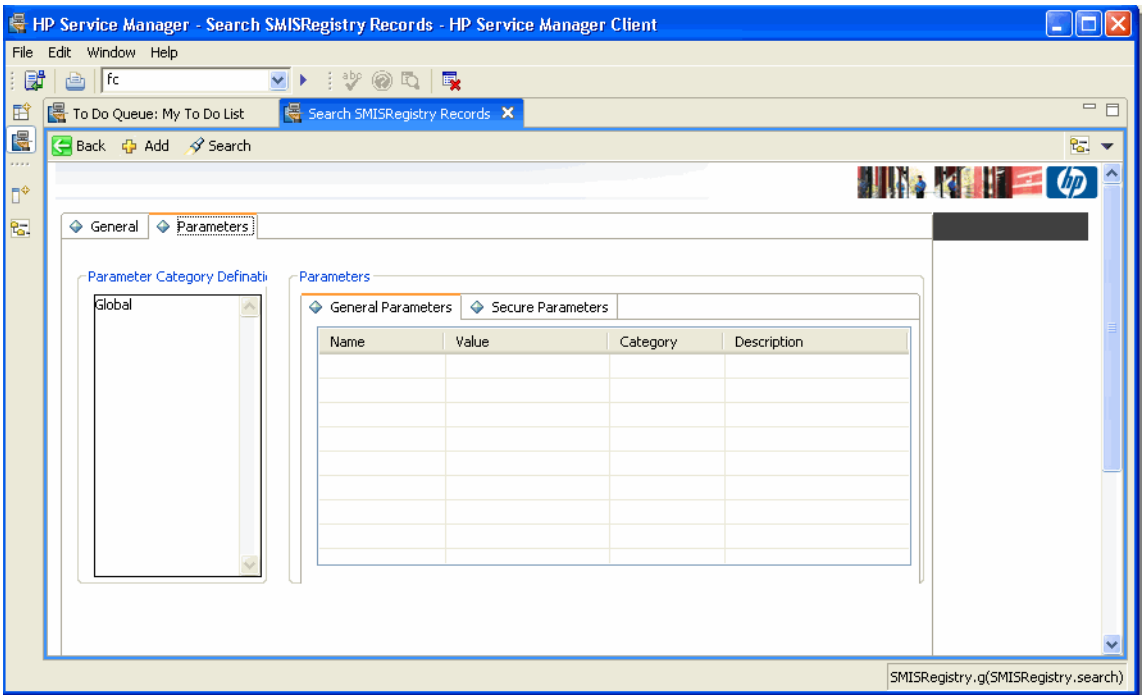
Once you have registered an integration template in SMIS, a record is created in the `SMISRegistry.g` form. You can view or edit the integration template.

To view or edit a registered integration template:

1 Log on to Service Manager as a system administrator.

2 From **Database Manager,** open the `SMISRegistry.g` form.

3 Select the **Parameters** tab, and remove the **Global** category from the Parameter Category Definition pane.



> The **Global** category is a default category, which displays on the **Parameters** tab by default. If you do not clear it before performing a search, the search returns only those records with global parameters.

4 Click **Search**. A list of integration templates displays.

5 Select a record to view or edit the details. See Register an integration template.

Exporting an integration template into an unload file

After you have registered an integration template in SMIS, you need to export it into an unload file, which you can then import into a testing system or production system.

To export an integration template into an unload file:

1 Create an unload script for the integration template. This unload script must include all the changes you made to the system when developing the integration template.

2 Go to **Tailoring** → **Unload Script Utility**. Select the unload script you created and export it into an unload file.

For more information, see the Service Manager help.

A Processing Logic of the Mapping Functionality

The following diagram illustrates the processing logic of the mapping functionality.