Oracle® Communications Services Gatekeeper

Portal Developer's Guide

Release 6.1

E64629-02

November 2016

ORACLE®

Oracle Communications Services Gatekeeper Portal Developer's Guide, Release 6.1

E64629-02

Contents

Preface vii

Audience vii
Documentation Accessibility vii
Related Documents viii

1 About Extending and Replacing Portal Graphical User Interfaces

Integrating the PRM Portals with External Systems 1-1
Securing the PRM API 1-1
Securing PRM Web Services 1-1
Deployment Example 1-2
Managing APIs, Partners, and Network Service Suppliers Using the Services Gatekeeper Portals 1-3
About the API and Partner Manager Portal 1-3
About the Network Service Supplier Portal 1-3
About the Partner Portal 1-3
Extending Portals 1-4
About the Default Modules and Pages 1-4
Points to Consider When Extending the Portals 1-5
About the Naming Conventions for Help Files 1-5
About the Naming Conventions for Help IDs 1-6
About the Extension Example 1-6
Adding a Page to a Portal Module 1-6
Deploying the New Page 1-7
Providing the Help Content for the New Page 1-7
Updating the Help WAR File for the Portal Application 1-8
Readying the Help WAR File for Redeployment 1-10
Redeploying the WAR File in Your WebLogic Server Domain 1-10
Adding a Custom Module to a Portal Application 1-11
Adding Custom Modules 1-11
Adding a Page to the Portal Application 1-11
Verifying the Addition in the Portal Application 1-12

2 Understanding the API Management Model

Understanding the Services Gatekeeper Management Model 2-1
Account States 2-2

Administering PRM Administrative Users 2-3

3 Common Partner Relationship Management Use Scenarios

Registering a new Service Provider Account 3-1
Registering a new Application Account 3-2
Registering a new Application Instance 3-2
Operator: Creating a Service Provider Group 3-3
Operator: Creating an Application Account Group 3-3
Service Provider Requests an Account Update 3-3
Service Provider Deactivates an Account 3-3
Service Provider Requests an Account Deletion 3-4
Communicating General Information Between Service Provider and Operator 3-4
Retrieving Charging Data Records 3-4
Retrieving Statistics 3-5
Retrieving Alarms 3-5

4 Creating Custom Actions for Your APIs

Understanding When to Create Custom Actions 4-1
Creating and Adding a Custom Action to Services Gatekeeper 4-1
Managing Custom Actions 4-4
Discovering Actions Chains Schemas 4-5
Example Custom Actions 4-5
RESTful Request for an Attribute Example 4-5
Header Validation Example 4-6
Example Black List Action 4-7
Example Action for Adding a New Key/Value to EDRs 4-8

5 Old Partner Relationship Management Interfaces

WSDLs 5-2
Service Provider Interfaces 5-3
Service Provider Service Interfaces 5-3
Management User 5-3
Service Provider Accounts 5-3
Application Account 5-4
Application Instances 5-4
Service Provider CDR Utility Interface 5-4
Service Provider Statistics Utility Interface 5-4
Service Provider Login Interface 5-5
Operator Interfaces 5-5
Operator Service Interfaces 5-5
Management User 5-5
Service Provider Account 5-5
Service Provider Group 5-6
Application Account 5-6
Application Account Group 5-6
Application Instance 5-7

Operator Alarm Utility Interface 5-7
Operator CDR Utility Interface 5-7
Operator Statistics Utility Interface 5-7
Operator Login Interface 5-8

A API Management REST-based API

Understanding the Partner Relationship Management API A-1
Creating and Updating Objects A-1
Understanding API Management API Error Handling A-2
API Management API Operations A-2
API and Partner Manager Portal Operations A-2
API Management Operations A-2
Group Management Operations A-2
Partner Management Operations A-3
Application Management Operations A-3
System Configuration Operations A-3
Partner Portal Operations A-3
Account Management Operations A-3
Application Management Operations A-3
Network Service Supplier Portal Operations A-4
Network Interface Management Operations A-4
Understanding the API Management API Objects A-4

B Actions Management REST-Based API

Understanding the Actions Management API B-1
Understanding Actions Management API Error Handling B-1
Actions Management Operations B-1
submitActionChain B-1
Authorization B-1
HTTP Method B-1
URI B-1
Request Body B-2
Sample requestAction Schemas B-2
Response Body B-3
Example Request B-3
Example Response B-4
retrieveActionChain B-4
Authorization B-4
HTTP Method B-4
URI B-4
Request Body B-4
Response Body B-4
Example Request B-4
Example Response B-4
loadActionSchemas B-5
Authorization B-5

HTTP Method B-5
URI B-5
Request Body B-5
Response Body B-5
Example Request B-5
Example Response B-5
verifyAction B-5
Authorization B-6
HTTP Method B-6
URI B-6
Request Body B-6
Example Request B-6
Example Response B-6

C Partner Relationship Management SOAP-based Web Services Interface

About the Interface Functionality C-1
Base Service Error Messages C-1

Preface

This document describes the Oracle Communications Service Gatekeeper's Partner Relationship Management module, and service provider and application provisioning. It includes a detailed description of the Partner Relationship Management interfaces exposed for integrating Services Gatekeeper service provider account management with operator PRM systems, both internal and customer-facing.

This guide also covers the following service provider and application provisioning topics:

- An overview of the Service Level Agreement (SLA) administration model
- Managing service provider groups
- Managing service provider accounts
- Managing application groups
- Managing application accounts
- Managing application instances

Audience

This book is intended for application developers who are interested in building customer relationship management (CRM) or partner relationship management (PRM) applications to manage Service Providers and Applications that are or will use the traffic interfaces exposed by the Services Gatekeeper.

It is also intended for telecom operators who perform service provider and application provisioning tasks.

This guide assumes a familiarity with Services Gatekeeper and general telecom concepts and terminology.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing

impaired.

Related Documents

For more information, see the following documents in the Services Gatekeeper set:

- *Oracle Communications Services Gatekeeper API Management Guide*
- *Oracle Communications Services Gatekeeper Application Developer's Guide*
- *Oracle Communications Services Gatekeeper Accounts and SLAs Guide*
- *Oracle Communications Services Gatekeeper Communication Service Reference Guide*
- *Oracle Communications Services Gatekeeper Security Guide*
- *Oracle Communications Services Gatekeeper Licensing Guide*
- *Services Gatekeeper Actions Java API Reference*
- *Services Gatekeeper Java API Reference*
- *Services Gatekeeper OAM Java API Reference*

1

About Extending and Replacing Portal Graphical User Interfaces

Oracle Communications Services Gatekeeper offers you both RESTful and SOAP based APIs that you can use to incorporate functionality from the Partner Portal and API Management Portal, Partner Portal, and Network Service Supplier Portal into your API management, customer relationship management (CRM)/partner relationship management (PRM) systems, intranets and extranets.

Integrating the PRM Portals with External Systems

Administering service provider and application accounts for Services Gatekeeper can be a work-intensive task for operators. Using CRM/PRM applications that are built using the Partner Relationship module can allow operators to shift some of that work to the service providers themselves. This gives those service provides a defined and structured channel both to communicate any changes they want and to monitor their own usage statistics. The operator's task is reduced to simply approving the pre-entered changes, dramatically reducing administration overhead. Using simple Web Service calls, the integrated PRM application can manage a wide range of Services Gatekeeper service provider account services.

The PRM interfaces support Services Gatekeeper's service provider and application administration model, which is described more fully in "Understanding the API Management Model".

> **Note:** Users, both operator-based and service-provider-based, who have been given appropriate permissions can also interact with Services Gatekeeper management systems using JMX-based solutions. For information on the MBeans available for use and the attributes and operations they expose, see "Managing SLAs" in *Services Gatekeeper Accounts and SLAs Guide*.

Securing the PRM API

This section explains how to secure the PRM API.

Securing PRM Web Services

The PRM Web Services module uses WS-Security to ensure the security of the Web Services based interaction between the PRM and its CRM/PRM application clients. Each request is authenticated using a username token or X.509 certificate that is included in the Simple Object Access Protocol (SOAP) header. For more information

on how this works, see "SOAP Header Elements for Authentication" in *Services Gatekeeper Application Developer's Guide*. Although the context in that section is the SOAP headers of traffic requests, the mechanism described is identical to the one used in with the PRM module.

Note: For backward compatibility purposes, a session ID-based login mode is also supported.

Deployment Example

The Partner Relationship Management module consists of two parts:

- The Web Service interfaces that are used by the CRM/PRM application
- The implementation of these Web Services

Figure 1–1 Deployment example

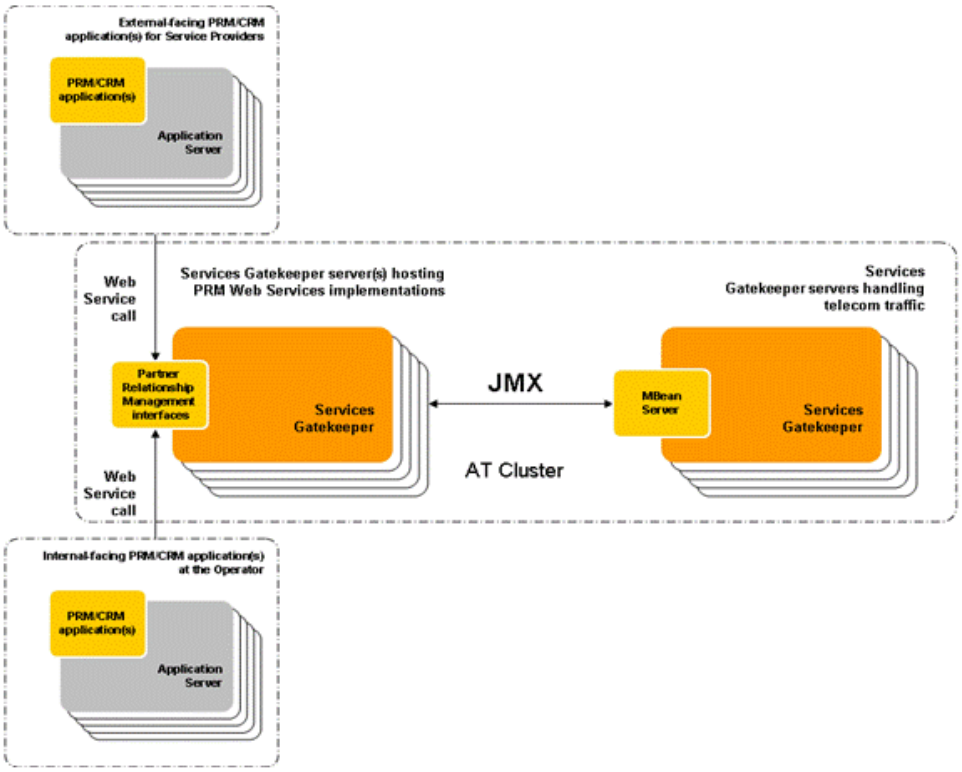


Figure 1–1 shows that there are two CRM/PRM applications, one supporting the Service Provider set of interfaces and the other supporting the more comprehensive Operator set of interfaces. Each of these applications uses Web Service calls to communicate with the host(s) running the PRM Web Services. The PRM Web Services module is deployed by default on the application tier (AT) Tier cluster. The PRM server(s) in turn use Java Management Extensions (JMX) to communicate with the Services Gatekeeper servers that actually handles the telecom traffic.

Managing APIs, Partners, and Network Service Suppliers Using the Services Gatekeeper Portals

Services Gatekeeper provides you with a trio of Web-based portal applications. The API and Partner Manager Portal along with the companion application, Partner Portal act as "front-office" offering to speed up the development and management of telecom applications. A third application, the Network Service Supplier Portal, can be used to provide network service interfaces configured from network resources for use in the creation of telecom applications.

About the API and Partner Manager Portal

The Partner Manager Portal is a private, internal portal hosted within a telecommunication firewall.

You use the Partner Manager Portal to manage the following:

- Service Provider Accounts
- Service Provider Groups
- APIs created in Network Service Supplier portal and communication service APIs and Web service APIs provided by Services Gatekeeper
- Contracts, service level agreements and services
- Application Instances

Menu selections and/or input fields on well-defined pages of the Partner Manager Portal application create a friendlier workflow, insulating you from the intricacies of OCSG administration and the complexity of the Services Gatekeeper environment.

For more information on how to use Partner Manager Portal, see *Services Gatekeeper Partner Manager Portal Online Help*.

About the Network Service Supplier Portal

Network service suppliers use the Network Service Supplier portal to configure network service interface APIs from their network resources. These network service interface APIs are published on the Partner Manager portal for use by the partner manager.

For more information on Network Service Supplier Portal, see *Services Gatekeeper API Management Guide* and *Services Gatekeeper Network Service Supplier Portal Online Help*.

About the Partner Portal

Partners who use the Partner portal are assigned to service provider groups and use Partner Portal to create applications.

Note: About partners and partner groups:

- Partner Manager Portal and Partner Portal use the term "partner" for the term "service provider" found in Services Gatekeeper.
- Partner Manager Portal and Partner Portal use the term "partner group" for the term "service provider group" found in Services Gatekeeper.

All applications created in the Partner portal require approval from within the API and Partner Manager portal before they are available to be marketed. Additionally, all subsequent changes made to an application when it is in use also require approval from within Partner Manager.

For more information on Partner Portal, see *Services Gatekeeper Partner Portal Online Help*.

Extending Portals

Services Gatekeeper allows you to extend the Partner and API Management Portal and Partner Portal by adding new functionality.

Note: To extend Partner and API Management Portal and Partner Portal, you must have:

- A valid partner manager account. Partner and network service supplier accounts do not have the necessary privileges to extend the portals.
- Access to Partner and API Management Portal.

You can extend Partner and API Management Portal and Partner Portal in the following ways:

- Adding a Page to a Portal Module
- Adding a Custom Module to a Portal Application

About the Default Modules and Pages

The Services Gatekeeper portals include the following default modules:

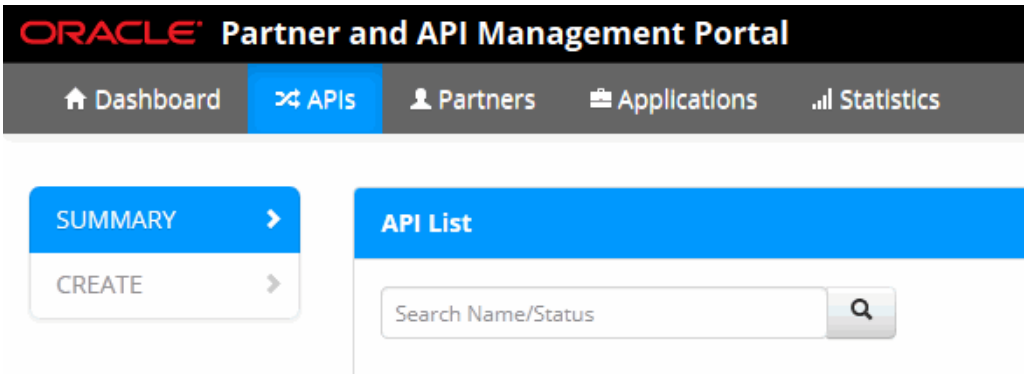
- Partner and API Management Portal: **Home**, **APIs**, **Partners**, **Applications**, and **Statistics**
- Partner Portal: **Home**, **Applications**, and **Statistics**

Each module includes one or more default pages. For example, the following list shows the default pages provided with the Partner and API Management Portal modules:

- Home module: **Overview** and **Workflow**.
- APIs module: **Summary** and **Create**
- Partners module: **Summary** and **Groups**
- Applications module: **Summary**
- Statistics module: **API Usage**, **API Response**, **API Failure**, **API Adoption**, **API Parameter**, **App Usage**, **App Response**, **App Failure**, and **Sub Usage**

Figure 1–2 shows the Partner and API Management Portal GUI. The default modules are listed horizontally along the top of the screen, and the default pages for the **APIs** module are listed vertically in the navigation menu.

Figure 1–2 Partner and API Management Portal Default User Interface



Points to Consider When Extending the Portals

When you extend Partner and API Management Portal and Partner Portal, keep in mind the following:

- Plan your extensions thoroughly before you implement them.
- Before you make changes, copy and store all important files, such as the **ocsgString.js** and WAR files.
- Test your extensions on a development environment before you apply it to a production environment.
- Preserve your users' data. It is a good idea to inform users when you plan to deploy changes to the production environment. This ensures that, before you update the portal application in your production environment, all users of the specific portal application have logged off.
- Maintain a uniform look and feel for your application. Your additions blend better with the existing portal application if you maintain the application's look and formatting. Maintain one default style sheet language.
- Ensure that the online help for any additional page or module can be integrated into the existing help.

About the Naming Conventions for Help Files

Online help files must adhere to the following naming conventions:

portalIdentifier_moduleName_fileName.**html**

Where:

- *portalIdentifier* identifies the portal application:
  - Use **pmp** for Partner and API Management Portal
  - Use **ppg** for Partner Portal
- *moduleName* identifies the module, such as **Home**, **Applications**, or **Statistics**
- *fileName* identifies the help file

For example, the help file name for a new page in the **Statistics** module of Partner Portal could be **ppg_statistics_pagehelp.html**.

About the Naming Conventions for Help IDs

Online help IDs must adhere to the following naming conventions:

*portalIdentifier_moduleName_***menu***sequenceNumber*

Where:

- *portalIdentifier* identifies the portal application:
  - Use **pm** for Partner and API Management Portal
  - Use **pp** for Partner Portal
- *moduleName* identifies the module, such as **Home**, **Applications**, or **Statistics**
- *sequenceNumber* identifies the position in the sequence.

For example, the help ID for a new help page in the **Statistics** module of Partner Portal could be **pp_statistics_menu2**.

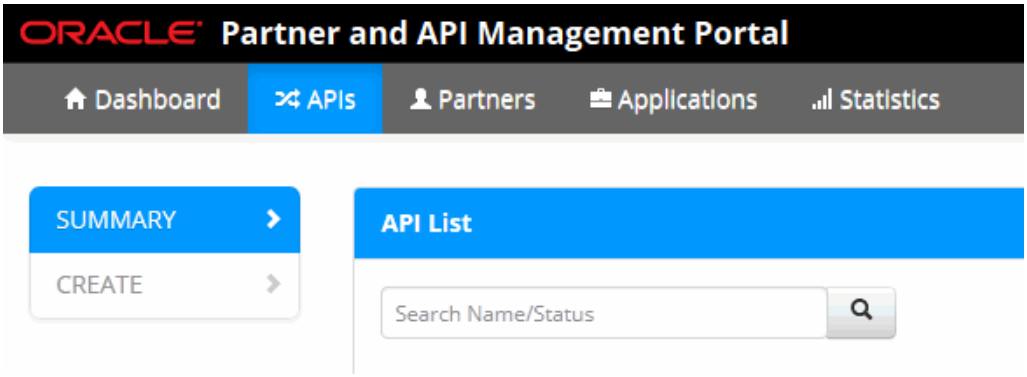
About the Extension Example

The subsequent sections cite an example for a simple new feature, which requires the following changes to the Partner and API Management Portal:

- Adding a new module named **asdfg**.
- Adding a new page named **hello** to the **asdfg** module.
- Adding a new help page for the **asdfg** module and **hello** page.

Figure 1–3 shows the new **asdfg** module and **hello** page added to the Partner and API Management Portal application.

Figure 1–3 Example of a Module Extended in Partner and API Management Portal



Adding a Page to a Portal Module

You can add a page to an existing module or to a new module in the portal applications.

To add a page to a portal module, you perform the following general steps:

1. Deploying the New Page
2. Providing the Help Content for the New Page
3. Updating the Help WAR File for the Portal Application
4. Redeploying the WAR File in Your WebLogic Server Domain

Deploying the New Page

To deploy a new page:

1. Prepare the web content for the new page. This step is performed outside of Services Gatekeeper.

   Create an HTML file for the new page using your preferred tool. Ensure that you include all required user interface elements.

2. Deploy the new page in the Administration Console.

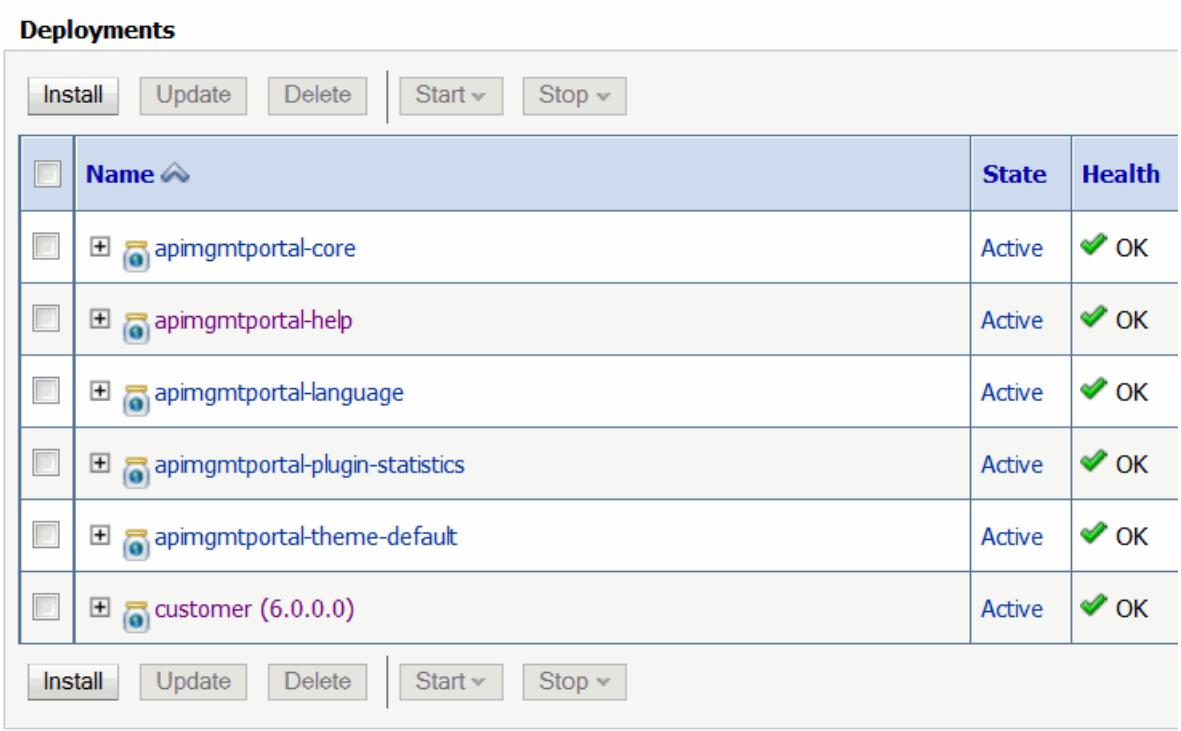
   Deploy your new HTML page as part of a web application (WAR archive) in WebLogic Server. For instructions on how to install a web application in your Oracle WebLogic Server instance, see *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help* here:

   `http://docs.oracle.com/middleware/1213/wls/WLACH/core/index.html`

   Figure 1–4 shows the sample **customer** web application deployed in a sample WebLogic Server instance which houses the *Penguin Partner and API Management Portal* application shown in Figure 1–3.

   The *customer*.**war** file contains the example **hello.html** file.

Figure 1–4 Customer Web Application



Providing the Help Content for the New Page

To create online help for the new page:

1. Create your online help content and save it in an HTML file. This step is performed outside of Services Gatekeeper.

   Create an HTML file with all of the help content that you require for the new page using your preferred tool. Save the file using the naming convention described in "About the Naming Conventions for Help Files".

For example, the help file for the **hello** page is saved as **pm_asdg_helloWorld.html**.

2. Create the help ID

   Create the help ID that the portal application uses to reference the online help you created in step 1.

   For example, the help ID for the **hello** page is set to **pm_asdfg_menu2**.

Updating the Help WAR File for the Portal Application

At this point, you have successfully added the content page to provide the new logic feature in your portal application. You now need to support that logic by providing online help for the users who visit and use that page.

You do this by updating the WAR archive file for the appropriate help JAR file using these steps:

1. Getting a Working Copy of the Portal Help File
2. Getting a Working Copy of the Help JAR File for the Portal Application
3. Adding the New Help HTML File to the Working Copy of the Help JAR File
4. Updating the map.xml File to Include the Target and URL for the Help HTML File
5. Updating the Navigation List for the Help to Include the New Topic

Getting a Working Copy of the Portal Help File

The **apimgmntportal-help** WAR file contains the help JAR file for the Portal application. Extract the files from the WAR file so you can update the help JAR file inside it.

To get a working copy of the Portal Help file:

1. Login to the Administration Console where your portal applications are deployed.
2. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
3. In the left pane of the Console, select **Deployments**.
4. In the **Deployments** table, click **apimgmntportal-help**.

   It is the second entry shown in Figure 1–4.
5. In the **Overview** tab of **Settings for apimgmtportal-help**, locate the entry for **Path**.
6. Right-click the URL and select **Copy**.
7. Save this file in a different location.

Getting a Working Copy of the Help JAR File for the Portal Application

Services Gatekeeper names the help JAR files as:

- **sgpmp_help.jar** for Partner and API Management Portal
- **sgppg_help.jar** for Partner Portal

To extract the JAR file for the Portal application:

1. Go to the location where you stored the WAR file for **apimgmntportal-help**.
2. Extract all of the files from **apimgmntportal-help** to a temporary directory.
3. Go to the temporary directory and locate the appropriate portal help JAR file:

- **sgpmp_help.jar** for Partner and API Management Portal
- **sgppg_help.jar** for Partner Portal

4. Right-click the appropriate portal help JAR entry and select **Copy**.
5. Save the copy of the portal help JAR file to a different location.

Adding the New Help HTML File to the Working Copy of the Help JAR File

To add the help HTML file you created to the working copy of the portal help JAR file:

1. Go to the location where you stored the portal help JAR file.
2. Extract all the files that are compressed in the portal help **.jar** file.
3. Go to the location where you stored the help **.htm** file.
4. Right-click the file and select **Copy**.

   In our example, *pm_asdfg_helloWorld***.htm** is copied.
5. Go to the location where you extracted the contents of portal help **.jar** file.
6. Paste the help HTML file.

   *pm_asdfg_helloWorld***.htm** is added to the set of extracted files.

The next steps are completed in this directory.

Updating the map.xml File to Include the Target and URL for the Help HTML File

At this point you have added to the contents of the portal help JAR file and are in that directory. You now need to add the mapping to refer to the help contents for the new application page. You do this by updating the **map.xml** file.

To update the **map.xml** file:

> **Note:** Verify that you are in the directory which contains the extracted HTML files of the appropriate portal help JAR file.

1. In the directory which contains the extracted HTML files of the portal help JAR file, locate the **map.xml** file.
2. Open the **map.xml** file to edit it.
3. Scroll down to the final entry (**</map>**).
4. Input the following line above the **</map>** entry.

   ```
   <mapID target="Your_New_HelpID " url="Your_Help_HTML_FileName" />
   ```

   In our example, the entry would read as:

   ```
   <mapID target="pm_asdfg_menu2 " url="pm_asdfg_helloWorld.htm" />
   ```

5. Save the **map.xml** file.

The next steps are completed in this directory.

Updating the Navigation List for the Help to Include the New Topic

You now need to update the bookmarks for the help page to display this entry. You do this by updating the **toc.xml** file.

To update the **toc.xml** file:

Note: Verify that you are in the directory which contains the extracted HTML files of the appropriate portal help JAR file.

1. In the directory which contains the extracted HTML files of the portal help JAR file, locate the **toc.xml** file.
2. Open the **toc.xml** file to edit it.
3. Scroll down to the final entry (**</toc>**).
4. Enter the following line above the **</toc>** entry.

   ```
   <tocitem target="Your_Help_HTML_FileName " text="Heading_for_the Topic" />
   ```

   In our example, the entry would read as:

   ```
   <tocitem target="pm_asdfg_helloWorld.htm" /> " url="How to Say Hello" />
   ```

5. Save the **toc.xml** file.

Readying the Help WAR File for Redeployment

Use the appropriate file compression tool, such as 7-zip, to ready the help WAR file:

1. Compress the updated contents for the help JAR for the portal application.

   In this example, the user compresses the extracted (and updated) set of the application help **.htm** files as the **sgpmp-help.jar** file.
2. Replace the help JAR file in the **apimgmntportal-help.war** file.

   In this example, the user uses the newer **sgpmp-help.jar** to replace the older **sgpmp-help.jar** in the directory where the user extracted the contents of the **apimgmntportal-help.war** file.
3. Compress the updated contents of the **apimgmntportal-help.war** file.

   In this example, the user compresses the extracted (and updated) set of the **apimgmntportal-help.war** file.

Redeploying the WAR File in Your WebLogic Server Domain

To redeploy the **apimgmntportal-help.war** file in your WebLogic Server domain:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
2. In the left pane of the Administration Console, click **Deployments**.
3. In the **Deployments** section, select the check box for **apimgmntportal-help**.
4. Click **Stop**, and then click **Force Stop Now**.

   The entry in the **State** column for **apimgmntportal-help** changes to display **Prepared**.
5. When the **State** column for **apimgmntportal-help** displays **Prepared**, click **Update**.

   The Update Application Assistant (no bold) dialog box appears.
6. For the **Source Path** field in the **Update Application Assistant** dialog, select **Change Path**.

7. In **Update Application Assistant**, select or enter the file path that represents the updated **apimgmntportal-help.war** file.
8. Click **Finish**.

Adding a Custom Module to a Portal Application

You can add custom modules to both portals. When you add a custom module to Partner API Management Portal or Partner Portal, the custom module is displayed to the right of **Statistics**, the last default module in the respective portal. Each custom module is displayed in the order in which it was added.

For each custom module:

- Use a unique name. Do not include special characters.
- Provide a valid plugin.
- Set up at least one page to display.
- Set up the help content for the page.

> **Caution:** The online help that Services Gatekeeper supplies with your Partner and API Management Portal provides support for the default configuration only.

Adding Custom Modules

To add a module to either portal:

1. Access the portal application to which you are adding a page.
   - For Partner and API management Portal, the default location is:

     `http://IP_address:port/portal/partner-manager/index/login.html`

   - For Partner Portal, the default location is:

     `http://IP_address:port/portal/partner/index/partnerLogin.html`

   Where, *IP_address* is the host system running Services Gatekeeper. The default port is 8001.
2. Click **CUSTOMIZATION** in the navigation list of the **Settings** module.
3. Scroll down to the **Module** field for the appropriate portal (**Partner and API management Portal** or **Partner Portal** section.
4. Click **Add New Plugin**.
5. In the **Name** field, enter a unique name for the module.
6. Select the **Visible** check box below the **Pages** table.
7. At this point, add at least one page for your module. Complete the steps described in "Adding a Page to the Portal Application".
8. Click **Submit**.

Adding a Page to the Portal Application

You can add a page to an existing module or to a custom module.

To add a page to a portal application:

1. Access the portal application to which you are adding a page.
   - For Partner and API management Portal, the default location is:

     `http://IP_address/portal/partner-manager/index/login.html`

   - For Partner Portal, the default location is:

     `http://address:port/portal/partner/index/partnerLogin.html`

   Where, *IP_address* is the host system running Services Gatekeeper. The default port is 8001.
2. Click **CUSTOMIZATION** in the navigation list of the **Settings** module.
3. Scroll down to the **Module** field for the appropriate portal (**Partner and API management Portal** or **Partner Portal** section.
4. Locate the module from the displayed modules.
5. Click the module to access its fields.
6. Click **Add Page**.

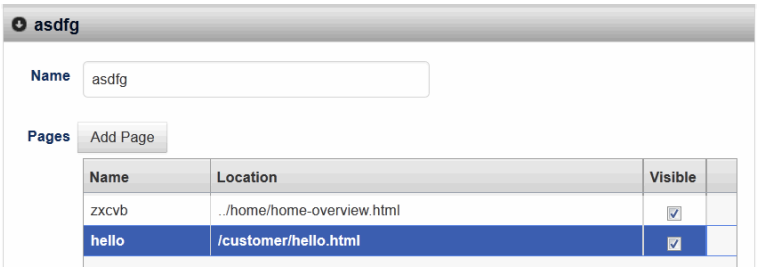
   A row is added to the Page table for the module you selected.
7. In the new row added to the table, enter the details for the new page:
   a. In the **Name** column, the name to display for this page. For example *hello*.
   b. In the **Location** column, the location for the URL for this page.

      The *customer***.war** file contains the example **hello.html** file. The location entry reads:

      /customer/hello.**html**
   c. In the **Visible** column, click the check box.
8. Click **Add**.
9. Click **Submit** in this section.

Figure 1–5 shows the completed entries.

Figure 1–5 Details Entered for the New Page



Verifying the Addition in the Portal Application

To verify the extensions to the portal application:

1. Do one of the following:
   - If you have logged out of the updated portal application, sign in to the portal application.

- If you are still signed in to the extended portal application, refresh the application. (Click **F5** on your keyboard).

2. In the Header bar, click the tab for the module to which you added the page.
3. In the vertical navigation menu list, select the new entry.

   In the example case, the user clicks the **asdfg** module and then the **hello** menu entry in Partner and API Management Portal.
4. To verify the help support for the updates to your portal application, Click **Help** on the Header bar. The help document should appear.
5. In the vertical navigation menu list, select the new entry.

2

Understanding the API Management Model

This chapter introduces the Oracle Communications Services Gatekeeper management model, lists the types of accounts and account states that it uses, and explains how these accounts relate to the PRM portals.

Understanding the Services Gatekeeper Management Model

The Services Gatekeeper management model defines roles for operators and application service providers, and defines the interactions between them. An operator runs the network in which Services Gatekeeper is installed. Operators have partners who provide one or more applications to interact with the operator's network. These partners are the application service providers and they can be in-house or external to the operator.

Operators manage application service provider accounts. An application service provider registers with Services Gatekeeper and is given a service provider account. To support tiering, service provider accounts are collected into Service Provider Account Groups. These groups that are associated with service level agreements (SLAs).

Service provider accounts include individual application accounts, registered on their respective service provider accounts. Like service provider accounts, application accounts are grouped into application account groups. Again, SLAs are associated with applications by using the application group.

Finally, the model also includes the idea of the application instance, which is tied to a specific instance of the application and is used in the traffic authentication process.

For more information on SLAs and accounts, see "About Service Level Agreements and Accounts" in *Services Gatekeeper Accounts and SLAs Guide*.

The Partnership Management module allows for management of:

- Service provider accounts
- Application accounts
- Application groups
- Service provider SLAs:
  - Provisioned and enforced in one cluster
  - Provisioned and enforced across clusters (used for establishing geo-redundancy)
- Service provider node SLAs
- Application SLAs

- Provisioned and enforced in one cluster
- Provisioned and enforced across clusters (used for establishing geo-redundancy)

Account States

Service provider accounts and application accounts have one of these states:

- REGISTERED

  The service provider has requested that an account be registered, but the operator has not yet approved or disapproved it.

- ACTIVE

  The operator has approved the account the service provider registered.

- INACTIVE

  The account has been deactivated, either temporarily or as a step toward being deleted.

- UPDATE_PENDING

  The service provider has requested an update of the account, and this update has not yet been approved by the operator.

- DELETE_PENDING

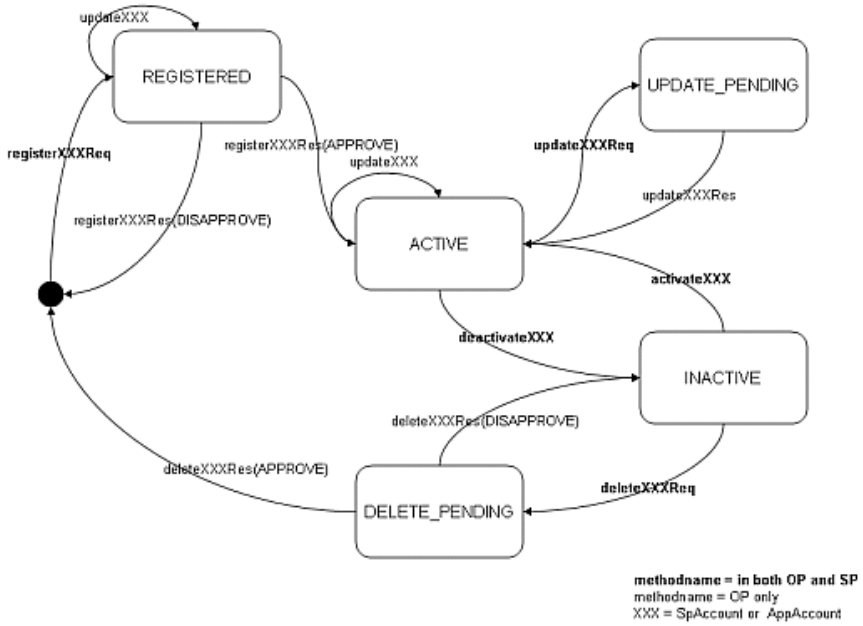
  The service provider or the operator has requested that the account be deleted. This is an intermediate state. The operator can, for example, use this state to process all charging data records for the account before deleting it.

  Note: Charging data records may still be in the Services Gatekeeper, even when the account information is deleted. Make sure all data has been processed before deleting an account.

  Once an account is deleted, all data about the account is removed from the Services Gatekeeper.

The possible state transitions are outlined in Figure 2–1.

Figure 2–1 States and state transitions



There are two sets of interfaces in the partner relationship module (PRM) module. The service provider interfaces give application service providers access to information relative to their own accounts and applications. The operator interfaces allow operators to manage their service providers. These include access to a much broader range of management functions.

In Figure 2–1, the method names in **bold** can be executed by both the operator and the application service provider. The non-bold methods names can only be executed by the operator. **XXX** indicates that the methods are valid for both service provider accounts and application accounts.

Administering PRM Administrative Users

You create and manage PRM administrative users in the same way as other Services Gatekeeper administrative users, using the Administration Console or the **ManagementUserMBean**. At least one PRM-OP user must be set up before the PRM interfaces can be used, and you are prompted to create this user during installation. For information on the **ManagementUserMBean**, see the "All Classes" section of the *Services Gatekeeper OAM Java API Reference*. For more information on managing administrative users, see "Managing Users and User Groups" in *Services Gatekeeper System Administrator's Guide*.

Table 2–1 describes the characteristics of an administrative user.

Table 2–1 Contents of wlng_mgmt_users Database Table

Field	Type	Description
username	varchar(255)	Authentication name of the administrative user.

Table 2–1 (Cont.) Contents of wlng_mgmt_users Database Table

Field	Type	Description
state	int(11)	Possible values are: ■ 0: Activated ■ 1: Deactivated Transitional states (Registered, Update pending, Delete pending) are stored temporarily as properties of the account.
type	int(11)	Kind of user. Options are: ■ 0: OAM (Console-based user) ■ 1: PRM-OP (Operator using PRM) ■ 2: PRM-SP (Service Provider using PRM)
password	varchar(255)	Administrative user password. 3DES Encrypted.
userlevel	int(11)	Privilege level of user. See Table 2–2 below for values.
groupname	varchar(255)	Allows administrative us.ers to be grouped for ease of management
stored_ts	bigint(20)	Tablespace

Table 2–2 Privilege Levels

Level	Services Gatekeeper Role Type
1000	Equivalent to Administrative Access on WebLogic Server (WLS). Can: ■ Manage servers and server configuration ■ Deploy applications ■ Control all Services Gatekeeper management functions
666	Equivalent to Deployer Access on WLS. Can: ■ View server configuration and make some changes ■ Have read/write access on Services Gatekeeper management functions
333	Equivalent to Monitor Access on WLS. Can ■ View server configuration ■ Have read-only access to Services Gatekeeper management functions
0	Equivalent to Anonymous Access on WLS. Can: ■ Use servers. PRM-SP users have this privilege level

Note: Service providers may also have direct access to account management functions by using JMX if the service provider has appropriate user permissions. Operator make this decision.

3

Common Partner Relationship Management Use Scenarios

This section outlines common use patterns using the Oracle Communications Services Gatekeeper Partner Relationship Management (PRM).

> **Note:** This chapter describes the use patterns associated with the older SOAP-based APIs. See "Partner Relationship Management SOAP-based Web Services Interface" for instructions on how to extend and customize the GUIs used in older (Pre-6.0) Services Gatekeeper releases.
>
> See "API Management REST-based API" for instructions on how to use the newer API Management API.

Registering a new Service Provider Account

1. The Service Provider applies for a Service Provider Account using Service Provider Login::registerSPAccountReq(...) providing basic information such as desired account name, contact details, and so forth as part of the application.
2. The Operator lists new requests using `&rsquor`. The list can be filtered to display only service provider accounts in the REGISTERED state.
3. The request is inspected by the Operator, and can be changed using Operator::updateSpAccount(...).
4. If the request is approved by the Operator, the Service Provider Account is first associated with a Service Provider Group and Service Level Agreement (SLA) and then approved using `Operator::registerSpAccountRes(...)`. For information on setting up a Service Provider Group, see "Operator: Creating a Service Provider Group". For information on setting up an SLA, see “Managing SLAs” in *Services Gatekeeper Accounts and SLAs Guide*.

   > **Note:** The Service Provider Group that the Service Provider Account belongs to can be changed at any time using Operator::moveSpToGroup(...).

5. Finally, the Service Provider is notified that the request has been approved by using any of the contact channels that were detailed when the request was submitted. The CRM/PRM application implementor must set up a mechanism for

this communication, or for the communication of any other account changes that may occur.

Registering a new Application Account

1. The Service Provider applies for an application account by using Service Provider::registerAppAccountReq(...).
2. The Operator lists new request using Operator::listAppAccounts(...) You can filter this list to display only applications in the REGISTERED state belonging to a specific service provider account.
3. The request is inspected by the Operator, and can be changed by using Operator::updateAppAccount(...).
4. If the request is approved by the Operator, the Application Account is first associated with an Application Account Group and SLA and then approved by using Operator::registerSpAccountRes(...). For information on setting up an Application Account Group, see "Operator: Creating an Application Account Group". For information on setting up an SLA, see “Managing SLAs” in *Services Gatekeeper Accounts and SLAs Guide*.

   Note: The Application Account Group that the Application Account belongs to can be changed at any time using Operator::moveAppAccountToGroup(...).

5. Finally, the Service Provider is notified that the request has been approved by using any of the contact channels that were supplied with the original request for a Service Provider Account. The CRM/PRM application implementor must set up a mechanism for this communication, or for the communication of any other account changes that may occur.

At this point the Application Account is in the ACTIVE state. To actually begin sending traffic to Services Gatekeeper, however, the application must now apply for an Application Instance ID to be used to authenticate the account. For information on Application Instances, see "Application Instances"; for information on how to register an Application Instances, see "Registering a new Application Instance".

Registering a new Application Instance

1. The Service Provider applies for an Application Instance by using Service Provider::registerAppInstGroupReq(...). The request include desired properties to be associated with the instance.
2. The Operator lists new requests using Operator::listAppInstGroups(...) The list can be filtered to display only requests from specific SP Accounts and App Accounts for instances in the REGISTERED state.
3. The request is inspected by the Operator, and can be changed using Operator::updateAppInstGroup(...).
4. If the request is approved by the Operator, it is approved using Operator::registerAppInstGroupRes(...).
5. Finally, the Service Provider is notified that the request has been approved by using any of the contact channels that was detailed when applying for a Service provider Account. The CRM/PRM application implementor must set up a

mechanism for this communication, or for the communication of any other account changes that may occur.

Once the Service Provider has a Service Provider Account, an Application Account, and an Application Instance ID, traffic can be sent to Services Gatekeeper.

Operator: Creating a Service Provider Group

1. The Operator defines the use privileges for a particular Service Provider Group.
2. These use privileges are formalized in a Service Provider Service Level Agreement XML file. For more information see "About Service Level Agreements and Accounts" in *Services Gatekeeper Accounts and SLAs Guide*.
3. The Operator creates the Service Provider Group using Operator::createSpGroupByType(...) and assigns the newly created Service Provider SLA to it. An ID for the group is also defined.

Operator: Creating an Application Account Group

1. The Operator defines the use privileges for a particular Application Account Group.
2. These use privileges are formalized in an Application Service Level Agreement XML file. For more information see "About Service Level Agreements and Accounts" in *Services Gatekeeper Accounts and SLAs Guide*.
3. The Operator creates the Application Account Group using Operator::createAppGroupByType(...) and assigns the newly created Application SLA to it. An ID for the group is also defined.

Service Provider Requests an Account Update

The Service Provider can request an update to any of its account entities. The request might cover SLA data or user defined properties. The Operator is responsible for approving or disapproving the request.

1. To request an update for a Service Provider account, the Service Provider uses Service Provider::update<SpAccount | AppAccount | AccountAppInstGroup>Req(...) as appropriate.
2. Once the Service Provider has requested an update, the state of the account is changed to UPDATE_PENDING until the Operator has inspected the update request and either approved it or disapproved it using Operator::update<SpAccount | AppAccount | AccountAppInstGroup>Res(...).

> **Note:** When an account is in the UPDATE_PENDING state, no further requests to update the account are allowed until the initial request has been approved or disapproved.

Service Provider Deactivates an Account

The Service Provider can deactivate any of its account entities. The deactivation takes affect immediately. No traffic is allowed through an account that is deactivated. The impact of a deactivation is cascaded through the Service Providers system:

- When a Service Provider Account is deactivated, none of the applications run by the service provider are able to send traffic through Services Gatekeeper.
- When an Application Account is deactivated, none of the applications that are associated with that Application Account are able to send traffic through Services Gatekeeper.
- When an Application Instance is deactivated, only that Application Instance is unable to send traffic through Services Gatekeeper. Other applications are not affected.

An account must always be deactivated before it can be deleted.

1. To deactivate an account the Service Provider can use either Service Provider::deactivate<SpAccount | AppAccount | AccountAppInstGroup>Req(...), depending on the type of account.
2. To deactivate an account the Operator can use either Operator::deactivate<SpAccount | AppAccount | AccountAppInstGroup>Req(...), depending on the type of account.

Service Provider Requests an Account Deletion

The Service Provider can request to have any of its account entities deleted. The deletion does not take effect until after the request has been approved by the Operator.

An account must always be in state INACTIVE before it can be deleted.

1. To request to delete an account, either Service Provider::delete<SpAccount | AppAccount | AppInstGroup>Req(...) is used.
2. When the Service Provider has requested an account deletion, the state of the account is changed to DELETE_PENDING until the Operator has inspected the request and either approved it or disapproved it using Operator::delete<SpAccount | AppAccount | AppInstGroup>Res(...).

Note: When the request to delete an account is approved the account is deleted from the database. It is the Operator's responsibility to make sure that outstanding charging data records are processed before the deletion takes place.

If the request to delete the account is disapproved, the state of the account becomes INACTIVE.

Communicating General Information Between Service Provider and Operator

The Service Provider can communicate desired updates to the Operator using the update methods, as described in "Service Provider Requests an Account Update". Each update request can contain a set of properties in the form of name-value pairs, which are defined by the implementors of the CRM/PRM application.

Retrieving Charging Data Records

Both the Service Provider and the Operator can retrieve Charging Data Records using <Operator | Service Provider>::listCdrs(...). The operator can retrieve call details records (CDRs) for all Service Providers, while the Service Provider only has access to Charging Data Records generated by its own applications. Results can be filtered.

Retrieving Statistics

Both the Service Provider and the Operator can retrieve statistics using <Operator | Service Provider>::getStatistics(...). The operator can retrieve statistics for all Service Providers, while the Service Provider only has access to its own applications. A set of filters can be used, including Application Account IDs and time intervals.

Retrieving Alarms

The Operator can retrieve alarms using Operator::listAlarms(...). The operator can retrieve alarms generated by all Service Providers and platform related alarms. A set of filters can be used, including timestamps and severity levels.

4

Creating Custom Actions for Your APIs

This chapter explains how to create a custom action to make available to your APIs on the **Actions** tab of the Services Gatekeeper Partner and API Management Portal.

Understanding When to Create Custom Actions

Services Gatekeeper provides a set of actions that you use to manipulate requests and response messages to your APIs. These default actions listed in "Actions Provided by Services Gatekeeper" in *Services Gatekeeper API Management Guide*.

Most of the default actions have specialized applications. In contrast, the Groovy action is specifically is available for you to add any level of sophistication required for manipulating request and response messages for API traffic. The drawback to using the Groovy action is that you must replicate that code for each API that requires it. To remove that restriction, you also have the option to create your own custom action code and make it available to all APIs. Once created and added to the external actions library, custom actions are available for use by all of your APIs. You can use it with the Actions Java API or by using the Partner and API Management Portal user interface. The portal is generated dynamically, so your custom actions appear automatically on the **Actions** tab of the portal.

You can use any of the classes in the Actions Java API Reference in the Java/Groovy code you use to create a custom action. See "All Classes" in the Actions Java API Reference document for information.

Creating and Adding a Custom Action to Services Gatekeeper

Simply put, you create a custom action by creating and compiling it as a custom Java program, and packaging it as an .ear file. You then add the .ear file to Services Gatekeeper, and your custom action is then available to use alongside the Services Gatekeeper default actions.

The "Example Custom Actions" section also be a help to you.

During this process you can also add custom events that appear in event data records (EDRs) for debugging purposes. For details on EDRs, see "Managing and Configuring EDRs, CDRs, and Alarms" in *Services Gatekeeper System Administrator's Guide*.

Creating a custom Action:

1. Create and compile the Java code for the functionality that your implementation requires into the **MyAction** (using **HttpContext**) and **action.java** files. See the "Example Black List Action" section for examples for these files, and see the "All Classes" section of *Actions Java API Reference* for details on **HttpContext**.

Use **HttpContext** for the context in your action file. Here is a Java example:

```
System.out.println(context.getClientRequest().getMethod())
```

The **MyActionConfig.java** file is limited to one level of properties that use the Java primitive data types. It cannot use hierarchies.

An example **MyActionConfig.java** file:

```
@XmlRootElement
public final class MyActionConfig implements Serializable{
  /**
   * UUID.
   */
  private static final long serialVersionUID = 1L;

  private String requestUrl;

  public String getRequestUrl() {
    return requestUrl;
  }

  public void setRequestUrl(String requestUrl) {
    this.requestUrl = requestUrl;
  }
}
```

2. Add the **DafAction** class to the action file.

   DafAction uses these parameters to define how the new action (**MyAction** in this example) is presented and used in Services Gatekeeper:

   - **configBean** (Class) - Defines the Java Bean class that holds the configuration for this class. The Action class must be included the class as a Generic in the Action interface that it implements, as well as have it as the first parameter in the **init()** method.
   - **description** (String) - And informal description of the action. Can be used by PRM API clients.
   - **flowRestriction** (FlowRestriction) - Optional. Defines where this Action can be used:
     - **NONE** - The default. Does not restrict the action, so it can be used in either the request or response flow.
     - **REQUEST** - Restricts the action to use on the request flow.
     - **RESPONSE** Restricts the action to use on the response flow.
   - **hidden** (Boolean) - True specifies that the action be hidden. The default is false.
   - **name** (String) - The name that identifies the action. Must be unique.
   - **flowRestriction** (FlowRestriction ENUM) - Defines where this Action can be used:

   DafAction example:

```
@DafAction(name = "My", hidden = false, configBean = MyActionConfig.class,
  description = "description here.")
public final class MyAction implements Action<MyActionConfig> {

  private MyActionConfig configuration;
```

```
  /**
   * Init configuration.
   * @param pConfiguration configuration
   * @param apiConfiguration api configuration
   */
  @Override
  public void init(MyActionConfig pConfiguration, final ApiConfiguration
apiConfiguration) {
    this.configuration = pConfiguration;
  }
  /**
   * Process the request.
   * @param context the context
   * @throws ActionProcessingError if an attribute using the
   * ${x}-syntax in the url does not exist.
   */
  @Override
  public void process(HttpContext context) throws ActionProcessingError {
 // to be added
  }
}
```

See "Header Validation Example" and "Example Black List Action" for other examples.

3. Register your action to use by adding it to the **MyListener.java** file.

```
public class MyListener extends ApplicationLifecycleListener {
  public void preStart(ApplicationLifecycleEvent evt) {
    try {
      ActionRegistrationManager.getInstance().
getActionRegistrationClientManager().registerAction(MyAction.class);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

4. (As needed) Add any custom events that you want included in event data records (EDRs) by:

   - Adding these classes to your **action.java** file:

     ```
     import com.bea.wlcp.wlng.api.plugin.context.RequestContext;
     import com.bea.wlcp.wlng.api.plugin.context.RequestContextManager;
     ```

   - Adding this statement to the process function of the actions class in your **action.java** file:

     ```
     final RequestContext reqCtx = RequestContextManager.getCurrent();
     reqCtx.putEdr("MYKEY", "MYVALUE");
     ```

     Where *MYKEY* and *MYVALUE* are the key/value pair that will appear in EDRs.

     Note: You also need to set server default logging priority to DEBUG to ensure that your events appear in EDRs. For details see "Changing Log Levels in Services Gatekeeper" in the *Services Gatekeeper System Administrator's Guide*.

5. Add the *Gatekeeper_home*/**ocsg/modules/oracle.sdp.registration.prs_6.0.0.0.jar** and *Gatekeeper_home*/**ocsg/modules/oracle.sdp.daf-6.0.0.0.jar** libraries to your CLASSPATH.

   Note: Do not add these libraries to the .ear file at run time. Load them as a system library automatically at run time.

6. Compile your actions java code.
7. Add this code to the **weblogic-application.xml** file.

   ```
   <wls:listener>
     <wls:listener-class>com.MyListener</wls:listener-class>
   </wls:listener>
   ```

   If **weblogic-application.xml** does not exist, navigate to **Project Properties**, then **Project Facets**, then **Oracle WebLogic EAR Extensions**, then click **OK**.
8. Package the .jar file, and any other files or directories that the action requires into an .ear file.
9. Deploy the .ear file to the network tier cluster in your implementation.

Your new custom action is displayed and available for use in the Partner and API Management Portal, or by the Actions API Management REST-based API. See "Discovering Actions Chains Schemas" for information on how to obtain the action schemas.

Managing Custom Actions

This section explains how to manage your custom actions and make them available in the Partner and API Management Portal. This section explains how to manage actions using the Administration Console. These functions are also available as MBeans so you can use them programmatically. See the "All Classes" section of the *OAM Java API* for details.

Note: You can only perform these management operations on your custom actions. These operations fail and return an error if you attempt to use them on the default Services Gatekeeper actions.

You have these options for managing custom actions:

- **deprecateAction** - Deprecates the action. APIs that already use the action continue to do so, but no other APIs can add the action.
- **activateAction** - Reactivates a deprecated action. After using this operation to reactivate an action, you need to register the action before new APIs can use it.
- **deregisterAction** - Prevents the action from being added by any API. You must remove this action from all APIs before using this operation. If any APIs are using the action when you run this operation, an error is returned with a list of those APIs.
- **isRegisteredAction** - Checks whether an action with the same name has already been registered. This function prevents you from unintentionally overwriting an existing action. Be sure to use the fully qualified action class name.

To manage custom actions:

1. Start the Administration Console

   See "Starting and Using the Administration Console" in *Services Gatekeeper System Administrator's Guide* for details.
2. Click **Lock and Edit**.
3. Navigate to **OCSG**, then *servername*, then **Container Services**, then **DafActionRegistration**.
4. Select **Operations**.
5. Select a management option from the **Select an Operation** menu. You have these options for managing actions:
   - **activateAction**
   - **deprecateAction**
   - **deregisterAction**
   - **isRegisteredAction**
6. Enter the class name of the action to perform this action on in the **actionClassName** field.
7. Click **Invoke**.

Discovering Actions Chains Schemas

Use the **"loadActionSchemas"** method of the API Actions Management REST-based API to return the schemas for all actions available to a Services Gatekeeper implementation. This allows you to discover action details, such as required fields, without using the Partner and API Management Portal.

Example Custom Actions

This section includes some examples to illustrate the types of custom actions that you can create.

RESTful Request for an Attribute Example

This code snippit sends a RESTful request to an external URL for value using the **createCallout** method of the **HttpContext** class, and stores it in a custom key/value pair. Add this line to a custom action early in the action chain:

```
context.createCallout().withRequestUrl("http://myurl.com").withRequestMethod("GET"
).build().send("myattribute");
```

Where *myurl* is the URL to request the attribute from, and *myattribute* is the name of the attribute to request. After using this line in a custom action, you can then use the value for *myattribute* in a subsequent action:

```
HttpResponse httpResponse = (HttpResponse) context.getAttribute("myattribute");
```

It must be in a subsequent action because action requests are asynchronous to keep from halting the action chain.

For details on **HttpContext**, see the "All Classes" section of the Actions Java API Reference.

Header Validation Example

This example action allows you to add name/value pair test for a header/value as a requirement for manipulating traffic for an API. To make this work you would need to register the action with the lifestyle listener (substitute **HeaderValidationAction** for **myAction** in Step 3 above).

The **HeaderValidationAction.java** file:

```
/* Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved. */
package oracle.sdp.daf.action;

import oracle.sdp.daf.action.api.Action;
import oracle.sdp.daf.action.DafAction;
import oracle.sdp.daf.action.FlowRestriction;
import oracle.sdp.daf.action.api.ActionConfigurationException;
import oracle.sdp.daf.action.api.ActionProcessingError;
import oracle.sdp.daf.action.api.HttpContext;
import oracle.sdp.daf.config.ApiConfiguration;

/**
 * My custom action.
 */
@DafAction(name = "HeaderValidation", configBean =
HeaderValidationActionConfig.class)
public final class HeaderValidationAction implements
Action<HeaderValidationActionConfig> {

  private static final int HTTP_INTERNAL_ERROR = 500;

  private HeaderValidationActionConfig configuration;

  @Override
  public void init(HeaderValidationActionConfig pConfiguration, ApiConfiguration
apiConfiguration)
    throws ActionConfigurationException {

    configuration = pConfiguration;
  }

  @Override
  public void process(HttpContext context) throws ActionProcessingError {
    if
(!configuration.getHeaderValue().equals(context.getClientRequest().getHeader(confi
guration.getHeaderKey()))) {
      throw new ActionProcessingError(HTTP_INTERNAL_ERROR, "Required Header value
not matching value");
    }
  }

}
```

The **HeaderValidationActionConfig.java** file:

```
/* Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved. */
package oracle.sdp.daf.action;

import javax.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 * The configuration file for the action.
```

```
 */
@XmlRootElement
public final class HeaderValidationActionConfig implements Serializable {

  /**
   * UUID.
   */
  private static final long serialVersionUID = 1L;

  private String headerKey;
  private String headerValue;

  public void setHeaderKey(String pHeaderKey) {
    headerKey = pHeaderKey;
  }

  public String getHeaderKey() {
    return headerKey;
  }

  public void setHeaderValue(String pHeaderValue) {
    headerValue = pHeaderValue;
  }

  public String getHeaderValue() {
    return headerValue;
  }

}
```

Example Black List Action

This example action allows you to create a list of IP addresses that are prohibited from sending requests to an API.

The **BlackListAction.java** file:

```
/* Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved. */
package oracle.sdp.daf.action;

import oracle.sdp.daf.action.api.Action;
import oracle.sdp.daf.action.DafAction;
import oracle.sdp.daf.action.FlowRestriction;
import oracle.sdp.daf.action.api.ActionConfigurationException;
import oracle.sdp.daf.action.api.ActionProcessingError;
import oracle.sdp.daf.action.api.HttpContext;
import oracle.sdp.daf.config.ApiConfiguration;

/**
 * BlackList action.
 */
@DafAction(name = "BlackList", configBean = BlackListActionConfig.class)
public final class BlackListAction implements Action<BlackListActionConfig> {

  private static final int HTTP_FORBIDDEN = 403;

  private BlackListActionConfig configuration;

  @Override
  public void init(BlackListActionConfig pConfiguration, ApiConfiguration
apiConfiguration)
```

```
    throws ActionConfigurationException {

    configuration = pConfiguration;
  }

  @Override
  public void process(HttpContext context) throws ActionProcessingError {
    if
(configuration.getAddress().equals(context.getClientRequest().getRemoteHost())) {
      throw new ActionProcessingError(HTTP_FORBIDDEN, "BlackListed!");
    }
  }

}
```

The **BlacklistActionConfig.java** file:

```
/* Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved. */
package oracle.sdp.daf.action;

import javax.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 * The configuration file for the action.
 */
@XmlRootElement
public final class BlackListActionConfig implements Serializable {

  /**
   * UUID.
   */
  private static final long serialVersionUID = 1L;

  private String address;

  public void setAddress(String pAddress) {
    address = pAddress;
  }

  public String getAddress() {
    return address;
  }

}
```

Example Action for Adding a New Key/Value to EDRs

This code snippet for a custom action adds a new key/value pair to an EDR:

```
com.bea.wlcp.wlng.api.plugin.context.RequestContextManager.getCurrent().putEdr("ne
w_key", "new_value")
```

5

Old Partner Relationship Management Interfaces

The Oracle Communications Services Gatekeeper pre-6.0 release Partner Relationship Management (PRM) module consists of these interfaces:

- The Service Provider interface set
- The Operator interface set

> **Note:** This chapter describes the interfaces for the older SOAP-based APIs. To extend and customize the GUIs used in older (Pre-6.0) Services Gatekeeper releases, see "Partner Relationship Management SOAP-based Web Services Interface".
>
> See "API Management REST-based API" for instructions on how to use the newer API Management API.

The Service Provider interface set contains a subset of the functionality of the Operator set. It can be used to perform operations for only one specific Service Provider Account; the one with which the user of the customer relationship management (CRM)/PRM application has authenticated. The Operator set can perform operations on all Service Provider Accounts and also has access to alarm reports. All operations are synchronous.

The Service Provider set of interfaces consists of the following groups:

- Service Provider Service Interface, see "Service Provider Service Interfaces".
- Service Provider call details record (CDR) Utility interface, see "Service Provider CDR Utility Interface".
- Operator Statistics Utility Interfaces, see "Service Provider Statistics Utility Interface".
- Service Provider Login interface, see "Service Provider Login Interface".

> **Note:** The login portion of this interface is provided only for backward compatibility purposes. The current implementation is sessionless and the only use for this interface in the current implementation is to request a new Service Provider Account be set up.

The Operator set of interface consists of the following groups:

- Operator Service Interface, see "Operator Interfaces".
- Operator Alarm interface, see "Operator Alarm Utility Interface"
- Operator CDR Utility interface, see "Operator CDR Utility Interface".
- Operator Statistics Utility Interfaces, see "Operator Statistics Utility Interface"
- Operator Login interface, see "Operator Login Interface".

> **Note:** This interface is only supplied for backward compatibility. The current implementation is sessionless.

WSDLs

The following is a list of the Web Services interfaces available for integration with CRM/PRM applications. The interfaces use document/literal encoding and assume Simple Object Access Protocol (SOAP) over HTTP.

The Web Service Definition Language (WSDL) files that correspond to the interface sets can be found at the URIs listed below:

> **Note:** In each of the example URIs, the values *host* and *port* are dependent upon your Services Gatekeeper deployment.

- **SpService**:

  `http://host:port/prm_sp/services/SpService?wsdl`

  For more information, see "Service Provider Interfaces".

- **SpCdrUtil**:

  `http://host:port/prm_sp/services/SpCdrUtil?wsdl`

  For more information, see "Service Provider CDR Utility Interface".

- **SpStatUtil**:

  `http://host:port/prm_sp/services/SpStatisticsUtil?wsdl`

  For more information, see "Service Provider Statistics Utility Interface".

- **SpLogin**:

  `http://host:port/prm_sp/services/SpLogin?wsdl`

  For more information, see "Service Provider Login Interface".

  > **Note:** The login portion of this interface is provided only for backward compatibility purposes. The current implementation is sessionless and the only use for this interface in the current implementation is to request a new Service Provider Account be set up.

- **OpService**:

  `http://host:port/prm_op/services/OpService?wsdl`

For more information, see "Operator Interfaces".

- **OpAlarmUtil**:

  `http://host:port/prm_op/services/OpAlarmUtil?wsdl`

  For more information, see "Operator Alarm Utility Interface".

- **OpCdrUtil**:

  `http://host:port/prm_op/services/OpCdrUtil?wsdl`

  For more information, see "Operator CDR Utility Interface".

- **OpStatUtil**:

  `http://host:port/prm_op/services/OpStatisticsUtil?wsdl`

  For more information, see "Operator Statistics Utility Interface".

- **OpLogin**:

  `http://<host>:<port>/prm_op/services/OpLogin?wsdl`

  For more information, see "Operator Login Interface".

Note: This interface is only supplied for backward compatibility. The current implementation is sessionless.

Service Provider Interfaces

Service Provider interfaces allow application service providers to request changes in their account and to monitor their accounts activities

Service Provider Service Interfaces

The Service Provider Service interface provides ways to interact with the following entities:

- Management User
- Service Provider Accounts
- Application Account
- Application Instances

Management User

The management user account authenticates the service provider with the PRM system. The following operation can be performed on an ongoing management user account:

- Change the password for the Service Provider Account

Service Provider Accounts

The following operations can be performed on Service Provider Accounts:

- Activate
- Deactivate
- Request deletion

- Request update
- Get information
- Get SLA
- Get state

Application Account

The following operations can be performed on an Application Account:

- Register new
- Activate
- Deactivate
- Request deletion
- Update
- Get information
- List current Application Accounts
- Get the SLA
- Get the state

Application Instances

The following operations can be performed on an Application Instance:

- Register new
- Activate
- Deactivate
- Request deletion
- Request update
- Get information
- Get the state
- List current Application Instances
- Set the password that the Application Instance uses when accessing Services Gatekeeper.

Service Provider CDR Utility Interface

The following operations are available:

- Count the number of charge data records (CDRs) that have been generated
- List the CDRs that have been generated

Service Provider Statistics Utility Interface

The following operations are available:

- Retrieve generated statistics information.
- List available statistics types.

Service Provider Login Interface

- backward compatibility only:
  - Login
  - Logout
- Request a new Service Provider Account

Operator Interfaces

Operator interfaces allow operators to approve changes in Service Providers' accounts and to perform other account maintenance tasks.

Operator Service Interfaces

The Operator Service interface provides ways to interact with the following entities:

- Management User
- Service Provider Account
- Service Provider Group
- Application Account
- Application Account Group
- Application Instance

Management User

The following operations can be performed on a management user account:

- Change password for the Operator Account
- Get user level for an Operator Account
- Change password that the Service Provider uses with PRM-SP

Service Provider Account

The following operations can be performed on a Service Provider Account:

- Approve or disapprove service provider's request to register a new Service Provider Account
- Approve or disapprove service provider's request to delete an existing Service Provider Account
- Approve or disapprove service provider's request to update an existing Service Provider Account
- Activate
- Deactivate
- Connect (move) an account into a Service Provider Group
- Make an operator request to delete an existing Service Provider Account
- Make an operator request to update an existing Service Provider Account
- Make an operator request to register a new Service Provider Account

- Get a list of accounts that have pending requests, and need approval or disapproval by the operator
- Get Service Provider Account information
- List all Service Provider Accounts
- Get Service Provider Account state

Service Provider Group

The following operations can be performed on a Service Provider Group:

- Create
- Update
- Delete
- Get information about the group
- List all groups
- Get the Service Provider Group to which a specific Application Account is assigned

Application Account

The following operations can be performed on an Application Account:

- Approve or disapprove service provider's request to register a new Application Account
- Approve or disapprove service provider's request to update an existing Application Account
- Approve or disapprove service provider's request to delete an existing Application Account
- Get a list of accounts that have pending requests, and need approval or disapproval by the operator
- Activate
- Deactivate
- Connect (move) the account into an Application Account Group
- Get information about the account
- Get the account's state
- List all accounts
- Make an operator request to delete an existing account
- Make an operator request to register a new account
- Make an operator request to update an existing account

Application Account Group

The following operations can be performed on an Application Account group:

- Create
- Update
- Delete

- Get the Application Account Group to which a specific Service Provider Account and Application Account are assigned
- Get information about the group
- List all groups

Application Instance

The following operations can be performed on an Application Instance:

- Approve or disapprove a service provider's request to delete an existing Application Instance
- Approve or disapprove a service provider's request to register a new Application Instance
- Approve or disapprove a service provider's request to update an existing Application Instance
- Activate
- Deactivate
- Get a list of instances that have pending requests, and need approval or disapproval by the operator
- Get information about the instance
- Get Application Instance's state
- List all instances
- Set password to use when an application authenticates to Services Gatekeeper
- Make an operator request to delete an existing instance
- Make an operator request to register new Application Instance
- Make an operator request to update an existing instance
- Get the authenticated operator's user level

Operator Alarm Utility Interface

The following operations are available:

- Count the number of alarms that have been generated
- List the alarms that have been generated

Operator CDR Utility Interface

The following operations are available:

- Count the number of charging data records that have been generated.
- List the charging data records that have been generated.

Operator Statistics Utility Interface

The following operations are available:

- Retrieve generated statistics information.
- List available statistics types.

Operator Login Interface

The operator login interface is provided only for backward compatibility purposes.

A

API Management REST-based API

This appendix describes the RESTful API interface that Oracle Communications Services Gatekeeper uses to manage APIs and create the Services Gatekeeper PRM GUI tools. You can use this API for manage APIs, or to extend or replace the default Services Gatekeeper GUI tools.

Understanding the Partner Relationship Management API

You use the API Management API to:

- Manage APIs within Services Gatekeeper
- Modify or replace these GUI tools:
  - The API and Partner Manager Portal
  - Partner Manager Portal
  - Network Service Supplier Portal

The API Management API is the interface for creating, changing, and deleting the objects that define APIs, PRM partners and groups, and so on. Like any RESTful API, the API Management API makes its services available to client applications through simple HTTP requests. However, secure HTTP (HTTPs) is required for all operations in this API.

The Services Gatekeeper servers must be up and running to access this API.

Several types of clients connect to Services Gatekeeper, including the Administration Console client process and the portal GUIs. Services Gatekeeper listens for API Management API at the port configured for client HTTP access. Before using this API, ensure that the managed server is configured to accept web requests.

Creating and Updating Objects

To add partners, groups, and APIs, you use the API Management API to create the data objects that represent the entities you want to add.

The body of PUT or POST requests should contain JSON or XML-formatted data that describes the entity, such as a partner's name and contact information.

To update an object, specify the new value for a parameter in the body of the PUT or POST request. You only need to supply the new or changing parameter values to modify and not for the entire object.

The message body is either JSON or XML object depending on the Content-Type header field setting.

Understanding API Management API Error Handling

When an API operation succeeds, it returns a response with HTTP status code 200.

If an error occurs, these operations return a response that has an HTTP code reflecting the nature of the error. These error codes include:

- 400 - Request not accepted because a request value is not acceptable. Details are in the response body.
- 401 - Username or password incorrect.
- 404 - Resource/URL does not exist.
- 405 - Method not allowed.
- 406 - Request method not supported by the URL.
- 500 - Internal system error.
- 503 - Service unavailable.

API Management API Operations

The following sections list the RESTful operations of the API Management API.

API and Partner Manager Portal Operations

The API and Partner Manager Portal GUI uses the operations listed in this section.

API Management Operations

These operations manage your APIs:

- getAPIs
- getAPI
- createAPI
- editAPI
- editAPIContextRoot
- deleteAPI
- updateAPIStatus
- listApplicationsForAPI
- listAPILifeCycle
- updateAPIApplicationRelationship

Group Management Operations

These operations manage both your groups and accounts:

- listAllGroups
- createServiceProviderGroup
- deleteGroup
- confirmMovePartnerToGroup

Partner Management Operations

These operations manage your partners:

- getUsers
- getUserByName
- approve
- reject
- deleteUser
- createUser

Application Management Operations

These operations manage your applications:

- listApplications
- getApplication
- updateCurrentSlaForApprove
- denyApplication

System Configuration Operations

These operations manage the configuration settings:

- getAllSysConfig
- getSysConfig
- updateAllSysConfig
- updateSysConfig
- getBlobSysConfig
- updateBlobSysConfig

Partner Portal Operations

The Partner Manager Portal GUI uses the operations listed in this section.

Account Management Operations

These operations manage your GUIs.

- registerSP
- editUser
- getUserByName

Application Management Operations

These operations manage your applications:

- listApplications
- createApplication
- updateApplication
- removeApplication

- removePendingApp

Network Service Supplier Portal Operations

The Network Service Supplier GUI uses these operations.

Network Interface Management Operations

These operations manage your network interfaces:

- getInterfaceList
- getInterfaceInfoByName
- updateInterface
- removeInterfaceByID

Understanding the API Management API Objects

Table A–1 Lists the **apiObject** parameters, which Services Gatekeeper uses to define an API.

Table A–1 apiObject Parameters

Parameter	Type	Description
accessType	String	One of: ■ **HTTP** ■ **HTTPS** ■ **BOTH**
accessUrl	URI	The URI used to access the object.
apiAuthTypes	String List	The authorization method used to protect the API. The options are: ■ **NONE** - Unauthenticated traffic is allowed. ■ **TEXT** - Use BasicAuthentication to authorize REST-based requests or **UsernameToken** to authorize SOAP-based requests. ■ **APPKEY** - Use an application key, defined by the **AppKeyAuthentication** action, for authentication. Use **AppKeyAuthenticaiton** to specify a key and/or query parameter for authentication. ■ **OAUTH** - Use Oauth 2.0 authentication. **TEXT** and **OAUTH** can be listed and used together. An empty list uses the **NONE** option, which represents security vulnerability. Use only with API traffic from trusted sources.
apiDisplayName	String	Optional. The value for the API **Name** field in the Partner and API Management Portal. The name of the API displayed in the Portal. The value for this field is also used for the apiName parameter (Context Root field) unless you change it.
apiId	String	The objects unique identifier generated by Services Gatekeeper.
apiInterfaces	apiInterface Object	API interface field list. See Table A–2, " apiInterface Parameters" for details.

Table A–1 (Cont.) apiObject Parameters

Parameter	Type	Description
apiName	String	The value for the **Context Root** field in the Partner and API Management Portal. By default the Portal uses the value for **apiDisplayName** in this field unless you change it. The value for this field is included in the API Access URL (visible by users) and sometimes used to convey information.
apiVersion	String	Optional. A version number for the API.
authToken	String	The authorization token for TEXT authorization. For the "createAPI" and "editAPI" operations, this password can be 32-byte encrypted with the AES key you set in "Encrypt Application Passwords."
authType	String	The application-facing authentication type. One of: ▪ **NONE** ▪ **TEXT** ▪ **OAUTH**
contractname	String	Reserved for future use.
contractversion	String	Reserved for future use.
csServiceType	String	The communication service. Used when **serviceType** is **by-ocsg-cs**, which indicates that the API uses an existing Services Gatekeeper communication service.
csOption	csOption	The services model for a communication service. Used when **serviceType** is **by-ocsg-cs**, which indicates that the API uses an existing Services Gatekeeper communication service. The options are: ▪ DAF - When **Service Mode** is **Dynamic** (Allows Editing Exposure and Action Flow). ▪ OCSG - When **Service Mode** is **Static.**
description	String	An informal description of the API
direction	String	One of: ▪ **AOMT** - Application Originated Mobile Terminated ▪ **MOAT** - Mobile Originated Application Terminated
facade	String	Either **REST** or **SOAP**.
groups	String List	If the value for the **privilege** parameter is **1** (private API), returns a list of the groups allowed. The syntax is: **["***group1***","***group2***"]**
icon	String	The relative path to the API icon graphic.
link	String	A link to the API documentation.
networkAuthorizationURI	URL	The network-facing authorization server URL.
networkClientRedirectURI	URL	The network-facing client redirection URL.
networkProxy	String	The network proxy in the form: *IP_address***:***port*
networkTokenkURI	URL	The network-facing token URL.
northBoundWadlFiles	WADL file	The WADL **fileName** and **fileContent** strings for application-facing WADL files

Table A–1 (Cont.) apiObject Parameters

Parameter	Type	Description
privilege	Integer	One of: ■ **0** - Specifies a public API which can be used by all groups. ■ **1** - Specifies a private API which can only be used by specific partners.
protocol	String	Usually the network URL. If the **serviceType** value is **by-registered**, then us the network interface ID.
serviceType	String	One of: ■ **by-url** - Uses and existing URL ■ **by-file** - Uses a WSDL or WADL file ■ **by-registered** - Uses an existing registered network service. ■ **by-ocsg-cs** - Uses an existing Services Gatekeeper communication services.
status	String	The object's status. Can be: CREATED, PUBLISHED, SUSPENDED, DEPRECATED, or RETIRED.
wadlfiles	WADL file	The WADL **fileName** and **fileContent** strings for network-facing WADL files.

Table A–2 lists the **apiInterface** object parameters.

Table A–2 apiInterface Parameters

Parameter	Type	Description
name	String	The interface name
displayName	String	The display name.
fileLocation	String	The URL of the file.
apiMethods	apiMethod object	A list of the apiMethod parameters. See Table A–3, " apiMethod Parameters" for details.

Table A–3 lists the **apiMethod** object parameters.

Table A–3 apiMethod Parameters

Parameter	Type	Description
changeCode	String	Reserved for future use.
displayName	String	The public name for the exposed resource.
expose	Boolean	True if the resource should be exposed, and false if not.
httpVerb	String	For application-facing REST requests, one of: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH. For application-facing SOAP messages, the SOAP method name.
name	String	The resource name.
path	URL	The path to the application resource.
parameterSetting	paramSettin g	Reserved for future use.

Table A–3 (Cont.) apiMethod Parameters

Parameter	Type	Description
servicePath	String	The path to the network service.
serviceHttpVerb	String	For network-facing REST messages, one of: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH. For network-facing SOAP requests, the SOAP method name.
settlementCode	String	Reserved for future use.
spslas	spSAL	Reserved for future use.

createAPI

Create a new API object for use with Services Gatekeeper.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/createAPI
```

Request Body

This operation uses these request parameters:

- **createAPI** (**apiObject**) Required. See Table A–1, " apiObject Parameters" for details on the **apiObject** parameters. All parameters are required, and the **authToken** parameter accepts 24 and 32-byte encryption.

Response Body

This operation does use any response parameters.

Example

Example A–1 createAPIs Request Example

```
POST https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/createAPI
{"createAPI":{"apiObject":{"apiDisplayName":
"HDAPI002ContextRoot""apiName": "HDAPI002ContextRoot","apiVersion":
"1","apiAuthTypes": { "TEXT" },"accessUrl":
"https://localhost:8081/api-0919-11/1","apiInterfaces": [{"name":
"OneAPI SMS-v1.1","apiMethods": [{"name": "createOutboundMessage","displayName":
"createOutboundMessage","path": "/outbound/{senderAddress}/requests","httpVerb":
"POST","servicePath": "/outbound/{senderAddress}/requests","serviceHttpVerb":
"POST","expose": true}, {"name": "createOutboundSubscription","displayName":
"createOutboundSubscription","path":
"/outbound/{senderAddress}/subscriptions","httpVerb": "POST","servicePath":
"/outbound/{senderAddress}/subscriptions","serviceHttpVerb": "POST","expose":
false}, {"name": "deleteOutboundSubscriptionById","displayName":
"deleteOutboundSubscriptionById","path":
"/outbound/subscriptions/{subscriptionId}","httpVerb": "DELETE","servicePath":
"/outbound/subscriptions/{subscriptionId}","serviceHttpVerb": "DELETE","expose":
false}, {"name": "getOutboundMessageDeliveryInfoById","displayName":
"getOutboundMessageDeliveryInfoById","path":
"/outbound/{senderAddress}/requests/{requestId}/deliveryInfos","httpVerb":
"GET","servicePath":
"/outbound/{senderAddress}/requests/{requestId}/deliveryInfos","serviceHttpVerb":
"GET","expose": false}],"displayName": "OneAPI SMS-v1.1"}],"description":
```

```
"kkkkkkkkkkkkkkkkkkkkkkkkkkkkkk","facade": "REST","seviceType":
"by-registered","protocol": "af4b3707-e4e3-4836-bf6d-df7f8bcd2e12","privilege":
1,"link": "https://1.com","accessType": "HTTP","authType": "TEXT","authToken":
"jack:apache","groups": ["aaaa"],"direction": "AOMT","networkAuthorizationURI":
"authuri","networkTokenURI": "tokenuri","networkClientRedirectURI":
"redirect","icon":
"icon","wadlFiles":[{"fileName":"aaaa","fileContent":"aaaa"},{"fileName":"bbbb",
"fileContent":"bbbb"}],"northBoundWadlFiles":[{"fileName":"north-aaaa",
"fileContent":"north-aaaa"},{"fileName":"north-bbbb","fileContent":"north-bbbb"}]}
}}
```

Example A–2 createAPI Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

editAPI

Change fields on an **apiObject** object.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/editAPI
```

Request Body

This operation uses these request parameters:

- **editAPI** (**apiObject**) Required. See Table A–1, " apiObject Parameters" for details on the **apiObject** parameters. The **authToken** parameter accepts 24 and 32-byte encryption.

Response Body

This operation does not use any response parameters.

Examples

Example A–3 editAPIs Request Example

```
POST https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/editAPI
{"editAPI":{"apiObject":{"apiId":"ba32cbb8-d97e-4d87-bf7c-0ca1bdd1d3e7",
"apiDisplayName":"api-0925-1","apiName":"api-0925-1","apiVersion":"1","accessUrl":
"https://localhost:
8081/daf/api-0925-1/1","apiInterfaces":[{"name":"api-0925-1-v1","apiMethods":
[{"name":"getLocation","displayName":"getLocation","path":"/getLocation/
{subscriber}","httpVerb":"GET","servicePath":"/getLocation/{subscriber}",
"serviceHttpVerb":"GET","expose":true}],"displayName":"api-0925-1-v1",
"fileLocation":"https://1.com"}],"wadlFiles":[],"northBoundWadlFiles":[],
"description":"api-0925-1 description hahahaha","facade":"REST",
"direction":"AOMT","seviceType":"by-url","protocol":"","privilege":0,"link":
"https://1.com","accessType":"HTTP","authType":"NONE","authToken":"","n
```

Example A–4 editAPIs Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

editAPIContextRoot

Change the context root (**apiName** field) of an API object.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/editAPIContextRoot
```

Request Body

This operation uses these request parameters:

- **apiId** (String) Mandatory. The unique API object identifier from the API Object see Table A–1, " apiObject Parameters".
- **contextRoot** (String) Mandatory. The new value to use for the **apiName** field in the API object.

Response Body

This operation does use any response parameters.

Examples

Example A–5 editAPIContextRoot Request Example

```
POST /prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/editAPIContextRoot HTTP/1.1
Host: 192.0.0.24:8001
Accept: application/json
Content-Type: application/json
Authorization: Basic b3A6d2VibG9naWMxMjM
Cache-Control: no-cache

{"editAPIContextRoot":{"apiId":"ID-TestAPI3",
"contextRoot":"TestAPI3Root2/SubRoot"}}
```

Example A–6 editAPIContextRoot Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"editAPIContextRootResponse":{}}
```

deleteAPI

Delete an **apiObject** object.

Authorization

Partner Manager Administrator

HTTP Method

DELETE

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/deleteAPI/apiName
```

Request Body

This operation uses these request parameters:

- **apiName** (String) Mandatory. The API name from an **apiObject** object. Table A–1, " apiObject Parameters" lists the **apiObject** parameters.

Response Body

This operation does use any response parameters.

Examples

Example A–7 deleteAPIs Request Example

```
DELETE https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/deleteAPI/
{"deleteAPI":{"apiName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8"}}
```

Example A–8 deleteAPIs Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

updateAPIStatus

Change the status parameter of an **apiObject**. See Table A–1, " apiObject Parameters" for a list of the values allowed for the **status** parameter.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/updateApiStatus
```

Request Body

This operation uses these request parameters:

- **apiName** (String) Mandatory. The **apiName** parameter of the **apiObject**.
- **apiVersion** (String) Mandatory. The **apiVersion** parameter of the **apiObject** object.
- **status** (String) Mandatory. The new **status** parameter to apply to the **apiObject**. See Table A–1, " apiObject Parameters" for a list of the values allowed for the **status** parameter.

Response Body

This operation does not return any parameters.

Examples

Example A–9 updateAPIStatus Request Example

```
POST https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/updateApiStatus
{"updateApiStatus":{"apiDisplayName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8",
"apiName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8",
"apiVersion":"1","status":"PUBLISHED"}}
```

Example A–10 updateAPIStatus Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getAPIs

Retrieve a list of **apiObjects** which represent APIs.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/getAPIs
```

Request Body

This operation does not use any request parameters:

Response Body

This operation returns these parameters:

- **getAPIsResponse** (**apiObject** object list). Table A–1, " apiObject Parameters" lists the **apiObject** parameters.

Examples

Example A–11 getAPIs Request Example

```
GET /prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/getAPIs
```

Example A–12 getAPIs Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getAPIsResponse": {"return": [{"apiId":
"32d03e51-1d57-43e2-9077-37cef5712750","apiDisplayName": "0919-1""apiName":
"0919-1","apiVersion": "1","status": "CREATED","accessUrl":
"https://localhost:8001/daf/0919-1/1","apiInterfaces": [{"name":
"Payment-v1.1","displayName": "Payment-v1.1","fileLocation":
"https://doc.payment.access.url/v11"}],"description": "api description","facade":
"REST","seviceType": "by-registered","protocol":
"b421bded-0974-4045-81cf-0cfb8a01c538","privilege": 0,"link": "https://1.com",
"accessType": "HTTP","authType": "TEXT","authToken": "jack:apache","direction":
"AOMT"}]}}
```

getAPI

Retrieve an API object based on it's **apiId** parameter.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/getAPI/apiName
```

Request Body

This operation uses these request parameters:

- **apiName** (String) Required. The value of the **apiName** parameter of the API's **apiObect** . See Table A–1, " apiObject Parameters" for details on this object.

Response Body

This operation returns these parameters:

- **getAPIsResponse** (API object list) - A list of the API object parameters. See Table A–1, " apiObject Parameters" for the list.

Examples

Example A–13 getAPI Request Example

```
GET https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/getAPI/
{"apiId": "d7f1068b-906f-4a8f-829a-2e7858ed07c9"
```

Example A–14 getAPI Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getAPIResponse": {"return": {"apiId":
 "d7f1068b-906f-4a8f-829a-2e7858ed07c9","apiDisplayName": "pjhweather""apiName":
"yourWeather","apiVersion":
"1","status": "DEPRECATED","accessUrl":
 "https://127.0.0.1/pjhweather/1","apiInterfaces": [{ "name": "pjhweather-v1",
"apiMethods": [{ "name": "weather", "displayName": "weather", "path": "weather",
 "httpVerb": "GET", "servicePath": "forecastrss", "serviceHttpVerb": "GET",
"expose": true }], "displayName": "pjhweather-v1", "fileLocation":
 "https://www.pjh.com/weather" }],"description": "pjh weather","facade":
"REST","seviceType": "by-url","protocol":
 "https://weather.yahooapis.com/","privilege": 1,"link":
"https://www.pjh.com/weather","accessType": "HTTP","authType": "NONE","groups":
["default_sp_group"],"direction": "AOMT"}}}
```

listApplicationsForAPI

List all applications using an API.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/listApplicationsForAPI/apiName
```

Request Body

This operation uses these request parameters:

- **apiName** (String) Mandatory. The **apiName** parameter from an **apiObject**. Identifies the API you want a list of applications for.

Response Body

This operation returns these parameters:

- **ListApplicationsForAPIResponse** (application list). A list of the applications currently using the API.

Examples

Example A–15 listApplicationsForAPI Request Example

```
GET https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_manager
/api/PartnerManagerApi/listApplicationsForAPI/apiName
{"apiName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8"}
```

Example A–16 listApplicationsForAPI Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"ListApplicationsForAPI":{"return":[{"applicationID":"fa0183a3-1a57-4acb-
9aac-1846b54cb18b","applicationName":"app-0828-1","partnerName":"guest1",
"partnerCompany":"oracle","description":"app-0828-1app-0828-1app-0828-1app-
0828-1app-0828-1app-0828-1app-0828-1app-0828-1","applicationAPIs":[{"apiDisplayNam
e":"api-0825-1","apiName":
"api-0825-1","accessURL":"https://localhost:8001/api-0825-1/1","apiVersion":"1",
"apiDescription":"api-0825-1api-0825-1api-0825-1api-0825-1",
"applicationMethodSLAs":[{"methodName":"","interfaceName":
"9380cdb2-c584-42ca-929f-761f216f0e69","quota":{"days":0,"limitExceedOK":
false,"qtaLimit":0},"rate":{"reqLimit":0,"timePeriod":0},"methodGuarantee":
{"reqLimitGuarantee":0,"timePeriodGuarantee":0}}],"needReadContract":false}],
"trafficUser":"rotterzeng_app-0828-1","trafficPassword":"guest1_
app-0828-1","submitDate":"2014-08-28+08:00","effectiveFrom":"2014-08-01+08:00",
```

"effectiveTo":"2014-08-31+08:00","status":"ACTIVE","lockStatus":"UNLOCKED",
"quota":{"days":1,"limitExceedOK":true,"qtaLimit":1},"rate":{
"reqLimit":1,"timePeriod":1}}]}

listAPILifeCycle

List the API life cycle for an API.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/listAPILifeCycle/apiName/apiVersion
```

Request Body

This operation uses these request parameters:

- **apiName** (String) Mandatory. A valid **apiName** parameter of an **apiObject**. Table A–1, " apiObject Parameters" lists the **apiObject** parameters.
- **apiVersion** (String) Mandatory. A valid **apiVersion** parameter for an **apiObject**. Table A–1, " apiObject Parameters" lists the **apiObject** parameters.

Response Body

This operation uses these response parameters:

- **id** (String). A valid record ID value of an **apiObject**.
- **apiName** (String). A valid **apiName** value of an **apiObject**.
- **apiVersion** (String). A valid **apiVersion** for the **apiObject**.
- **operator** (String). The name of the operator who last updated the **apiObject** status.
- **date** (String). The date when the status was last updated.
- **content** (String). The value for the status that was changed.

Examples

Example A–17 listAPILifeCycle Request Example

```
GET https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/listAPILifeCycle/
{"listAPILifeCycle":{"apiName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8",
"apiVersion":"1","status":"PUBLISHED"}}
```

Example A–18 listAPILifeCycle Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"ListAPILifeCycleResponse":{"return":[{"id":"9471f463-8c74-42c0-9832-
a73ebb060404","apiDisplayName":"api-0925-1","apiName":"api-0925-1","apiVersion":"1
```

","operator":"op",
"date":"09/25/2014 16:59:37","content":"Created"},
{"id":"8c97e48b-3251-45d8-8b6e-dd1b16fbd369","apiName":"api-0925-1","apiVersion"
:"1","operator":"op","date":"09/25/2014 16:44:31","content":"Created"}]}}

updateAPIApplicationRelationship

Change the suspense state of an application-API relationship.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

`/prm_pm_rest/services/prm_pm/services/partner_manager/api/PartnerManagerApi/updateAplicationrelationship/`*`apiName`*`/`*`apiVersion`*`/`*`suspended`*

Request Body

This operation uses these request parameters:

- **apiName** - (String). A valid **apiName** value (context root) of an **apiObject**.
- **applicationID** - (String). A valid **apiVersion** for the **apiObject**.
- **suspended** - (Integer). **0** allows the API to handle all traffic for the application; **1** prevents the API from processing any traffic for the application.

Response Body

This operation does not return any response parameters.

Examples

Example A–19 updateApplicationRelationship Request Example

```
POST https://10.182.98.78:9001/apis
/prm_pm_rest/services/prm_pm/services/partner_
manager/api/PartnerManagerApi/listAPILifeCycle/
{"updateAPIApplicationRelationship":{"apiName":"c305f2a5-8667-40a6-8510-
3d4627e96710
","applicationID":"ccb5755d-0cb1-4452-a5a3-6c970b014307","suspended":"0"}}
```

Example A–20 updateApplicationRelationship Response Example

```
HTTP/1.1 200 OKContent-Length: 0Server: Jetty(8.0.1.0)Content-Length: 0
```

listAllGroups

This operations returns a list of all **serviceProviderGroup** objects.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/partner_
manager/group/PartnerManagerSlaGroup/listAllGroups
```

Request Body

This operation does not use any request parameters.

Response Body

This operation returns a **serviceProviderGroup** object and the associated **quota** and **rate** objects. Table A–4, " serviceProviderGroup Object Parameters" lists the **serviceProviderGroup** object parameters.

Table A–4 serviceProviderGroup Object Parameters

Parameter	Data Type	Description
group	string	The group name.
quota	**quota** object	Defines the maximum number of messages that a subscriber can send during a quota period. Table A–5, " quota Object Parameters" lists the **quota** object parameters.
rate	rate object	Defines the maximum number of messages allowed during the quota period. Table A–6, " rate Object Parameters" lists the **rate** object parameters.
totalPartners	Integer	The number of partners that belong to the group.

Table A–5 quota Object Parameters

Parameter	Data Type	Descriptions
days	Integer	The number of days the quota is valid.
limitExceedOK	Boolean	Whether the user can exceed the quota.
qtaLimit	Integer	The maximum number of messages allowed during the quota.

Table A–6 rate Object Parameters

Parameter	Data Type	Description
reqLimit	Integer	The maximum number of messages allowed furing the quota period.

Table A–6 (Cont.) rate Object Parameters

Parameter	Data Type	Description
timePeriod	Integer	The time period in seconds.

Examples

Example A–21 listAllGroups Request Example

```
GET https://10.182.98.78:9001/groups
/prm_pm_rest/services/partner_manager/group/PartnerManagerSlaGroup/listAllGroups
```

Example A–22 listAllGroups Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"listAllGroupsResponse":{"return":[{"group":"0903-1","quota":{"days":1,
"limitExceedOK":false,"qtaLimit":1},"rate":{"reqLimit":1,"timePeriod":1000},
"totalPartners":0},{"group":"0903-2","quota":{"days":1,"limitExceedOK":false,
"qtaLimit":1},"rate":{"reqLimit":1,"timePeriod":1000},"totalPartners":0},
{"group":"aaaa","quota":{"days":1,"limitExceedOK":false,"qtaLimit":1},"rate":
{"reqLimit":1,"timePeriod":1000},"totalPartners":0},{"group":"default_sp_
group","quota":{"days":0,"limitExceedOK":false,"qtaLimit":0},"rate":
{"reqLimit":0,"timePeriod":0},"totalPartners":1},{"group":"group0903",
"quota":{"days":1,"limitExceedOK":false,"qtaLimit":1},"rate":{"reqLimit":1,
"timePeriod":1000},"totalPartners":0},{"group":"sxg_test",
"quota":{"days":1,"limitExceedOK":false,"qtaLimit":50},"rate":{"reqLimit":10,
"timePeriod":1},"totalPartners":0},{"group":"sxh","quota":{"days":7,
"limitExceedOK":false,"qtaLimit":20000},"rate":{"reqLimit":1000,"timePeriod":
1000},"totalPartners":1},{"group":"sysdefault_sp_group",
"quota":{"days":0,"limitExceedOK":false,"qtaLimit":0},"rate":{"reqLimit":0,
"timePeriod":0},"totalPartners":0}]}}
```

createServiceProviderGroup

Creates a service provider group.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/partner_
manager/group/PartnerManagerSlaGroup/createServiceProviderGroup
```

Request Body

This operation uses these request parameters:

- **groupName** (String) required. The name of the group to create.
- **rate** (**rate** object) required. The maximum number of messages allowed during a quota period. Table A–5, " quota Object Parameters" lists the **rate** object parameters.
- **quota** (**quota** object) required. The maximum number of messages allowed during the quota period. Table A–6, " rate Object Parameters" lists the **quota** object parameters.

Response Body

This operation does not return any parameters:

Examples

Example A–23 createServiceProviderGroup Request Example

```
GET https://10.182.98.78:9001/apis
{"createServiceProviderGroup":{"groupName":"VIP","rate":{"reqLimit":
"10000","timePeriod":1000},"quota":{"qtaLimit":"1000000","days":"5"}}}
```

Example A–24 createServiceProviderGroup Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

deleteGroup

Deletes a service provider group.

Authorization

Partner Manager Administrator

HTTP Method

DELETE

URI

`/prm_pm_rest/services/partner_manager/group/PartnerManagerSlaGroup/deleteGroup/`*groupName*

Request Body

This operation uses these request parameters:

- **groupName** (String) Required. The name of the **group** object to delete.

Response Body

This operation does not return any parameters.

Examples

Example A–25 deleteGroup Request Example

```
DELETE https://10.182.98.78:9001/apis
/prm_pm_rest/services/partner_manager/group/
PartnerManagerSlaGroup/deleteGroup/testgroup1
```

Example A–26 deleteGroup Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

confirmMovePartnerToGroup

Assigns a partner to a group.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/partner_
manager/group/PartnerManagerSlaGroup/confirmMovePartnerToGroup
```

Request Body

This operation uses these request parameters:

- **partnerName** (String) Required. The name of the partner to associate with the group.
- **newGroupName** (String) Required. The new group name.
- **action** (String) Required. An action. Either **CHANGE_APP** or **EXPAND_SLA**.

Response Body

This operation does not return any parameters.

Examples

Example A–27 confirmMovePartnerToGroup Request Example

```
POST https://10.182.98.78:9001/apis
{"confirmMovePartnerToGroup":{"partnerName":"joechin33","newGroupName":
"VIP","action":"EXPAND_SLA"}}
```

Example A–28 confirmMovePartnerToGroup Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getUsers

Retrieves the **userInfo** object parameter for all partners.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/accountmanage/AccountManagement/getUsers
```

Request Body

This operation does not send any request parameters.

Response Body

This operation returns these parameters:

- **getUserResponse** (**userInfo** object). The **userInfo** parameters for each user. Table A–7, " userInfo Object Parameters" lists the **userInfo** parameters.

Examples

Example A–29 getUsers Request Example

```
GET https://10.182.98.78:9001/apis
/prm_pm_rest/services/accountmanage/AccountManagement/getUsers
```

Example A–30 getUsers Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getUsersResponse":{"return":[{"company":"Oracle","companyURL":"www.oracle.com","
emailAddr":"demouser@oracle.com","financial":{},"firstName":"Demo","lastName":
"User","password":"demouser","phone":"12345678","secureityAnswer":"Attention",
"secureityAnswerChoice":"0","status":2,"userName":"demouser","userType":
"PRM_SP"},{"city":"bj","company":"oracle","companyURL":
"https://www.oracle.com","contacts":[{"city":"","contactTimeFrom":"",
"contactTimeTo":"","country":"","emailAddress":"","firstName":"","lastName":""},
{"city":"","contactTimeFrom":"3:0:0","contactTimeTo":"6:0:0","country":"China",
"emailAddress":"helen@oracle.com","firstName":"","lastName":""}]}}
```

getUserByName

Retrieves the details for a single partner.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

`/prm_pm_rest/services/accountmanage/AccountManagement/getUsers/`*userName*

Request Body

This operation uses these request parameters:

- **userName** (String) Required. The name of the user to retrieve information for.

Response Body

This operation returns the **userInfo** object for the user. Table A–7, " userInfo Object Parameters" lists the **userInfo** parameters.

Examples

Example A–31 getUserByName Request Example

```
GET https://10.182.98.78:9001/users
/prm_pm_rest/services/accountmanage/AccountManagement/getUsers/testuser1
```

Example A–32 getUserByName Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getUserByNameResponse":{"return":{"company":"Oracle","companyURL":
"www.oracle.com","emailAddr":"demouser@oracle.com","financial":{},"firstName":
"Demo","lastName":"User","password":"demouser","phone":"12345678",
"secureityAnswer":"Attention","secureityAnswerChoice":"0","status":2,
"userName":"demouser","userType":"PRM_SP"}}}
```

approve

Approves a partner registration. When a partner or network service supplier successfully registers, a notification is sent to the partner manager for approval. This operation approves the registration.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/accountmanage/AccountManagement/approve
```

Request Body

This operation uses these request parameters:

- **userInfo** (**userInfo** object) Required. See Table A–7, " userInfo Object Parameters" for a list of the **userInfo** parameters.

Response Body

This operation does not return any parameters.

Examples

Example A–33 approve Request Example

```
POST https://10.182.98.78:9001/users
/prm_pm_rest/services/accountmanage/AccountManagement/approve
{"approve":{"userInfo":{"city":"bj","company":"oracle","companyURL":"https://
www.oracle.com","country":"China","emailAddr":"testuser@oracle.com","financial":
{"bankAccountNumber":"","bankAddress":"","bankName":"","bankRoutingNumber":"",
"city":"","country":"","invoiceTo":"","referenceAccount":"","stateOrProvince":
"","taxID":"","zipOrPostalCode":""},"firstName":"test","lastName":"test",
"password":"weblogic123","phone":"1-1-12345678","secureityAnswer":"testuser",
"secureityAnswerChoice":"0","status":0,"userName":"testuser","userType":
"PRM_SP","notificationId":"874add7a-9474-4698-b9a8-84440f64a7da"}}}
```

Example A–34 approve Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

reject

Rejects a partner registration.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/accountmanage/AccountManagement/reject
```

Request Body

This operation uses these request parameters:

- **userInfo** (**userInfo** object) Required. See Table A–7, " userInfo Object Parameters" for a list of the **userInfo** parameters.

Response Body

This operation does not return any parameters.

Examples

Example A–35 reject Request Example

```
GET https://10.182.98.78:9001/users
/prm_pm_rest/services/accountmanage/AccountManagement/reject
{"reject":{"userInfo":{"city":"bj","company":"oracle","companyURL":"https:
//www.oracle.com","country":"China","emailAddr":"testuser1@oracle.com",
"financial":{"bankAccountNumber":"","bankAddress":"","bankName":"",
"bankRoutingNumber":"","city":"","country":"","invoiceTo":"","referenceAccount":
"","stateOrProvince":"","taxID":"","zipOrPostalCode":""},"firstName":"test",
"lastName":"test","password":"weblogic123","phone":"1-1-12345678",
"secureityAnswer":"testuser","secureityAnswerChoice":"0","status":0,"userName":
"testuser1","userType":"PRM_SP"}}}
```

Example A–36 reject Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

deleteUser

Deletes a user's **userInfo** object.

Authorization

Partner Manager Administrator

HTTP Method

DELETE

URI

`/accountmanage/AccountManagement/deleteUser/`*userName*

Request Body

This operation uses these request parameters:

- **userName** (String) Required. The **userName** parameter of the **userInfo** object to delete. Table A–7 lists the **userInfo** object parameters.

Response Body

This operation does not return any parameters.

Examples

Example A–37 deleteUser Request Example

```
DELETE https://10.182.98.78:9001/users
/accountmanage/AccountManagement/deleteUser/{userName}
{"deleteUser":{userInfo":{userName":"testuser1"}}}
```

Example A–38 deleteUser Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

createUser

Creates a partner.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
services/accountmanage/AccountManagement/createUser
```

Request Body

This operation uses these request parameters:

- **userInfo** (userInfor object) Required. Table A–7, " userInfo Object Parameters" lists the **userInfo** object parameters. The **userInfo** object also includes contact objects. Table A–8, " contact Object Parameters" lists those parameters.

Table A–7 userInfo Object Parameters

Parameter	Data Type	Description
city	String	The city where the user lives.
company	String	The company the user works for.
companyURL	URL	A URL for the company
contacts	**contact** object	Contact information. Table A–8, " contact Object Parameters" lists the **contact** parameters.
country	String	The country where the user lives.
emailAddr	String	An email address in *xxx@xxx.xx* format.
firstName	String	The user's first name.
lastName	String	The user's last name.
password	String	The user password. For the "registerSP" operation, this password can be 32-byte encrypted with the AES key you set in "Encrypt Application Passwords."
phone	String	The user's phone number in \+?[0-9][0-9,\-,\s]{2,} format.
securityAnsswer	String	The answer to a security question.
securiytAnswerChoice	String	The user's choice of security questions.
stateOrProvince	String	The state or province where the user lives.
status	String	The user's status. Can be one of **active** or **registered**.
streetAddress	String	The user's street address
userName	String	The user's online username.

Table A–7 (Cont.) userInfo Object Parameters

Parameter	Data Type	Description
zipOrPostalCode	String	The user's zip code or other postal code.
slaGroup	String	The user's SLA group.
userType	String	The user type. Can be one of: PRM_SP, PRM_SS, or PRM_OP.

Table A–8 contact Object Parameters

Parameter	Data Type	Description
address	String	The street address for the contact.
city	String	The contact city.
contactTimeFrom	String	A time of day start time after which the contact can be contacted in HH:MM:SS format.
contactTimeTo	String	And time of day end time end time for the contact in HH:MM:SS format.
contactType	STring	A contact type value.
country	String	The contact's country.
emailAddress	String	The contact's email address
firstName	String	The contact's first name.
lastName	String	The contact's last name.
phone	phone list	The contact phone list.
preferredLanguage	String	The language the user wishes to use.
stateOrProvince	String	The contact state or province
title	String	A title for the contact.
zipOrPostalCode	String	A postal code for the contact

Response Body

This operation does not return any parameters.

Examples

Example A–39 createUser Request Example

```
POST https://10.182.98.78:9001/users
services/accountmanage/AccountManagement/createUser
{"createUser":{"userInfo":{"userName":"sxh_nss","userType":
"PRM_SS","password":"weblogic123","emailAddr":"sxh_
nss@oracle.com","phone":"1234567890",
"secureityAnswerChoice":0,"secureityAnswer":
"2","firstName":"rotter","lastName":"zeng","company":"oracle","companyURL":"https:
//123.com","stateOrProvince":"bj","zipOrPostalCode":"","streetAddress":"a","city":
"a","country":"Afghanistan","contacts":[{"city":"","country":"Afghanistan",
"emailAddress":"testuser@oracle.com","firstName":"","lastName":""},{"city":
"","country":"Afghanistan","emailAddress":"","firstName":"","lastName":""}]}}}
```

Example A–40 createUser Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

listApplications

Retrieve a list of all **application** objects.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/listApplications
```

Request Body

This operation does not use any request parameters.

Response Body

This operation returns these parameters:

- **listApplicationResponse** (**application** object list). A list of the requested **application** objects. Table A–9, " application Object Parameters" lists the **application** object parameters.

Table A–9 application Object Parameters

Parameter	Data Type	Description
appKey	String	(Optional) A unique ID used to authorize the application. Once set, this cannot be changed. This is set by a Partner unless the automatic approval setting is on, in which case it can be set by a Partner Manager.
applicationAPIs	application API object	The **applicationAPI** object that defines the API. Table A–10, " applicationAPI Object Parameters" lists the **applicationAPI** object parameters. This object can be null, which creates an application without any subscribed APIs.
applicationID	String	A unique ID used to identify the object.
applicationName	String	The application's name
applicationStatus	String	One of: ■ CREATE PENDING APPROVAL ■ DELETE PENDING APPROVAL ■ UPDATE PENDING APPROVAL ■ PASSWORD RESET ■ ACTIVE ■ DENY ■ UNKNOWN ■ SUSPENDED

Table A–9 (Cont.) application Object Parameters

Parameter	Data Type	Description
clientId	String	(Optional) A unique ID used to identify the client. This field can be changed. This is set by a Partner unless the automatic approval setting is on, in which case it can be set by a Partner Manager
description	String	An informal description of the object.
effectiveFrom	Date	The date the application starts being effective, in YYYY-MM-DD+HH:mm format.
effectiveTo	Date	That date that the application expires, in YYYY-MM-DD+HH:mm format.
lockStatus	String	One of: LOCKED, UNLOCKED.
partnerName	String	The name of the partner that the application belongs to.
quota	quota object	Table A–5, " quota Object Parameters" lists the quota parameters.
rate	rate object	Table A–6, " rate Object Parameters" lists the rate parameters
icon	String	The path to an icon to use. Used to specify custom graphics.
submitDate	Date	The date the application was submitted in YYYY-MM-DD+HH:mm format
trafficUser	String	The traffic user, used for authentication.
trafficPassword	String	A password to authenticate traffic. Minimum length is 8 characters. For the "updateApplication" operation, this password can be 32-byte encrypted with the AES key you set in “Encrypt Application Passwords.”

Table A–10 applicationAPI Object Parameters

Parameter	Data Type	Description
apiName	String	The API name.
accessURL	URL	The URl used to access the API.
apiVersion	String	The API version number
apiDescription	String	An informal description for the API
applicationMethodSLAs	application methodSLA object	The **applicationMethodSLA** object that defines rate and quota information for the API. Table A–11, " applicationMethodSLA Object Parameters" lists the **applicationMethodSLA** parameters.

Table A–11 applicationMethodSLA Object Parameters

Parameter	Data Type	Description
methodName	String	The name of the method given SLA restrictions.
inerfaceName	String	The name of the interface the method uses.
quota	quota object	Table A–5, " quota Object Parameters" lists the **quota** object parameters.
rate	rate object	Table A–6, " rate Object Parameters" lists the **rate** object parameters.

Examples

Example A–41 listApplications Request Example

```
GET https://10.182.98.78:9001/apps
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/listApplications
```

Example A–42 listApplications Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"listApplicationsResponse":{"return":[{"applicationID":
"07c02413-0eeb-4a73-8920-ba7b8cbb97de","applicationName":"app-04","partnerName":
"rotterzeng","partnerCompany":"oracle","description":"aaaaaaaaaaaaaaaaaaaaaaaaa",
"applicationAPIs":[{"apiName":"api-0918-1","accessURL":"https://localhost:8001/
daf/api-0918-1/1","apiVersion":"1","apiDescription":
"hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh","applicationMethodSLAs":[{"methodName":"
","interfaceName":"eda339e1-f8a5-4307-bf3e-654f2e0cd09c","quota":{"days":0,
"limitExceedOK":false,"qtaLimit":0},"rate":{"reqLimit":0,"timePeriod":0},
"methodGuarantee":{"reqLimitGuarantee":0,"timePeriodGuarantee":0}}],
"needReadContract":false}],"submitDate":"2014-09-18+08:00","effectiveFrom":
"2014-09-01+08:00","effectiveTo":"2014-09-30+08:00","status":"CREATE PENDING
 APPROVAL","lockStatus":"UNLOCKED","quota":{"days":1,"limitExceedOK":true,
"qtaLimit":1},"rate":{"reqLimit":1,"timePeriod":1}}]}}
```

getApplication

Retrieve **application** object parameters.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/getApplication/*applicationID*}

Request Body

This operation uses these request parameters:

- **applicationID** (String) Required. The identifier of the **application** object to return.

Response Body

This operation returns these parameters:

- **getApplicationResponse** (**application** object). Table A–9, " application Object Parameters" lists the **application** object parameters.

Examples

Example A–43 getApplication Request Example

```
GET https://10.182.98.78:9001/apps
prm_pm_rest/services/partner_manager/application/PartnerManagerApplication/
getApplication/07c02413-0eeb-4a73-8920-ba7b8cbb97de
```

Example A–44 getApplication Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getApplicationResponse":{"return":{"applicationID":"07c02413-0eeb-4a73-8920-
ba7b8cbb97de","applicationName":"app-04","partnerName":"guest1",
"partnerCompany":"oracle","description":"aaaaaaaaaaaaaaaaaaaaaaaaa",
"applicationAPIs":[{"apiName":"api-0918-1","accessURL":"https://localhost:8001/daf
/
api-0918-1/1","apiVersion":"1","apiDescription":
"hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh","applicationMethodSLAs":[{"methodName":"",
"interfaceName":"eda339e1-f8a5-4307-bf3e-654f2e0cd09c","quota":{"days":0,
"limitExceedOK":false,"qtaLimit":0},"rate":{"reqLimit":0,"timePeriod":0},
"methodGuarantee":{"reqLimitGuarantee":0,"timePeriodGuarantee":0}}],
"needReadContract":false}],"submitDate":"2014-09-18+08:00","effectiveFrom":
"2014-09-01+08:00","effectiveTo":"2014-09-30+08:00","status":"CREATE PENDING
 APPROVAL","lockStatus":"UNLOCKED","quota":{"days":1,"limitExceedOK":true,
"qtaLimit":1},"rate":{"reqLimit":1,"timePeriod":1}}}}
```

updateCurrentSlaForApprove

Approve an application registration request.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/updateCurrentSlaForApprove
```

Request Body

This operation uses these request parameters:

- **updateCurrentSlaForApprove** (**application** object) Mandatory. The application object of the application. Table A–9, " application Object Parameters" lists the **application** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–45 updateCurrentSLAForApprove Request Example

```
POST https://10.182.98.78:9001/apps
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/updateCurrentSlaForApprove
{"updateCurrentSlaForApprove":{"application":{"notificationId":
"18ddcd29-d883-4fab-a1f8-83f5322feb67","applicationID":
"5548f811-18a5-45d6-b5d9-6cc0b75d6bfa","applicationName":
"app_0925_1","partnerName":"rotterzeng","partnerCompany":"oracle",
"description":"app_0925_1app_0925_1app_0925_1","trafficUser":
"rotterzeng_app_0925_1","trafficPassword":"rotterzeng_app_0925_1",
"submitDate":"2014-09-28+08:00","effectiveFrom":"2014-09-01+08:00",
"effectiveTo":"2014-09-30+08:00","status":"CREATE PENDING APPROVAL",
"lockStatus":"UNLOCKED","quota":{"qtaLimit":1,"limitExceedOK":false,"days"
:1},"rate":{"reqLimit":1,"timePeriod":1}}}}
```

Example A–46 updateCurrentSLAForApprove Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

denyApplication

Deny an application registration request.

Authorization

Partner Manager Administrator

HTTP Method

POST

URI

```
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/denyApplication
```

Request Body

This operation uses these request parameters:

- **application** (**application** object) Required. The **application** object of the application request you are denying. Table A–9, " application Object Parameters" lists the **application** object parameters.

Response Body

This operation does not use any response these parameters.

Examples

Example A–47 denyApplication Request Example

```
POST https://10.182.98.78:9001/apps
/prm_pm_rest/services/partner_
manager/application/PartnerManagerApplication/denyApplication{"denyApplication":
{"application":{"notificationId":"de6a031c-30a2-48c5-9908-3adac59510f0",
"applicationID":"7781766f-38ec-4adc-a1e6-b7acd0c501bf","applicationName":
"app0925_2","partnerName":"rotterzeng","partnerCompany":
"oracle","description":"app0925_2app0925_2app0925_2app0925_2",
"trafficUser":"rotterzeng_app0925_2","trafficPassword":
"rotterzeng_app0925_2","submitDate":"2014-09-28+08:00","effectiveFrom":
"2014-09-01+08:00","effectiveTo":"2014-09-30+08:00","status":
"CREATE PENDING APPROVAL","lockStatus":"UNLOCKED","quota":
{"days":1,"limitExceedOK":true,"qtaLimit":1},"rate":{"reqLimit":1,
"timePeriod":1}}}}
```

Example A–48 denyApplication Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getAllSysConfig

Retrieve all system configuration settings as a list of key/value pairs.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/getAllSysConfig
```

Request Body

This operation use any request parameters.

Response Body

This operation returns these parameters:

- **getAllSysConfigResponse** (**sysConfig** object list) Required. Table A–12, " sysConfig Object Parameters" lists the **sysConfig** object parameters.

Table A–12 sysConfig Object Parameters

Parameter	Data Type	Description
key	String	The name of a configuration setting.
value	String	The value for a configuration setting.

Examples

Example A–49 getAllSysConfig Request Example

```
GET https://10.182.98.78:9001/config
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/getAllSysConfig{"GetAllSysConfig":{"return":
[{"key":"KEY_AUTO_APPROVE_FOR_REGISTER","value":"false"}]}}
```

Example A–50 getAllSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"GetAllSysConfigResponse":{"return":[{"key":"KEY_AUTO_APPROVE_FOR_
REGISTER","value":"false"}]}}
```

getSysConfig

Retrieve a single system configuration object.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/getSysConfig/key
```

Request Body

This operation uses these request parameters:

- **key** (String) Required. The name of the configuration setting to retrieve.

Response Body

This operation returns these parameters:

- **getSysConfigResponse** (**sysConfig** object). The **sysConfig** object containing the configuration setting. Table A–12, " sysConfig Object Parameters" lists the **sysConfig** object parameters.

Examples

Example A–51 getSysConfig Request Example

```
GET https://10.182.98.78:9001/config
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/getSysConfig/"key":
"KEY_AUTO_APPROVE_FOR_REGISTER"
```

Example A–52 getSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"GetSysConfigResponse": {"return": {"key":
"KEY_AUTO_APPROVE_FOR_REGISTER","value": "false"}}}
```

updateAllSysConfig

Update all system configuration settings, and optionally configure a list of IP addresses allowed to communicate with Serivices Gatekeeper.

Authorization

Partner Manager Administrator

HTTP Method

PUT

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/updateAllSysConfig
```

Request Body

This operation uses these request parameters:

- **updateAllSysConfig** (**sysConfig** object list) Required. Table A–12, " sysConfig Object Parameters" lists the **sysConfig** object parameters.

  This operation also takes a special key/value pair (**prmIPAddressWhitelist**/*list of IP addresses*) to use as a "white list" of IP addresses allowed to communicate with Services Gatekeeper. See "Protecting REST APIs with a White List of IP Addresses" for details.

Response Body

This operation does not use response parameters.

Examples

Example A–53 updateAllSysConfig Request Example

```
PUT https://10.182.98.78:9001/config
/prm_pm_rest/services/prm_pm/services/partner_manager/sysconfig/updateAllSysConfig
{"updateAllSysConfig":{"sysConfig":[{"key":"APP","value":"true"},
{"key":"APP1","value":"false"}]}
```

Example A–54 updateAllSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

updateSysConfig

Retrieve a single configuration setting key/value pair.

Authorization

Partner Manager Administrator

HTTP Method

PUT

URI

`/prm_pm_rest/services/prm_pm/services/partner_manager/sysconfig/updateSysConfig/`*key*

Request Body

This operation uses these request parameters:

- **key** (String) Required. The name of the configuration setting to retrieve.

Response Body

This operation does not use response parameters.

Examples

Example A–55 updateSysConfig Request Example

```
PUT https://10.182.98.78:9001/config
/prm_pm_rest/services/prm_pm/services/partner_manager/sysconfig/updateSysConfig/
{"updateSysConfig":{"key":"APP","value":"true"}}
```

Example A–56 updateSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getBlobSysConfig

Retrieve complex configuration information, such as long text descriptions, or graphics.

Authorization

Partner Manager Administrator

HTTP Method

GET

URI

`/prm_pm_rest/services/prm_pm/services/partner_manager/sysconfig/getBlobSysConfig/`*key*

Request Body

This operation uses these request parameters:

- **key** (String) Required. The name of the configuration setting to retrieve.

Response Body

This operation returns these parameters:

- **return** (Blob list). The configuration information.

Examples

Example A–57 getBlobSysConfig Request Example

```
GET https://10.182.98.78:9001/config
/prm_pm_rest/services/prm_pm/services/partner_manager/sysconfig/getBlobSysConfig/
{"getBlobSysConfig":{"key":"TITLE","value":"icon1.jpeg"}}
```

Example A–58 getBlobSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

updateBlobSysConfig

Update a complex configuration setting.

Authorization

Partner Manager Administrator

HTTP Method

PUT

URI

```
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/updateBlobSysConfig/key
```

Request Body

This operation uses these request parameters:

- **key** (String) Required. The name of the configuration setting to retrieve.

Response Body

This operation does not use response parameters.

Examples

Example A–59 updateBlobSysConfig Request Example

```
GET https://10.182.98.78:9001/subscribers
/prm_pm_rest/services/prm_pm/services/partner_
manager/sysconfig/updateBlobSysConfig
{"updateBlobSysConfig":{"key":"TITLE","value":"icon1.jpeg"}}
```

Example A–60 updateBlobSysConfig Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

registerSP

Register an account (subscriber profile).

Authorization

Partners are authorized to use this operation.

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_pr/services/register/Register/registerSP
```

Request Body

This operation uses these request parameters:

- **registerSP** (**userInfo** object) Required. The **userInfo** object contains details for the subscriber to register. Table A–7, " userInfo Object Parameters" lists the **userInfo** object parameters. The password parameter supports 24 and 32-byte encryption.

Response Body

This operation does not use response parameters.

Examples

Example A–61 registerSP Request Example

```
GET https://10.182.98.78:9001/subscribers
{"registerSP":{"spInfo":{"userName":"testuser1","userType":
"PRM_SP","emailAddr":"testuser1@testcompany.com","password":"weblogic123","phone":
"1-1-12345678","secureityAnswerChoice":0,"secureityAnswer":"testuser","firstName":
"test","lastName":"test","company":"oracle","companyURL":"https://www.oracle.com",
"
city":"bj","country":"China","contacts":[]}}}
```

Example A–62 registerSP Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

editUser

This operation changes **userInfo** object subscriber profile parameters for one subscriber.

Authorization

Partners are authorized to use this operation.

HTTP Method

PUT

URI

`/prm_pm_rest/services/prm_pr/services/account/PortalAccount/editUser`

Request Body

This operation uses these request parameters:

- **editUser** (**userInfo** object) Required. The **userInfo** object contains changes to the user's subscriber profile. Table A–7, " userInfo Object Parameters" lists the **userInfo** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–63 editUser Request Example

```
GET https://10.182.98.78:9001/subscribers
/prm_pm_rest/services/prm_pr/services/account/PortalAccount/editUser
{"editUser":{"userInfo":{"userName":"rotterzeng","userType":
"PRM_SP","password":"{AES}GrVKe6y4ccO3wKB5DuJ3ng==","emailAddr":
"testuser@testcompany.com","phone":
"1234567890","secureityAnswerChoice":0,"secureityAnswer":"2","firstName":"rotter",
"lastName":"zeng","company":"oracle","companyURL":"https://123.com",
"stateOrProvince":"bj","zipOrPostalCode":"","streetAddress":"a","city":"a",
"country":"Afghanistan","contacts":[{"city":"","country":"Afghanistan",
"emailAddress":"testuser@testcompany.com","firstName":"","lastName":""},
{"city":"","country":"Afghanistan","emailAddress":"","firstName":"",
"lastName":""}]}}}
```

Example A–64 editUser Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getUserByName

Retrieves subscriber profile information for a subscriber based on their user name.

Authorization

Partners are authorized to use this operation

HTTP Method

GET

URI

`/prm_pm_rest/services/prm_pr/services/account/PortalAccount/getUserByName/`*userName*

Request Body

This operation uses these request parameters:

- **userName** (String) Required. Identifies the **userInfo** object that you are changing. Table A–7, " userInfo Object Parameters" lists the **userInfo** object parameters.

Response Body

This operation returns these parameters:

- **getUserByNameResponse** (**userInfo** object). The **userInfo** object of containing the user's subscriber profile information. Table A–7, " userInfo Object Parameters" lists the **userInfo** parameters.

Examples

Example A–65 getUserByName Request Example

```
GET https://10.182.98.78:9001/subscribers
/prm_pm_rest/services/prm_
pr/services/account/PortalAccount/getUserByName/{testuser2}
```

Example A–66 getUserByName Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getUserByNameResponse":{"return":{"city":"a","company":"oracle","companyURL":
"https://123.com","contacts":[{"city":"","country":"Afghanistan","emailAddress":"
","firstName":"","lastName":""},{"city":"","country":"Afghanistan","emailAddress":
"testuser@testcompany.com","firstName":"","lastName":""}],"country":"Afghanistan",
"emailAddr":"testuser@testcompany.com","financial":{"bankAccountNumber":"",
"bankAddress":"","bankName":"","bankRoutingNumber":"","city":"","country":"",
"invoiceTo":"","referenceAccount":"","stateOrProvince":"","taxID":"",
"zipOrPostalCode":""},"firstName":"testuser2","lastName":"zeng","password":
"{AES}GrVKe6y4ccO3wKB5DuJ3ng==","phone":"1234567890","secureityAnswer"
:"2","secureityAnswerChoice":
"0","stateOrProvince":"bj","status":0,"streetAddress":"a","userName":"rotterzeng",
"zipOrPostalCode":"","slaGroup":"VIP","userType":"PRM_SP"}}}
```

listApplications

This operation lists operations that belong to a partner.

Authorization

Partners are authorized to use this operation.

HTTP Method

GET

URI

```
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/listApplications
```

Request Body

This operation does not use any request parameters.

Response Body

This operation returns these parameters:

- **listApplicationResponse** (**application** object). The **application** object lists the application parameters. Table A–9, " application Object Parameters" lists the **application** object parameters.

Examples

Example A–67 listApplications Request Example

```
GET https://10.182.98.78:9001/apps
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/listApplications
```

Example A–68 listApplications Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"listApplicationsResponse":{"return":[{"applicationID":"07c02413-
0eeb-4a73-8920-b
a7b8cbb97de","applicationName":"app-04","partnerName":"rotterzeng","partnerCompany
":"oracle","description":"aaaaaaaaaaaaaaaaaaaaaaaaaa","applicationAPIs":[{"apiName
":"api-0918-1","accessURL":"https://localhost:8001/daf/api-0918-1/1",
"apiVersion":"1","apiDescription":"hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh",
"applicationMethodSLAs":[
{"methodName":"","interfaceName":"eda339e1-f8a5-4307-bf3e-654f2e0cd09c","quota":{"
days":0,"limitExceedOK":false,"qtaLimit":0},"rate":{"reqLimit":0,"timePeriod":0},"
methodGuarantee":{"reqLimitGuarantee":0,"timePeriodGuarantee":0}}],"needReadContra
ct":false}],"submitDate":"2014-09-18+08:00","effectiveFrom":"2014-09-01+08:00","ef
fectiveTo":"2014-09-30+08:00","status":"ACTIVE","lockStatus":"UNLOCKED","quota":{"
days":1,"limitExceedOK":true,"qtaLimit":1},"rate":{"reqLimit":1,"timePeriod":1}}}]
}}
```

createApplication

This operation creates an **application** object.

Authorization

Partners are authorized to use this operation.

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/createApplication
```

Request Body

This operation uses these request parameters:

- **createApplication** (**application** object) Required. The **application** object contains the application parameters. Table A–9, " application Object Parameters" lists the application parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–69 createApplication Request Example

```
POST https://10.182.98.78:9001/apps
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/createApplication
{"createApplication":{"application":{"applicationName":"app-0929","description":
"app-0929app-0929app-0929app-0929","effectiveFrom":"2014-09-01","effectiveTo":
"2014-09-30","partnerName":"rotterzeng","quota":{"days":"1","limitExceedOK":true,"
qtaLimit":"1"},"rate":{"reqLimit":"1","timePeriod":"1"},"applicationAPIs":
[{"apiName":"8cdc8cdc-61e0-4ec2-9fc1-c9e71c1821e8"}]}}}
```

Example A–70 createApplication Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

updateApplication

This operation changes one or more parameters of an application.

Authorization

Partners are authorized to use this operation.

HTTP Method

PUT

URI

```
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/updateApplication
```

Request Body

This operation uses these request parameters:

- **updateApplication** (**application** object) Required. The **application** object contains the new **application** object parameters. Table A–9, " application Object Parameters" lists the application parameters. The **trafficPassword** parameter supports 24 and 32-byte encryption.

Response Body

This operation does not use any response parameters.

Examples

Example A–71 updateApplication Request Example

```
PUT https://10.182.98.78:9001/apps
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/updateApplication
{"updateApplication":{"application":{"applicationID":"53f91602-01af-4483-a7ee-
3fc4af6b1280","applicationName":"app-0929","description":"app-0929app-0929app-
0929app-0929","effectiveFrom":"2014-09-01+08:00","effectiveTo":"2014-09-30+08:00",
"trafficUser":"rotterzeng_
app-0929","partnerName":"rotterzeng","quota":{"days":"1","qtaLimit":"1"},"rate":
{"reqLimit":"100","timePeriod":"1"},"applicationAPIs":[{"apiName":"8cdc8cdc-61e0-4
ec2-9fc1-c9e71c1821e8"}]}}}
```

Example A–72 updateApplication Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

removeApplication

This operation deletes an **application** object.

Authorization

Partners are authorized to use this operation.

HTTP Method

DELETE

URI

`/prm_pm_rest/services/prm_pm/services/partner/application/PartnerApplication/removeApplication/`*`applicationID`*

Request Body

This operation uses these request parameters:

- **applicationID** (String) Required. The **applicationID** of the **application** object to remove. Table A–9, " application Object Parameters" lists the **application** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–73 removeApplication Request Example

```
DELTET https://10.182.98.78:9001/apps
/prm_pm_rest/services/prm_pm/services/partner/application/
PartnerApplication/removeApplication/{applicationID}
{"applicationID":"53f91602-01af-4483-a7ee3fc4af6b1280"}
```

Example A–74 removeApplication Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

removePendingApp

This operation removes an application in a PENDING state. Used when a partner wants to cancel an pending operation before the partner administrator has acted on it.

Authorization

Partners are authorized to use this operation.

HTTP Method

DELETE

URI

`/prm_pm_rest/services/prm_pm/services/partner/application/PartnerApplication/removePendingApp/`*applicationID*

Request Body

This operation uses these request parameters:

- **applicationID** (String) Required. The **applicationID** value that identifies the **application** object to remove. Table A–9, " application Object Parameters" lists the **application** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–75 removePendingApp Request Example

```
DELETE https://10.182.98.78:9001/apps
/prm_pm_rest/services/prm_
pm/services/partner/application/PartnerApplication/removePendingApp/
{"applicationID":"53f91602-01af-4483-a7ee3fc4af6b1280"}
```

Example A–76 removePendingApp Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

getInterfaceList

Lists all interfaces that belong to a network service supplier.

Authorization

Network service suppliers and partner managers are authorized to execute this operation.

HTTP Method

GET

URI

`/prm_pm_rest/services/prm_pr/services/svrinterface/Interface/getInterfaceList/`*supplierName*

Request Body

This operation uses these request parameters:

- **supplierName** (String) Required. The name of the supplier to return **svrinterfaceInfo** objects for.

Response Body

This operation returns these parameters:

- **getInterfaceListResponse** (**svrinterfaceInfo** object list). This operation returns a list of **svrinfterfaceInfo** objects, one for each interface that the network service supplier owns.Table A–13, " svrinterfaceListResponse Object Parameters" lists the **svrinterfaceListResponse** parameters.

Table A–13 svrinterfaceListResponse Object Parameters

Parameter	Data Type	Description
id	String	The interface identifier.
name	String	The interface name
version	String	The version number of the interface
status	String	One of: UNKNOWN, CREATED, APPROVED, ACTIVE, DEPRECATED, REMOVED, DENY, SUSPENDED.
supplierName	String	The name of the network service supplier that created the interface.
accessURL	String	The interface's access URL.
wsdlurl	String	The URL of the WSDL/WADL file that the interface uses.
validtime	Date	The date and time the interface was validated.
deprecatetime	Date	The date and time the interface was deprecated.
suspendtime	Date	The date and time the interface was suspended.
docURL	String	The URL for documentation for the interface.

Table A–13 (Cont.) svrinterfaceListResponse Object Parameters

Parameter	Data Type	Description
securityType	String	One of: NONE, TEXT, or OAUTH.
authToken	String	The authorization token. Can be one of: ■ For security type NONE, no value ■ For security type TEXT, uses the *userName*:*password* format ■ For security type OAUTH, uses the access token.
icon	String	The location of the directory of the icon to use
description	String	An informal description of the interface.
throughput	Interger	The maximum throughput allowed.

Examples

Example A–77 getInterfaceList Request Example

```
GET https://10.182.98.78:9001/interfaces
/prm_pm_rest/services/prm_
pr/services/svrinterface/Interface/getInterfaceList/{supplierName}
{"supplierName":"sxh"}
```

Example A–78 getInterfaceList Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
{"getInterfaceListResponse":{"return":[{"id":"b454f603-925b-4b04-9f0e-d207f4db
80a0","name":"first
interface","version":"1.0","status":"ACTIVE","supplierName":"sxh","accessURL":
"https://access.url/v1","WSDLURL":"https://access.url/v1?WADL","validtime":
1399618800,"deprecatetime":1420041599,"suspendtime":32503651199,"docURL":"https:
//doc.access.url/v1","securityType":"NONE","icon":"0","description":"desc",
"throughput":100}]}}
```

getInterfaceInfoByName

Retrieve a **svrinterface** object parameters that match the interface name and a version number that you request.

Authorization

Network service suppliers and partner managers are authorized to execute this operation.

HTTP Method

GET

URI

`/prm_pm_rest/services/prm_pr/services/svrinterface/Interface/getInterfaceInfoByName/`*svrinterfaceName*`/`*svrinterfaceVersion*

Request Body

This operation uses these request parameters:

- **svrinterfaceName** (String) Required. The name of the **svrinterface** object to return.
- **svrinterfaceVersion** (String) Required. The version number of the interface object to return.

Response Body

This operation does not use any response parameters.

Examples

Example A–79 getInterfaceInfoByName Request Example

```
GET https://10.182.98.78:9001/interfaces
/prm_pm_rest/services/prm_
pr/services/svrinterface/Interface/getInterfaceInfoByName
{"svrinterfaceName":"firstinterface","svrinterfaceVersion":"v1"}
```

Example A–80 getInterfaceInfoByName Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

updateInterface

This operation makes changes to the object containing an interface.

Authorization

Network service suppliers

HTTP Method

POST

URI

```
/prm_pm_rest/services/prm_
pr/services/svrinterface/Interface/updateInterface
```

Request Body

This operation uses these request parameters:

- **updateInterface** (**svrinterface** object) Required. The **svrinterface** object containing the change to make. Table A–13, " svrinterfaceListResponse Object Parameters" lists the **svrinterface** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–81 updateInterface Request Example

```
POST https://10.182.98.78:9001/interfaces
/prm_pm_rest/services/prm_pr/services/svrinterface/Interface/updateInterface
{"updateInterface":{"svrinterfaceInfo":{"id":"b454f603-925b-4b04-9f0e-
d207f4db80a0","name":"first
interface","version":"2.0","status":"ACTIVE","supplierName":"sxh","accessURL":
"https://access.url/v1","WSDLURL":"https://access.url/v1?WADL","validtime":1411979
30
2,"deprecatetime":1414571290,"suspendtime":1411979302,"docURL":"https://doc.access
.
url/v1","securityType":"NONE","icon":"0","description":"desc","throughput":100,
"activeDate":"2014-05-09","activeTime":"15:00:00","deprecateTime":"16:28:10",
"deprecateDate":"2014-10-29"}}}
```

Example A–82 updateInterface Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

removeInterfaceByID

This operation deletes a network service supplier interface based on the interface ID.

Authorization

Network service suppliers

HTTP Method

DELETE

URI

`/prm_pm_rest/services/prm_pr/services/svrinterface/Interface/removeInterfaceByID/`*svrinterfaceID*

Request Body

This operation uses these request parameters:

- **svrinterfaceID** (String) Required. The **svrinterface** object identifier of the interface to delete. Table A–13, " svrinterfaceListResponse Object Parameters" lists the **svrinterface** object parameters.

Response Body

This operation does not use any response parameters.

Examples

Example A–83 removeInterfaceByID Request Example

```
GET https://10.182.98.78:9001/interfaces
/prm_pm_rest/services/prm_
pr/services/svrinterface/Interface/removeInterfaceByID/{svrinterfaceID}
{"svrinterface":{"id":"b454f603-925b-4b04-9f0e-d207f4db80a0"}}
```

Example A–84 removeInterfaceByID Response Example

```
HTTP/1.1 200 OK
Content-Length: 0
Server: Jetty(8.0.1.0)
```

B

Actions Management REST-Based API

This appendix describes the RESTful API interface that Oracle Communications Services Gatekeeper uses to manage the actions that operate on traffic for an API.

Understanding the Actions Management API

You use this API to replicate the functionality found in the **Actions** tab for individual APIs in the Partner and API Management Portal.

Like any RESTful API, the Actions Management API makes its services available to client applications through simple HTTP requests. However, HTTP secure (https) is required for all operations in this API.

Several types of clients connect to Services Gatekeeper, including the Administration Console client process and the portal GUIs. Services Gatekeeper listens for Actions Management API requests at the port configured for client HTTP access. Before using this API, ensure that the Managed Server is configured to accept web requests.

This API is defined in the **actionChains.wadl** file.

Understanding Actions Management API Error Handling.

This API shares the same error handling characteristics as the API Management API. See "Understanding API Management API Error Handling" for details.

Actions Management Operations

The following sections list the RESTful operations of the Actions Management API.

submitActionChain

You use this method to submit a chain of actions for an API.

Authorization

Basic

HTTP Method

POST

URI

/submitActionChain

Request Body

This operation accepts these request body parameters:

- The **serviceURI** that identifies the API.
- A list of **requestActions** values to perform, each includes the action name and a string defining the action. See "Sample requestAction Schemas" for examples.
- The version of the API to run the actions against.

Sample requestAction Schemas

This section lists example action schemas. You use these actions in the **requestAction** field of the **submitActionChain** method.

This example schema is for a callout action:

Name: **Callout**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="calloutActionConfig" type="calloutActionConfig"/>
    <xs:complexType final="extension restriction" name="calloutActionConfig">
        <xs:sequence>
            <xs:element minOccurs="0" name="requestUrl" type="xs:string"/>
            <xs:element minOccurs="0" name="storeResponse" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This example schema is for a Groovy action:

Name: **Groovy**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="groovyActionConfig" type="groovyActionConfig"/>
    <xs:complexType final="extension restriction" name="groovyActionConfig">
        <xs:sequence>
            <xs:element minOccurs="0" name="groovyScript" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This example schema is for an XLST action:

Name: **XLST**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="xsltActionConfig" type="xsltActionConfig"/>
    <xs:complexType final="extension restriction" name="xsltActionConfig">
        <xs:sequence>
            <xs:element minOccurs="0" name="xslt" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This example schema is for a schema validation action:

Name: **SchemaValidation**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="schemaValidateActionConfig"
type="schemaValidateActionConfig"/>
    <xs:complexType name="schemaValidateActionConfig">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="apiSchemas" nillable="true" type="apiSchema"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType final="extension restriction" name="apiSchema">
        <xs:sequence>
            <xs:element minOccurs="0" name="content" type="xs:string"/>
            <xs:element minOccurs="0" name="name" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This example schema is for a Jason2Xml action:

Name: **Json2Xml**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="json2XmlAction">
        <xs:sequence/>
    </xs:complexType>
</xs:schema>
```

This example schema is for an Xml2Jason action (empty):

Name: **Xml2Json**

Action:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="xml2JsonAction">
        <xs:sequence/>
    </xs:complexType>
</xs:schema>
```

Response Body

This operation returns an HTTP 200 status code on success, and an HTTP error status code on failure.

Example Request

```
http://op:192.168.10.1@localhost:8001/prm_pm_rest/services/prm_pm/
services/partner_manager/actionchain/submitActionChain
'{"createAPI":{"apiObject":{"apiName":"weather","apiVersion":"1","accessUrl":
"http://localhost:8001/daf/weather/1","apiInterfaces":[{"name":"weather-v1",
"apiMethods":[{"name":"main","displayName":"main","path":"/","httpVerb":"GET",
"servicePath":"/","
serviceHttpVerb":"GET","expose":true}],"displayName":"weather-v1","fileLocation":"
```

```
http://www.weather-documentation.com/"}],"wadlFiles":[],"northBoundWadlFiles":[],"
description":"An API Describing the weather in a specific
zipcode","facade":"REST","direction":"AOMT","seviceType":"by-url","protocol":
"http://www.weather.com","privilege":0,"link":
"http://www.weather-documentation.com/","accessType":"HTTP","authType":"NONE",
"authToken":"","networkProxy":"","networkAuthorizationURI":"","networkTokenURI":"
","networkClientRedirectURI":"","groups":null,"icon":"expressive/weather.png"}}}'
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

retrieveActionChain

You use this method to retrieve the actions for an API, by sending in the **serviceUrl** that identifies the API.

Authorization

Basic

HTTP Method

GET

URI

/retrieveActionChain/*serviceURI*

Where *serviceURI* identifies the API to retrieve an action chain for.

Request Body

This operation does not send any request body parameters:

Response Body

A successful response includes:

- An HTTP 200 OK status message.
- The name of the API.
- A list actions to perform, each includes the action name and a string defining the action.
- The current API version.

An unsuccessful response returns an HTTP 400 status message.

Example Request

```
http://op:192.168.10.1@localhost:8001/prm_pm_rest/services/prm_pm/
services/partner_manager/actionchain/retrieveActionChain/testWeatherApp
```

Example Response

Some of the actions in this example response are custom actions:

```
 {"retrieveActionChainResponse":{"requestActions":[{"name":"HeaderValidation","
content":"<?xml version=\"1.0\" encoding=\"UTF-8\"
```

```
 standalone=\"yes\"?>\n<headerValidationActionConfig>\n
 <headerKey>bob</headerKey>\n
 <headerValue>123</headerValue>\n</headerValidationActionConfig>\n"},
{"name":"BlackList","content":"<?xml version=\"1.0\"
 encoding=\"UTF-8\" standalone=\"yes\"?>\n<blackListActionConfig>\n
 <address>localhost</address>\n</blackListActionConfig>\n"}],"configVersion":9}}
```

loadActionSchemas

You use this method to retrieve the action schemas.

Authorization

Basic

HTTP Method

GET

URI

`/loadActionSchemas`

Request Body

This operation does not send any request body parameters. It returns actions for all schemas.

Response Body

A successful operation returns these parameters:

- The name of the action.
- The schema itself.
- Any flow restriction (including the flow restriction string and request/response.
- Any PRE or POST-processing restriction, including the restriction itself, and a POST or PRE value.
- A description of the action schema.

An unsuccessful response returns an HTTP 400 status message.

Example Request

```
http://op:192.168.10.1@localhost:8001/prm_pm_rest/services/prm_pm/
services/partner_manager/actionchain/loadActionSchemas
```

Example Response

The response will look like the schemas in "Sample requestAction Schemas".

verifyAction

You use this method to verify the validity of an action. This method is used by the **Validate** button on the **Actions** sub-tab of the API and Partner Management Portal for individual APIs. A successful response returns an HTTP 200 status message. An unsuccessful response returns an HTTP failure status message, and whatever details are available.

Authorization

Basic

HTTP Method

POST

URI

```
/verifyAction
```

Request Body

This operation accepts these request body parameters:

- An array of **name** parameters that identify actions.
- An array of **content** parameters includes the XML data that corresponds to the actions configuration.

Example Request

```
http://op:192.168.10.1@localhost:8001/prm_pm_rest/services/prm_pm/
services/partner_manager/actionchain/verifyAction
 '{"verifyAction":{"name":"Groovy","actionConfig":"<?xml version=\"1.0\"
 encoding=\"UTF-8\" standalone=\"yes\"?><groovyActionConfig><groovyScript>&amp;
^&amp;$%^&amp;$^%&amp;$%^&amp;</groovyScript></groovyActionConfig>"}}'
```

Example Response

This example shows a successful validation response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

This example shows a failed Groovy action response:

```
HTTP/1.1 500 Internal Server Error
Date: Wed, 23 Sep 2015 18:00:23 GMT
Content-Length: 251
Content-Type: text/plain

RESTful facade receives internal exception,refer the server log for detail:
class oracle.ocsg.portal.ws.partner_manager.actionchain.ActionChainException
Detail Message: 1 problem found in configuration.
[1]: unexpected token: & @ line 14, column 1.
```

C

Partner Relationship Management SOAP-based Web Services Interface

This interface is deprecated and will be removed in a future release. Oracle recommends that you use the REST-based interfaces instead.

This appendix describes the SOAP-based Web Service interface that Oracle Communications Services Gatekeeper supports. You use this interface to extend and customize the GUIs supplied with pre-6.0 Services Gatekeeper releases.

About the Interface Functionality

The functionality in the Web services interface is organized in the following way:

- Interface: SpLogin
- Interface: OpLogin
- Interface: SpService
- Interface: OpService
- Interface: SpCdrUtil
- Interface: SpStatisticsUtil
- Interface: OpCdrUtil
- Interface: OpStatisticsUtil
- Interface: OpAlarmUtil

Base Service Error Messages

Individual interfaces have their own error messages. These error messages are common across all interfaces:

- "The appAccount in registered PENDING state"
- "The spAccount in registered PENDING state"
- "The AppInstGroup isn't in update pending status"
- "The AppAccount not in update pending status"
- "access failed"
- "The AppInstanceGrp not exist"
- "The AppInstanceGrp not belong to the AppAccount"
- "Application not existed"

- "the Application doesn't belong to the SpAccount"
- "Application in register pending status"
- "The AppAccount not exist"
- "The AppAccount doesn't belong to the Sp"
- "Sp Account in register pending state"
- "Sp Account in register pending state"

Interface: SpLogin

Services Gatekeeper supports the following functionality in the Service Provider Login Web Service interface:

- Login/logout capabilities for backward compatibility only
- A way for prospective service providers to request a Service Provider Account

The endpoint for this interface is:

`http://host:port/prm_sp/services/SpLogin`

where the value of *host* and *port* depend on the Services Gatekeeper deployment.

registerSpAccountReq

The **registerSpAccountReq** method is used to request a new Service Provider Account. When this request has been approved by the operator, the Service Provider can log in to the Partner Relationship Management module. For more information on the approval by the operator, see "registerSpAccountRes".

Method Signature

```
registerSpAccountReq(spAccountId, spAccount, password)
```

Input Parameters

Table C–1 describes the input parameters:

Table C–1 registerSpAccountReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The desired ID the Service Provider Account.
spAccount	tns1:SpAccount	Data structure with details on the Service Provider account. See "SpAccount".
password	xsd:string	The password associated with the Service Provider login account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

Complex Data Types

The Service Provider Login Web Service interface uses the following complex data types.

SpAccount

Table C–2 describes of a Service Provider Account, including contact details.

Table C–2 Contents of an SpAccount

Element name	Data Type	Description
Name	xsd:string	Name of the Service Provider.
Address	xsd:string	Address of the Service Provider.
EMailAddress	xsd:string	E-mail address of the Service Provider.
ContactPerson	xsd:string	Contact person at the Service provider.
PhoneNumber	xsd:string	Phone number to the Service Provider.
Properties	mpl:ArrayOf_tns1_Property	Customer relationship management (CRM)/Partner relationship management (PRM) application-defined name value pairs. See "Property".

Property

This is an array of name-value pairs. This data type is used in several other data types specific for this interface. The properties are accessible from the Service Provider interface and the Operator interface, so they can be used for communicating information between the Service Provider and the Operator.

Table C–3 describes the elements of a property.

Table C–3 Property Elements

Element name	Data Type	Description
Name	xsd:string	Name of the property, with the value defined in Value. Unique with the array.
Value	xsd:string	The data associated with Name.

Interface: OpLogin

The Operator Login Web Service in Services Gatekeeper's Partner Relationship Management module is provided for backward compatibility only. The current implement of PRM is sessionless.

The endpoint for this interface is:

`http://<host>:<port>/prm_op/services/OpLogin`

Where the value of *host* and *port* depend on the Services Gatekeeper deployment.

Interface: SpService

The Service Provider Service Web Service provides the Service Provider with operations for handling Service Provider Accounts, Application Accounts, and Applications Instances in Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_sp/services/SpService`

where the value of *host* and *port* depend on the Services Gatekeeper deployment.

SpService Error Messages

This interface can generate these error messages:

- "changeSpAccountPassword failed"
- "setAppInstGroupPassword failed"
- "getAppAccountSla failed"
- "getSpAccountSla failed"
- "getAppAccountSlaByType failed"
- "getSpAccountSlaByType failed"
- "getAppInstGroupState failed"
- "getAppInstGroup failed"
- "listAppInstGroups failed"
- "deactivateAppInstGroup failed"
- "activateAppinstGroup failed"
- "deactivateAppAccount failed"
- "activateAppAccount failed"
- "getAppAccountState failed"
- "getAppAccount failed"
- "listAppAccounts failed"
- "Only ACTIVE status can be updated"
- "Only DEACTIVE status can be deleted!"
- "the AppAcount existed already!"

deleteSpAccountReq

The **deleteSpAccountReq** method makes a request to delete the Service Provider Account. The request must be approved before the Service Provider Account is deleted. This is done by the operator, using "deleteSpAccountRes".

Method Signature

```
deleteSpAccountReq()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Service Provider Account's state cannot be deleted. For a Service Provider Account to be deleted, it must be in the INACTIVE state.

deactivateSpAccount

Deactivates the Service Provider Account, which changes the state of the account to INACTIVE.

Method Signature

```
deactivateSpAccount()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Service Provider Account is not in an appropriate state to allow the account to be deactivated.

activateSpAccount

Activates the Service Provider Account. which changes the state of the account to ACTIVE.

Method Signature

```
activateSpAccount()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Service Provider Account is not in an appropriate state to allow the account to be activated.

getSpAccount

Retrieves details about the Service Provider Account. The details include contact details and customer relationship management (CRM)/partner relationship management (PRM) application-defined properties in the form of name-value pairs.

Method Signature

```
getSpAccount()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

A data structure with details on the Service Provider account. See "SpAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getSpAccountState

Retrieves the state of the Service Provider Account. This operation takes no input parameters.

Method Signature

```
getSpAccountState()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

Information of the state of the Service Provider Account. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

registerAppAccountReq

Requests registration of an Application Account for the Service Provider Account. This request must be approved by the Operator: see "registerAppAccountRes".

Method Signature

```
deleteSpAccountReq(appAcountId, appAccount)
```

Input Parameters

Table C–4 describes the input parameters.

Table C–4 registerAppAccountReq Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appAccount	tns1:AppAccount	A data structure with details about the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "AppAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application account is not in the correct state to be registered.

deleteAppAccountReq

Requests deletion of an Application Account associated with the Service Provider Account. The request must be approved by the Operator before the Application Account is deleted using "deleteAppAccountRes". The Application Account must be in state INACTIVE in order for this call to be accepted.

Method Signature

`deleteAppAccountReq(`*`appAcountId`*`)`

Where *appAcountId* is the ID of the application account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application account is not in the right state to be deleted. The application account must be in the INACTIVE state before it can be deleted.

updateSpAccountReq

Requests an update of the Service Provider Account with new data.

Method Signature

```
updateSpAccountReq(spAccountId, spAccount)
```

Input Parameters

Table C–5 describes the input parameters.

Table C–5 updateSpAccountReq Input Parameters

Parameter Name	Type	Description
spAccountID	xsd:string	The ID of the service provider account.
spAccount	tns1:AppAccount	The data structure with details on the Service Provider Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "SpAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Service Provider Account is in an incorrect state and cannot be updated.

updateAppAccountReq

Requests an update to an Application Account associated with the Service Provider with new data.

Method Signature

`updateAppAccount(appAcountId, appAccount)`

Input Parameters

Table C–6 describes the input parameters.

Table C–6 updateAppAccount Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appAccount	tns1:AppAccount	A data structure with details about the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "AppAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application account is in an invalid state and cannot be updated.

updateAppInstGroupReq

Requests an update on an Application Instance. The request must be approved by the Operator. See "updateAppInstGroupRes".

Method Signature

```
updateAppInstGroupReq(appAcountId, appInstGroupId, appInstGroup)
```

Input Parameters

Table C–7 describes the input parameters.

Table C–7 updateAppInstGroupReq Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.
appInstGroup	tns1:AppInstGroup	A data-structure with the Application Instance CRM/PRM application-defined properties in the form of name-value pairs. See "AppInstGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application instance is in an invalid state and an update request cannot be made.

deleteAppInstGroupReq

Requests a deletion of an Application Instance. The request must be approved by the Operator. This is done using "deleteAppInstGroupRes".

Method Signature

`deleteAppInstGroupReq(`*`appAcountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–8 describes the input parameters.

Table C–8 deleteAppInstGroupReq Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application instance is in an invalid state and the deletion request cannot be processed.

listAppAccounts

Lists all Application Account IDs for the Service Provider. The result is filtered based on the state of the Application Account.

Method Signature

```
listAppAccounts(state)
```

Where *state* indicates the states on which to filter the result. See "State".

Return Parameters

Returns a list of application account IDs in a string array.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppAccount

Retrieves details about a specific Application Account. The operation returns a data structure with details on the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs.

Method Signature

`getAppAccount(`*`appAccountId`*`)`

Where *appAccountId* is the ID of the application account.

Return Parameters

Returns a data structure with details about the Application Account. See "AppAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppAccountState

Retrieves the state of a specific Application Account.

Method Signature

`getAppAccountState(`*`appAccountId`*`)`

Where *appAccountId* is the ID of the application account.

Return Parameters

Returns information on the state of the Application Account. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

activateAppAccount

Activate an Application Account, which changes the state of the account to ACTIVE. The current state of the account must be INACTIVE.

Method Signature

`activateAppAccount(`*`appAcountId`*`)`

Where *appAcountId* is the ID of the application account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application account is in an invalid state and cannot be activated.

deactivateAppAccount

Deactivates an Application Account, which changes the state of the account to INACTIVE. The current state of the account must be ACTIVE.

Method Signature

`deactivateAppAccount(`*`appAcountId`*`)`

Where *appAcountId* is the ID of the application account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application account is in an invalid state and cannot be deactivated.

activateAppInstGroup

Activates an Application Instance, which changes the state of the instance to ACTIVE.

Method Signature

```
activateAppInstGroup(appAcountId, appInstGroupId)
```

Input Parameters

Table C–9 describes the input parameters.

Table C–9 activateAppInstGroup Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Application Instance is not in an appropriate state to allow the account to be activated.

getAppInstGroupState

Gets the state of an Application Instance.

Method Signature

`getAppInstGroupState(`*`appAcountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–10 describes the input parameters.

Table C–10 getAppInstGroupState Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

Returns the state of the Application Instance. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

deactivateAppInstGroup

Deactivates an Application Instance, which changes the state of the group to INACTIVE.

Method Signature

`deactivateAppInstGroup(`*`appAcountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–11 describes the input parameters.

Table C–11 deactivateAppInstGroup Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The Application Instance is not in an appropriate state to allow the account to be deactivated.

registerAppInstGroupReq

Requests the registration of an Application Instance for a specific Application Account. When this request has been approved by the Operator (see "registerAppInstGroupRes") an application has all credentials necessary to be authenticated on the traffic interfaces of the Services Gatekeeper.

Method Signature

`registerAppInstGroup(`*`appAcountId`*`, `*`appInstGroupId`*`, `*`appInstGroup`*`, `*`password`*`)`

Input Parameters

Table C–12 describes the input parameters.

Table C–12 registerAppInstGroup Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance to be registered.
appInstGroup	tns1:AppInstGroup	The CRM/PRM application-defined properties in the form of name-value pairs. See "AppInstGroup".
password	xsd:string	The password the application will use when authenticating on Services Gatekeeper

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.
- **INVALID_STATE**: The application instance is in an invalid state and cannot be registered.

listAppInstGroups

Lists all Application Instances for an Application Account. Filtering is possible on the state of the Application Instance

Method Signature

`listAppInstGroup(appAcountId, state)`

Input Parameters

Table C–13 describes the input parameters.

Table C–13 listAppInstGroup Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
state	tns1:State	Indicates which states to filter the result on. See "State".

Return Parameters

Returns a list of all Service Application Instance IDs matching the given criteria in a string array.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppInstGroup

Retrieves details about a specific Application Instance.

Method Signature

`getAppInstGroup(appAcountId, appInstGroupId)`

Input Parameters

Table C–14 describes the input parameters.

Table C–14 getAppInstGroup Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance to be registered.

Return Parameters

Returns a data structure with details about the Application Instance. See "AppInstGroup".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getSpAccountSlaByType

Retrieves an SLA of a given type for the Service Provider Account.

Method Signature

```
getSpAccountSlaByType(slaType)
```

Where *slaType* is the SLA type to retrieve.

Use:

- service_provider
- system:geo_service_provider
- service_provider_node
- a custom SLA type ID

For information on the different types, see "*Managing SLAs*" in *Services Gatekeeper Accounts and SLAs Guide*.

Return Parameters

Returns the service provider SLA in a string.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppAccountSlaByType

Retrieves an SLA of a given type for the Application Account.

Method Signature

`getAppAccountSlaByType(slaType, appAccountId)`

Input Parameters

Table C–15 describes the input parameters.

Table C–15 getAppAccountSlaByType Input Parameters

Parameter Name	Type	Description
slaType	xsd:string	The SLA type to retrieve. Use: ■ application ■ system:geo_application ■ a custom SLA type ID For information on the different types, see "*Managing SLAs*" in *Services Gatekeeper Accounts and SLAs Guide*.
appAccountId	xsd:string	The Application Account to retrieve the SLA for.

Return Parameters

Returns the application level SLA in a string.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

setAppInstGroupPassword

Set the password associated with an Application Instance. This password is a part of the credentials an application uses to be authenticated on the traffic interfaces exposed by Services Gatekeeper.

Method Signature

```
setAppInstGroupPassword(appAccountId, appInstGroupId, newPassword)
```

Input Parameters

Table C–16 describes the input parameters.

Table C–16 setAppInstGroupPassword Input Parameters

Parameter Name	Type	Description
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.
newPassword	xsd:string	The new password.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

changeSpAccountPassword

Change the Service provider Account password. This password is the password the Service Provider use to login to the Service Provider part of the Partner Management Interfaces exposed by Services Gatekeeper.

Method Signature

`changeSpAccountPassword(`*`oldPassword`*`, `*`newPassword`*`)`

Input Parameters

Table C–17 describes the input parameters.

Table C–17 changeSpAccountPassword Input Parameters

Parameter Name	Type	Description
oldPassword	xsd:string	The current password.
newPassword	xsd:string	The new password.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

Service Provider Service Complex Data Types

This section describes the complex data types used by the Service Provider Service interface.

AppAccount

Table C–18 provides the description of an Application Account.

Table C–18 appAccount

Element name	Data Type	Description
Name	xsd:string	Descriptive name of the Application Account.
Description	xsd:string	Short description of the Application Account.
Properties	impl:ArrayOf_ tns1_Property	CRM/PRM application-defined name value pairs. See "Property".

AppInstGroup

Table C–19 describes the container for the CRM/PRM application-defined properties.

Table C–19 AppGroup

Element name	Data Type	Description
Name	xsd:string	Descriptive name of the Application Instance.
Description	xsd:string	Short description of the Application Instance.
SLA	xsd:string	The SLA for the Application Instance. Is always “1” as the SLA feature is no longer used.
Properties	mpl:ArrayOf_tns1_ Property	CRM/PRM application-defined properties. Name-value pairs. See "Property".

Property

Array of name-value pairs, as shown in Table C–20. This data type is used in several other data types specific for this interface. The properties are accessible from the Service Provider interface and the Operator interface, so they can be used for communicating information between the Service Provider and the Operator.

Table C–20 Property

Element name	Data Type	Description
Name	xsd:string	Name of the property, with the value defined in Value. Unique with the array.
Value	xsd:string	The data associated with Name.

SpAccount

Table C–21 provides the description of a Service Provider Account, including contact details.

Table C–21 appAccount

Element name	Data Type	Description
Name	xsd:string	Name of the Service Provider.

Table C–21 (Cont.) appAccount

Element name	Data Type	Description
Address	xsd:string	Address of the Service Provider.
EMailAddress	xsd:string	E-mail address of the Service Provider.
ContactPerson	xsd:string	Contact person at the Service provider.
PhoneNumber	xsd:string	Phone number to the Service Provider.
Properties	mpl:ArrayOf_tns1_Property	CRM/PRM application-defined name value pairs. See "Property".

State

Defines the state of a Service Provider Account, Service Provider Group, Application Account, Application Group, or Application Instance. Enumeration, as shown in Table C–22.

See "Account States" for more information about states, and transitions between different states.

Table C–22 State

Element name	Data Type	Description
REGISTERED	xsd:string	The account or group is has been registered. The Operator must respond to this registration request.
ACTIVE	xsd:string	Normal mode.
INACTIVE	N/A	No traffic is allowed through the Services Gatekeeper when the account or group is in this state.
UPDATE_PENDING	N/A	There is a pending update request. The request must be responded to by the Operator.
DELETE_PENDING	N/A	There is a pending delete request on the account or group. The request must be responded to by the Operator
LOCKED	N/A	Only valid for an Applicator Instances. For backward compatibility only. The group can be locked due to too many consecutive failed login attempts from an application.

Interface: OpService

The Operator Service Web Service provides the Operator with operations for handling Service Provider Accounts, Service Provider Groups, Application Accounts, Application Account Groups and Applications Instances in Services Gatekeeper's Partner Relationship Management module.

The endpoint for this interface is:

`http://host:port/prm_op/services/OpService`

The value of *host* and *port* depend on the Services Gatekeeper deployment.

OpService Error Messages

This interface generates these error messages:

- "The SP Account not exist"
- "Only DEACTIVE status can be deleted!"
- "the AppAcount existed already!"
- "SP no exist!"
- "Application no exist"

listAppGroups

Lists all Application Groups.

Method Signature

```
listAppGroups()
```

Input Parameters

This operation takes no input parameters.

Return Parameters

Returns a list of all application group IDs in a string array.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppGroup

Retrieves details about a specific Application Group.

Method Signature

`getAppGroup(`*`appGroupId`*`)`

Where *appGroupId* is the ID of the group.

Return Parameters

Returns a data structure with details about the Application Group. See "AppGroup".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

createAppGroupByType

Creates a new Application Group with a certain ID with an associated SLA.

Method Signature

```
createAppGroupByType(slaType, appGroupId, appGroup)
```

Input Parameters

Table C–23 describes the input parameters.

Table C–23 createAppGroupByType Input Parameters

Parameter Name	Type	Description
slaType	xsd:string	The SLA type to update. Use: ■ application ■ system:geo_application ■ a custom SLA type ID For information on the different types, see "Managing SLAs" in *Services Gatekeeper Accounts and SLAs Guide*.
appGroupId	xsd:string	ID of the Application Group.
appGroup	tns1:AppGroup	A data structure describing the group. See "AppGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

deleteAppGroup

Deletes an Application Group. All Application Accounts must be removed from the group before it can be deleted.

Method Signature

`deleteAppGroup(`*`appGroupId`*`)`

Where *appGroupId* is the ID of the group.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

moveAppAccountToGroup

Associates an Application Account for a specific Service Provider with an Application Group.

Method Signature

`moveAppAccountToGroup(spAccountId, appAccountId, appGroupId)`

Input Parameters

Table C–24 describes the input parameters.

Table C–24 moveAppAccountToGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appGroupId	xsd:string	The ID of the Application Account Group.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when access to the method is denied.
- **CommonException**: Exception thrown if none of the accounts or groups do not exist.

getAppGroupId

Retrieves the ID of the Application Group for a given Service Provider Account and Application Account combination.

Method Signature

`getAppGroupId(spAccountId, appAccountId)`

Input Parameters

Table C–25 describes the input parameters.

Table C–25 getAppGroupId Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if any of the accounts does not exist or no group is associated.

updateAppGroupByType

Updates an Application Group with a new SLA.

Method Signature

```
updateAppGroupByType(slaType, appGroupId, appGroup)
```

Input Parameters

Table C–26 describes the input parameters.

Table C–26 updateAppGroupByType Input Parameters

Parameter Name	Type	Description
slaType	xsd:string	The SLA type to update. Use: ■ application ■ system:geo_application ■ a custom SLA type ID For information on the different types, see “Managing SLAs” in Services Gatekeeper Accounts and SLAs Guide.
appGroupId	xsd:string	The ID of the group.
appGroup	tns1:AppGroup	The Application-level SLA, and CRM/PRM application-defined properties in the form of name-value pairs. See "AppGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

listSpGroups

Lists all Service Provider Groups. This operation requires no input.

Method Signature

```
updateAppGroupByType()
```

Input Parameters

This method accepts no parameters.

Return Parameters

Returns a list of all service provider group IDs in a string array.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getSpGroup

Retrieves details about a specific Service Provider Group.

Method Signature

```
updateAppGroupByType(spGroupId)
```

Where *spGroupId* is the ID of the group.

Return Parameters

Returns a data structure with details about the Service Provider Group. See "SpGroup".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

createSpGroupByType

Creates a new Service Provider Group with a certain ID with a certain SLA type associated.

Method Signature

```
createSpGroupByType(slaType, spGroupId, spGroup)
```

Input Parameters

Table C–27 describes the input parameters.

Table C–27 createSpGroupByType Input Parameters

Parameter Name	Type	Description
slaType	xsd:string	The SLA type to update. Use: ■ service_provider ■ system:geo_service_ provider ■ service_provider_node ■ a custom SLA type ID For information on the different types, see “Managing SLAs” in *Services Gatekeeper Accounts and SLAs Guide*.
spGroupId	xsd:string	ID of the Service Provider Group ID.
spGroup	tns1:SpGroup	A data structure describing the group. See "SpGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

deleteSpGroup

Deletes a Service Provider Group. All Service Provider Accounts associated with the group must be removed before it can be deleted

Method Signature

```
deleteSpGroup(spGroupId)
```

Where *spGroupId* is the ID of the group.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if there are Service Provider accounts associated with the group.

moveSpToGroup

Associates a Service Provider Account with a Service Provider Group.

Method Signature

`moveSpToGroup(spAccountId, spGroupId)`

Input Parameters

Table C–28 describes the input parameters.

Table C–28 moveSpToGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
spGroupId	xsd:string	The ID of the Service provider Group.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if any of the accounts does not exist.

getSpGroupId

Retrieves the ID of the Service Provider Group for a given Service Provider Account.

Method Signature

```
getSpGroupId(spAccountId)
```

Where *spAccountId* is the ID of the group.

Return Parameters

Returns the ID of the service provider group as a string.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the account does not exist.

updateSpGroupByType

Updates a Service Provider Group with a new SLA.

Method Signature

```
updateSpGroupByType(slaType, spGroupId, spGroup)
```

Input Parameters

Table C–29 describes the input parameters.

Table C–29 updateSpGroupByType Input Parameters

Parameter Name	Type	Description
slaType	xsd:String	The SLA type to update. Use: ■ service_provider ■ system:geo_service_ provider ■ service_provider_node ■ a custom SLA type ID For information on the different types, see “Managing SLAs” in Services Gatekeeper Accounts and SLAs Guide.
spGroupId	xsd:string	The ID of the group.
spGroup	tns1:SpGroup	The Service Provider-level SLA, and CRM/PRM application-defined properties in the form of name-value pairs. See "SpGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

listAppInstGroups

Lists all Application Instances for a given combination of Service Provider Account and Application Account. Filtering is possible on the state of the Application Instance

Method Signature

```
listAppInstGroups(spAccountId, appAccountId, state)
```

Input Parameters

Table C–30 describes the input parameters.

Table C–30 listAppInstGroups Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account. Use null to not filter on this parameter.
appAccountId	xsd:string	The ID of the Application Account. Use null to not filter on this parameter.
state	tns1:State	Indicates which states to filter the result on. See "State".

Return Parameters

Returns a data structure containing a references to the Application Instance. See "AppInstGroupRef".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppInstGroup

Retrieves details about a specific Application Instance.

Method Signature

```
getAppInstGroup(spAccountId, appAccountId, appInstGroupId)
```

Input Parameters

Table C–31 describes the input parameters.

Table C–31 getAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

Returns a data structure with details about the Application Instance. See "AppInstGroup".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

getAppInstGroupState

Gets the state of specific Application Instance.

Method Signature

`getAppInstGroupState(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–32 describes the input parameters.

Table C–32 getAppInstGroupState Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

Returns the state of the Application Instance. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

registerAppInstGroupReq

Requests registration of an Application Instance for a specific combination of a Service Provider Account and Application Account. When this request has been approved (see "registerAppInstGroupRes") the newly registered application has all the credentials necessary to be authenticated on the traffic interfaces of the Services Gatekeeper.

Method Signature

`registerAppInstGroupReq(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`, `*`appInstGroup`*`, `*`password`*`)`

Input Parameters

Table C–33 describes the input parameters.

Table C–33 registerAppInstGroupReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance to be registered.
appInstGroup	tns1:AppInstGroup	CRM/PRM application-defined properties in the form of name-value pairs.
password	xsd:string	The password the newly created Application Instance will use when authenticating on the Services Gatekeeper

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired or there are communication problems with the underlying platform.

deleteAppInstGroupReq

Requests deletion of an Application Instance. The Application Instance must be in state INACTIVE in order for this call to be accepted. The request must be approved before the Application Instance is deleted. This is done using "deleteAppInstGroupRes".

Method Signature

`deleteAppInstGroupReq(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–34 describes the input parameters.

Table C–34 deleteAppInstGroupReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if there is an SLA associated with the combination of Service Provider Accounts, Application Account, and Application Instance associated with the group.
- **INVALID_STATE**: This exception is raised if the application is not in an INACTIVE state.

deleteAppInstGroupRes

Responds to a request for deleting an Application Instance for a specific combination of Service Provider Account and Application Account.

It is possible to approve or disapprove the request. Both cases triggers a state transition for the Application Instance. If approved, the Application Instance is deleted, and the Application Instance can no longer be used to authenticate to send the traffic to Services Gatekeeper.

Method Signature

```
deleteAppInstGroupRes(spAccountId, appAccountId, appInstGroupId)
```

Input Parameters

Table C–35 describes the input parameters.

Table C–35 deleteAppInstGroupRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance

Return Parameters

See "RequestResponse" for details on the return parameters.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if there is an SLA associated with the combination of Service Provider Accounts, Application Account, and Application Instance associated with the group.
- **INVALID_STATE**: This exception is raised if the application is not in an INACTIVE state.

updateAppInstGroup

Updates an Application Instance Group with new data.

Method Signature

```
updateAppInstGroup(spAccountId, appAccountId, appInstGroupId, appInstGroup)
```

Input Parameters

Table C–36 describes the input parameters.

Table C–36 updateAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance
appInstGroup	tns1:AppInstGroup	A data-structure with the Application Instance SLA and application-defined properties in the form of name-value pairs. See "AppInstGroup".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

updateAppInstGroupRes

Responds to a request for updating Application Instance for a specific combination of Service Provider Account and Application Account.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Application Instance to state ACTIVE. If approved, the Application Instance is updated with the new information.

Method Signature

`updateAppInstGroupRes(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–37 describes the input parameters.

Table C–37 updateAppInstGroupRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance

Return Parameters

See "RequestResponse" for details on the return parameters.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

getUpdatePendingAppInstGroup

Gets details about an specific Application Instance that is in state UPDATE_ PENDING. The details include CRM/PRM application-defined properties in the form of name-value pairs.

Method Signature

```
getUpdatePendingAppInstGroup(spAccountId, appAccountId, appInstGroupId)
```

Input Parameters

Table C–38 describes the input parameters.

Table C–38 getUpdatePendingAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance

Return Parameters

Returns CRM/PRM application-defined properties in the form of name-value pairs. See "AppInstGroup".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.
- **INVALID_STATE**: This exception is raised if the application is not in an INACTIVE state.

setAppInstGroupPassword

Sets the password associated with an Application Instance. This password is a part of the credentials an application uses to authenticate to send traffic to Services Gatekeeper.

Method Signature

```
setAppInstGroupPassword(spAccountId, appAccountId, appInstGroupId, newPassowrd)
```

Input Parameters

Table C–39 describes the input parameters.

Table C–39 setAppInstGroupPassword Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance
newPassword	xsd:string	The new password.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

unlockAppInstGroup

Unlock a locked Application Instance Group. The group may have been locked by too many faulty login attempts to the traffic interfaces exposed by the Services Gatekeeper.

Method Signature

`unlockAppInstGroup(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–40 describes the input parameters.

Table C–40 unlockAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account
appInstGroupId	xsd:string	The ID of the Application Instance

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if group does not exist or if the SLA contains errors.

activateSpAccount

Activates a Service Provider Account, which changes the state of the account to ACTIVE.

Method Signature

`activateSpAccount(`*`spAccountId`*`)`

Input Parameters

Table C–41 describes the input parameters.

Table C–41 activateSpAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider Account is not in an appropriate state to allow the account to be activated.

deactivateSpAccount

Deactivates a a Service Provider Account, which changes the state of the account to inactive.

Method Signature

`deactivateSpAccount(`*`spAccountId`*`)`

Input Parameters

Table C–42 describes the input parameters.

Table C–42 deactivateSpAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider Account is not in an appropriate state to allow the account to be deactivated.

getSpAccount

Retrieves details about a specific Service Provider Account. The details include contact details and CRM/PRM application-defined properties in the form of name-value pairs.

Method Signature

`getSpAccount(`*`spAccountId`*`)`

Input Parameters

Table C–43 describes the input parameters.

Table C–43 getSpAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

Returns a structure with details on the Service Provider account. See "SpAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the account does not exist.

getSpAccountState

Retrieves the state of a specific Service Provider Account.

Method Signature

`getSpAccountState(`*`spAccountId`*`)`

Input Parameters

Table C–44 describes the input parameters.

Table C–44 getSpAccountState Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

Returns information on the state of the Service Provider account. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the account does not exist.

registerAppAccountReq

Requests registration for an Application Account for a specific Service Provider Account. When this request has been approved by the Operator (see "registerAppAccountRes") the Application Account can be associated with an Application Account Group and an Application Instance.

Method Signature

```
registerAppAccountReq(spAccountId, appAccountId, appAccount)
```

Input Parameters

Table C–45 describes the input parameters.

Table C–45 registerAppAccountReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appAccount	tns1:AppAccount	A data structure with details on the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "AppAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: This exception is raised if the application is in an invalid state.

registerAppAccountRes

Responds to a request to register an Application Account for a specific Service Provider Account. An Application Account Group is also associated with the Application Account.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Application Account. If approved, the Application Account is transferred into state ACTIVE. If Disapproved, the Application Account is deleted.

Method Signature

`registerAppAccountRes(spAccountId, appAccountId, appGroupId, appAccountRef)`

Input Parameters

Table C–46 describes the input parameters.

Table C–46 registerAppAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appGroupId	xsd:string	The ID of the Application Account Group to be associate with the Application Account.
appAccountRef	xsd:string	Internal ID of the Application Account. This ID is used to correlate the Application Account ID with an Operator-internal ID.

Return Parameters

See "RequestResponse" for details on the return parameters.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: This exception is raised if the application is in an invalid state.

updateAppAccountRes

Responds to a request to update an Application Account for a specific Service Provider.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Application Account state to Active. If approved, the Application Account is updated with the new information.

Method Signature

`updateAppAccountRes(`*`spAccountId`*`, `*`appAccountId`*`)`

Input Parameters

Table C–47 describes the input parameters.

Table C–47 updateAppAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

See "RequestResponse" for details on the return parameters.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Application Account is not in state UPDATE_PENDING.

getUpdatePendingAppAccount

Gets details about pending update requests for a specific combination of Service Provider and Application Account. The details includes descriptions and CRM/PRM application-defined properties in the form of name-value pairs. Valid only for Application Accounts in state UPDATE_PENDING.

Method Signature

```
getUpdatePendingAppAccount(spAccountId, appAccountId)
```

Input Parameters

Table C–48 describes the input parameters.

Table C–48 getUpdatePendingAppAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

Returns a data structure describing the Application Account. See "AppAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Application Account is not in state UPDATE_PENDING.

deleteAppAccountReq

Requests the deletion of an Application Account. In order to be deleted, there must be no Application Instance associated with the combination of Service Provider Account and Application Account. The request must be approved before the Application Account is deleted, which is done using "deleteAppAccountRes". The Application Account must be in state INACTIVE in order for this call to be accepted.

Method Signature

`deleteAppAccountReq(`*`spAccountId`*`, `*`appAccountId`*`)`

Input Parameters

Table C–49 describes the input parameters.

Table C–49 deleteAppAccountReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Application Account is not in state INACTIVE.

deleteAppAccountRes

Responds to a request to delete an Application Account for a specific Service Provider Account. The Application Account must be in state DELETE_PENDING in order for this call to be accepted.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Application Account. If approved, the Application Account is simply deleted. If Disapproved, the Application Account is transferred into state INACTIVE.

Method Signature

`deleteAppAccountRes(`*`spAccountId`*`, `*`appAccountId`*`)`

Input Parameters

Table C–50 describes the input parameters.

Table C–50 deleteAppAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Application Account is not in state DELETE_PENDING.

updateAppAccount

Updates an Application Account with new data.

Method Signature

`updateAppAccount(spAccountId, appAccountId)`

Input Parameters

Table C–51 describes the input parameters.

Table C–51 updateAppAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appAccount	tns1:AppAccount	Data structure with details on the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "AppAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Application Account is in an invalid state.

getAppAccount

Retrieves details about a specific Application Account. The return includes a data structure with details on the Application Account, including CRM/PRM application-defined properties in the form of name-value pairs.

Method Signature

`getAppAccount(spAccountId, appAccountId)`

Input Parameters

Table C–52 describes the input parameters.

Table C–52 getAppAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

Returns a data structure with details about the Application Account. See "AppAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.

getAppAccountState

Retrieves the state of a specific Application Account.

Method Signature

```
getAppAccountState(spAccountId, appAccountId)
```

Input Parameters

Table C–53 describes the input parameters.

Table C–53 getAppAccountState Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

Returns information of the state of the Application Account. See "State".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.

activateAppAccount

Activates an Application Account, which changes the state of the account to ACTIVE. The current state of the account must be INACTIVE.

Method Signature

`activateAppAccount(`*`spAccountId`*`, `*`appAccountId`*`)`

Input Parameters

Table C–54 describes the input parameters.

Table C–54 activateAppAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.
- **INVALID_STATE**: Exception thrown if the state of the Application Account does not allow the account to be activated.

deactivateAppAccount

Deactivates an Application Account, which changes the state of the account to INACTIVE. The current state of the account must be ACTIVE.

Method Signature

```
deactivateAppAccount(spAccountId, appAccountId)
```

Input Parameters

Table C–55 describes the input parameters.

Table C–55 deactivateAppAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application account does not exist.
- **INVALID_STATE**: Exception thrown if the state of the Application Account does not allow the account to be deactivated.

registerAppInstGroupRes

Responds to a request to register an Application Instance for a specific combination of a Service Provider Account and Application Group.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Application Instance. If approved, the Application Instance is transferred into state ACTIVE and the application can authenticate with the traffic interfaces exposed by Services Gatekeeper. If disapproved, the Application Instance is deleted.

Method Signature

```
registerAppInstGroupRes(spAccountId, appAccountId, appInstGroupId,
appInstGroupRef, response)
```

Input Parameters

Table C–56 describes the input parameters.

Table C–56 registerAppInstGroupRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.
appInstGroupRef	xsd:string	Internal ID of the Application Instance. This ID is used to correlate the Application Instance ID with an Operator-internal ID.
response	tns1:RequestResponse	The response to the request. See "RequestResponse".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application instance does not exist.
- **INVALID_STATE**: Exception thrown if the state of the Application instance is invalid.

activateAppInstGroup

Activates an Application Instance, which changes the state of the Instance to ACTIVE.

Method Signature

```
activateAppInstGroup(spAccountId, appAccountId, appInstGroupId)
```

Input Parameters

Table C–57 describes the input parameters.

Table C–57 activateAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application instance does not exist.
- **INVALID_STATE**: Exception thrown if the state of the Application instance is invalid.

deactivateAppInstGroup

Deactivates an Application Instance, which changes the state of the Instance to INACTIVE.

Method Signature

`deactivateAppInstGroup(`*`spAccountId`*`, `*`appAccountId`*`, `*`appInstGroupId`*`)`

Input Parameters

Table C–58 describes the input parameters.

Table C–58 deactivateAppInstGroup Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
appInstGroupId	xsd:string	The ID of the Application Instance.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Application instance does not exist.
- **INVALID_STATE**: Exception thrown if the state of the Application instance does not allow the account to be deactivated.

registerSpAccountReq

Requests registration for a Service Provider Account. Contact details are supplied in the request, together with CRM/PRM application-defined properties. This request must be approved by the Operator (see "registerAppAccountRes").

Method Signature

```
registerSpAccountReq(spAccountId, appAccountId, password)
```

Input Parameters

Table C–59 describes the input parameters.

Table C–59 registerSpAccountReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
appAccountId	xsd:string	The ID of the Application Account.
password	xsd:string	The password the Service Provider will use when authenticating to the Service Provider part of the Partner Relationship Management interface.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider account is not in the correct state.

listAppAccounts

Lists all Application Account IDs for a specific Service Provider. The result is filtered on the state of the Application Account.

Method Signature

`listAppAccounts(spAccountId, state)`

Input Parameters

Table C–60 describes the input parameters.

Table C–60 listAppAccounts Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
state	tns1:State	Input parameter. Indicates the state on which to filter the result. See "State".

Return Parameters

Returns an array containing references to Application Account Groups. See "AppAccountRef".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if no Application account IDs exist.

listSpAccounts

Lists all Service Provider Account IDs. The result is filtered on the state of state of the Service Provider Account.

Method Signature

`listAppAccounts(`*`state`*`)`

Input Parameters

Table C–61 describes the input parameters.

Table C–61 listAppAccounts Input Parameters

Parameter Name	Type	Description
state	tns1:State	Input parameter. Indicates the state on which to filter the result. See "State".

Return Parameters

Returns an array of Service Provider account IDs.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if no Service Provider accounts exist.

registerSpAccountRes

Responds to a request to register a Service Provider Account.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Service Provider Account. If approved, the Service Provider Account is transferred into state ACTIVE and the Service provider can authenticate with the PRM-SP Web Services. If Disapproved, the Service Provider Account is deleted.

Method Signature

```
registerSpAccountRes(spAccountId, spGroupId, spAcccountRef, response)
```

Input Parameters

Table C–62 describes the input parameters.

Table C–62 registerSpAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
spGroupId	xsd:string	The ID of the Service Provider Group the Service Provider Account should be associated with.
spAccountRef	xsd:string	Internal ID of the Service Provider. This ID is used to correlate the Service Provider ID with an Operator-internal ID.
response	tns1:RequestResponse	The response to the request. See "RequestResponse".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Service Provider account is not in state REGISTERED.

deleteSpAccountReq

Requests deletion of a Service Provider Account. In order to be deleted, the Service Provider Account must be state INACTIVE. The request must be approved before the Service Provider Account it is deleted. This is done using "deleteSpAccountRes".

Method Signature

`deleteSpAccountReq(spAccountId)`

Input Parameters

Table C–63 describes the input parameters.

Table C–63 deleteSpAccountReq Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Service Provider account is not in state INACTIVE.

deleteSpAccountRes

Responds to a request to delete a Service Provider Account. The Service Provider Account must be in state DELETE_PENDING in order for this call to be accepted.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Service Provider Account. If approved, the Service Provider Account is simply deleted. If disapproved, the Service Provider Account is transferred into state INACTIVE.

Method Signature

`deleteSpAccountRes(spAccountId, response)`

Input Parameters

Table C–64 describes the input parameters.

Table C–64 deleteSpAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
response	tns1:RequestResponse	The response to the request. See "RequestResponse".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the status of the Service Provider account is not in state INACTIVE.

updateSpAccount

Updates a Service Provider Account with new data.

Method Signature

```
updateSpAccount(spAccountId, spAccount)
```

Input Parameters

Table C–65 describes the input parameters.

Table C–65 updateSpAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
spAccount	tns1:SpAccount	Data structure with details on the Service Provider Account, including CRM/PRM application-defined properties in the form of name-value pairs. See "SpAccount".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider account is not in a valid state.

updateSpAccountRes

Responds to a request to update a Service Provider Account.

It is possible to approve or disapprove the request. Both cases trigger a state transition for the Service Provider Account to state ACTIVE. If approved, the Service Provider Account is updated with the new information.

Method Signature

`updateSpAccountRes(spAccountId, response)`

Input Parameters

Table C–66 describes the input parameters.

Table C–66 updateSpAccountRes Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
response	tns1:RequestResponse	The response to the request. See "RequestResponse".

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider account is in state INACTIVE.

getUpdatePendingSpAccount

Gets details about a specific Service Provider account that is in state UPDATE_PENDING. The details include contact information and CRM/PRM application-defined properties in the form of name-value pairs.

Method Signature

`getUpdatePendingSpAccount(`*`spAccountId`*`)`

Input Parameters

Table C–67 describes the input parameters.

Table C–67 getUpdatePendingSpAccount Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.

Return Parameters

Returns details about contact information and CRM/PRM application-defined properties in the form of name-value pairs. See "SpAccount".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.
- **INVALID_STATE**: Exception thrown if the Service Provider account is not in state UPDATE_PENDING.

setSpAccountPassword

Sets the password the Service Provider uses to authenticate to use the Partner Relationship Management Interface exposed by Services Gatekeeper.

Method Signature

`setSpAccountPassword(spAccountId, newPassword)`

Input Parameters

Table C–68 describes the input parameters.

Table C–68 setSpAccountPassword Input Parameters

Parameter Name	Type	Description
spAccountId	xsd:string	The ID of the Service Provider Account.
newPassword	xsd:string	The new password.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service Provider account does not exist.

changeOpAccountPassword

Changes the password the Operator uses to authenticate with the Partner Relationship Management Interface exposed by Services Gatekeeper. The Operator Account is the one the Operator is currently logged in as.

Method Signature

`changeOpAccountPassword(`*`oldPassword, newPassword`*`)`

Input Parameters

Table C–69 describes the input parameters.

Table C–69 changeOpAccountPassword Input Parameters

Parameter Name	Type	Description
oldPassword	xsd:string	The password to be changed.
newPassword	xsd:string	The new password.

Return Parameters

None

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Operator account does not exist.

getUserLevel

Retrieves the user level of the currently logged in Operator Account. Different user levels have different privileges, and are authorized to different sets of operations. This operation takes no input.

Method Signature

```
getUserLevel()
```

Input Parameters

This method takes no input.

Return Parameters

Returns the user level of the currently logged in Operator Account. See "UserLevel".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Data Types

These complex data types are used by this interface.

AppAccount

Table C–70 describes of an Application Account.

Table C–70 appAccount

Element name	Data Type	Description
name	xsd:string	Descriptive name of the Application Account.
description	xsd:string	Short description of the Application Account.
properties	impl:ArrayOf_ tns1_Property	CRM/PRM application-defined name value pairs. See "Property".

AppAccountRef

Table C–71 lists the reference to the IDs of an Application Account.

Table C–71 AppInstGroupRef

Element name	Data Type	Description
spAccountId	xsd:string	ID of the Service Provider Account associated with the Application Account.
appAccountId	xsd:string	ID of the Application Account.

AppInstGroupRef

Table C–72 lists the reference to IDs of an Application Instance.

Table C–72 AppInstGroupRef

Element name	Data Type	Description
spAccountId	xsd:string	ID of the Service Provider Account associated with the Application Instance.
appAccountId	xsd:string	ID of the Application Account associated with the Application Instance.
appInstGroupId	xsd:string	ID of the Application Instance.

AppGroup

Table C–73 describes the container for Application-level SLA and CRM/PRM application-defined properties.

Table C–73 AppGroup

Element name	Data Type	Description
sla	xsd:string	The SLA for the Application Group.
properties	mpl:ArrayOf_tns1_ Property	CRM/PRM application-defined properties. Name-value pairs. See "Property".

AppInstGroup

Table C–74 describes the container for Application Instance CRM/PRM application-defined properties.

Table C–74 AppInstGroup

Element name	Data Type	Description
name	xsd:string	Descriptive name of the Application Instance.
description	xsd:string	Short description of the Application Instance.
sla	xsd:string	The SLA for the Application Instance. Always set to 1.
properties	mpl:ArrayOf_tns1_ Property	CRM/PRM application-defined properties. Name-value pairs. See "Property".

Property

Array of name-value pairs, as seen in Table C–75. This data type is used in several other data types specific to this interface. The properties are accessible from the Service Provider interface and the Operator interface, so they can be used to communicate information between them.

Table C–75 Property

Element name	Data Type	Description
name	xsd:string	Name of the property, with the value defined in Value. Unique with the array.
value	xsd:string	The data associated with Name.

RequestResponse

Table C–76 lists the numeration defining the operator's response to an request .

Table C–76 RequestResponse

Element name	Data Type	Description
APPROVE	xsd:string	Used when the operator approves the request.
DISAPPROVE	xsd:string	Used when the operator disapproves the request.

SpAccount

Table C–77 provides the description of a Service Provider Account, including contact details.

Table C–77 appAccount

Element name	Data Type	Description
name	xsd:string	Name of the Service Provider.
address	xsd:string	Address of the Service Provider.
eMailAddress	xsd:string	E-mail address of the Service Provider.
contactPerson	xsd:string	Contact person at the Service provider.
phoneNumber	xsd:string	Phone number to the Service Provider.

Table C–77 (Cont.) appAccount

Element name	Data Type	Description
properties	mpl:ArrayOf_tns1_ Property	CRM/PRM application-defined name value pairs. See "Property".

SpGroup

Table C–78 describes the container for Service Provider SLA and CRM/PRM application-defined properties.

Table C–78 SpGroup

Element name	Data Type	Description
sla	xsd:string	The SLA for the Service Provider.
properties	impl:ArrayOf_ tns1_Property	CRM/PRM Application-defined properties. Name-value pairs. See "Property".

UserLevel

Table C–79 lists the enumeration defining the user level of the currently logged in Operator user.

The user level reflects the user levels defined for the operations and maintenance of Services Gatekeeper. Each operation performed by using the Partner Management Interface results in one or more standard OAM operations. The user level of the currently authenticated user must satisfy the user level necessary for each of these operations. If this is not the case, the operation performed through the Partner Management Interface is denied.

Table C–79 UserLevel

Element name	Data Type	Description
UNAUTHORIZED	xsd:string	The currently authenticated in user is not authorized to perform any OAM operations.
READ_ONLY	xsd:string	The currently authenticated user is authorized to perform OAM read- or get- operations.
READ_WRITE	xsd:string	The currently authenticated user is authorized to perform OAM write- or set- operations.
ADMINISTRATOR	xsd:string	The currently authenticated user is authorized to perform administrator OAM operations tasks

State

Table C–80 defines the state of a Service Provider Account, Service Provider Group, Application Account, Application Group, or Application Instance. Enumeration.

See "Account States" for more information about states, and transitions among different states.

Table C–80 State

Element name	Data Type	Description
REGISTERED	xsd:string	The account or group is has been registered. The registration request must be responded to by the Service Provider.

Table C–80 (Cont.) State

Element name	Data Type	Description
ACTIVE	xsd:string	Normal mode.
INACTIVE	N/A	No traffic is allowed through the Services Gatekeeper when the account or group is in this state.
UPDATE_PENDING	N/A	There is a pending update request. The request must be responded to by the Service Provider.
DELETE_PENDING	N/A	There is a pending delete request on the account or group. The request must be responded to by the Service Provider.
LOCKED	N/A	Only valid for an Application Instance. The group can be locked due to too many consecutive failed login attempts from an application.

Interface: SpCdrUtil

The Service Provider Utility Web Service allows the Service Provider to retrieve the Call Detail Records (CDRs) it has generated from Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_sp/services/SpCdrUtil`

where the value of *host* and *port* depend on the Services Gatekeeper deployment.

countCdrs

Counts the number of CDRs for a certain Service for a specified time interval.

> **Note:** A Service is the generic name for a Services Gatekeeper communication service, without regard for the Web Service version or the network plug-in being used. So, for example, the Service name for Parlay X 2.1 Third Party Call using Session Initiation Protocol (SIP) or Intelligent Network Application Part (INAP) or Parlay X 3.0 using Parlay 3.3 MultiProtocol Communication Controller (MPCC) is simply Third Party Call.

Method Signature

```
countCdrs(serviceName, fromDate, toDate, completionStatus, appAccountId)
```

Input Parameters

Table C–81 describes the input parameters.

Table C–81 countCdrs Input Parameters

Parameter Name	Type	Description
serviceName	xsd:string	The name of the Service for which to retrieve charge data records (CDRs). Use null to not filter on this parameter
fromDate	xsd:dateTime	From date and time. Use null to not filter on this parameter
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR. See "CdrCompletionStatus". Use null to not filter on this parameter.
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter

Return Parameters

Returns the number of CDRs matching the given criteria.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service does not exist.

listCdrs

Retrieves all CDRs matching the given criteria.

Method Signature

```
listCdrs(serviceName, fromDate, toDate, completionStatus, appAccountId,
startIndex, maxEntries)
```

Input Parameters

Table C–82 describes the input parameters.

Table C–82 listCdrs Input Parameters

Parameter Name	Type	Description
serviceName	xsd:string	The name of the Service for which to retrieve charge data records (CDRs). Use null to not filter on this parameter
fromDate	xsd:dateTime	From date and time. Use null to not filter on this parameter
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR. See "CdrCompletionStatus". Use null to not filter on this parameter.
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter
startIndex	xsd:long	Input parameter. Which entry, in the overall result set, to start the result list on (cursor).
maxEntries	xsd:int	Input parameter. The maximum number of alarms returned.

Return Parameters

Returns a list of CDRS. See "CdrInfo".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service does not exist.

Data Types

These complex data types are used by this interface.

CdrInfo

Table C–83 lists the data structure defining a CDR. All services that produce charging data do not use all fields, and they use the fields in a slightly different, depending on the type of the service. See *Services Gatekeeper Communication Service Reference Guide* for details of which fields that are relevant for the different services.

Table C–83 CdrInfo

Element name	Data Type	Description
transactionId	xsd:long	The Services Gatekeeper transaction sequence number.
serviceName	xsd:string	The communication service whose use is being tracked
timeStamp	xsd:dateTime	The time at which the event was triggered (in milliseconds from midnight 1 January 1970)
origAddr	xsd:string	The address of the originating party.
destAddr	xsd:string	The address of the destination party.
spAccountD	xsd:string	The ID of the Service Provider that generated the CDR.
appAccountID	xsd:string	The ID of the Application Account that generated the CDR.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR. See "CdrCompletionStatus".
info	xsd:string	Additional info provided by the communication service
additionalProperties	impl:ArrayOf_tns1_Property	Application defined data. See "CdrCompletionStatus".

CdrCompletionStatus

Table C–84 lists the enumeration defining the completion status of a CDR.

Table C–84 CdrCompletionStatus

Element name	Data Type	Description
COMPLETED	xsd:string	The operation generating the CDR succeeded.
FAILED	xsd:string	The operation generating the CDR failed.
PARTIAL	xsd:string	The operation generating the CDR succeeded partially. May be supported, depending on the communication service.
COMPLETED_NOTIFICATION_FAILED	xsd:string	The CDR is completed, but the notification was not delivered to the application.
POLICY_DENIED	xsd:string	Policy denied the operation.

Property

Table C–85 lists the array of name-value pairs.

Table C–85 Property

Element name	Data Type	Description
Name	xsd:string	Name of the property, with the value defined in Value. Unique within the array.
Value	xsd:string	The data associated with Name.

Interface: SpStatisticsUtil

The Service Provider Statistics Utility Web Service allows the Service Provider to retrieve the statistics it has generated from Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_sp/services/SpStatisticsUtil`

Where the value of *host* and *port* depend on the Services Gatekeeper deployment.

listStatisticTypes

Lists the statistics types registered in Services Gatekeeper. This operation takes no input parameters.

Method Signature

```
listStatisticTypes()
```

Input Parameters

This method takes no input.

Return Parameters

Returns the descriptions of the available statistics types. See "StatisticTypeDescriptor".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

getStatistics

Retrieves statistics matching the given criteria.

Method Signature

`getStatistics(statisticType, fromDate, toDate, appAccountId)`

Input Parameters

Table C–86 describes the input parameters.

Table C–86 getStatistics Input Parameters

Parameter Name	Type	Description
statisticType	xsd:int	Number representing the type of statistics to retrieve. Use null to not filter on this parameter
fromDate	xsd:dateTime	From date and time. Use null to not filter on this parameter.
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter.

Return Parameters

Returns statistics. See "StatisticsInfo".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Data Types

These complex data types are used by this interface.

StatisticsInfo

Table C–87 lists the data structure defining a statistics record.All services that produces statistics do not use all fields, and they use the fields in a slightly different, depending on the type of the service. See *Services Gatekeeper Communication Service Reference Guide* for details of which fields that are relevant for the different services.

Table C–87 StatisticsInfo

Element name	Data Type	Description
statisticsType	xsd:string	The statistics type. See WebLogic Product Description for information on available statistics types.
timeStampStart	xsd:dateTime	The starting time of the interval during which the statistics were gathered.
timeStampEnd	xsd:dateTime	The end time of the interval during which the statistics were gathered.
numberOfTransactions	xsd:int	The number of transactions during the interval.
spAccountD	xsd:string	The ID of the Service Provider that generated the statistics.
appAccountID	xsd:string	The ID of the Application Account that generated the statistics.

StatisticTypeDescriptor

Table C–88 describes the data structure that holds a description for each type of statistics.

Table C–88 StatisticTypeDescriptor

Element name	Data Type	Description
transactionTypeName	xsd:string	Name of a transaction type that statistics can be generated for.
transactionTypeID	xsd:int	Numeric ID of the transaction type.

Interface: OpCdrUtil

The Operator CDR Utility Web Service allows the Operator to retrieve call details records (CDRs) from Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_op/services/OpCdrUtil`

where the value of *host* and *port* depend on the Services Gatekeeper deployment.

OpCdrUtil Error Messages

This interface can generate these error messages:

- listCdr failed

countCdrs

Counts the number of CDRs for a certain Service for a specified time interval

> **Note:** A Service is the generic name for a Services Gatekeeper communication service, without regard for the Web Service version or the network plug-in being used. So, for example, the Service name for Parlay X 2.1 Third Party Call using SIP or INAP or Parlay X 3.0 using Parlay 3.3 MPCC is simply Third Party Call.

Method Signature

```
countCdrs(serviceName, fromDate, toDate, completionStatus, spAccountId,
appAccountId)
```

Input Parameters

Table C–89 describes the input parameters.

Table C–89 countCdrs Input Parameters

Parameter Name	Type	Description
serviceName	xsd:string	The name of the Service for which to retrieve charge data records (CDRs). Use null to not filter on this parameter
fromDate	xsd:dateTime	From date and time. Use null to not filter on this parameter
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR. See "CdrCompletionStatus". Use null to not filter on this parameter.
spAccountId	xsd:string	Input parameter. ID of the Service Provider Account to filter the result on. Use null to not filter on this parameter.
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter

Return Parameters

Returns the number of CDRs matching the given criteria.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: Exception thrown if the Service does not exist.

listCdrs

Retrieves all CDRs matching the given criteria.

Method Signature

```
listCdrs(serviceName, fromDate, toDate, completionStatus, spAccountId,
appAcountId, startIndex, maxEntries)
```

Input Parameters

Table C–90 describes the input parameters.

Table C–90 listCdrs Input Parameters

Parameter Name	Type	Description
serviceName	xsd:string	The name of the Service for which to retrieve CDRs. Use null to not filter on this parameter
fromDate	xsd:dateTime	From the date and time. Use null to not filter on this parameter
toDate	xsd:dateTime	To the date and time. Use null to not filter on this parameter.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR.See "CdrCompletionStatus". Use null to not filter on this parameter.
spAccountId	xsd:string	ID of the Service Provider Account to filter the result on. Use null to not filter on this parameter.
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter.
startIndex	xsd:long	Which entry, in the overall result set, to start the result list on (cursor).
maxEntries	xsd:int	The maximum number of alarms returned.

Return Parameters

Returns a list of CDRS. See "CdrInfo".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Data Types

These are complex data types used by this interface.

CdrInfo

Table C–91 describes the data structure defining a CDR. All services that produce charging data do not use all fields, and they use the fields slightly different, depending on the type of the service. See *Services Gatekeeper Communication Service Reference Guide*for details of which fields that are relevant for the different services.

Table C–91 CdrInfo

Element name	Data Type	Description
transactionId	xsd:long	The Services Gatekeeper transaction sequence number
serviceName	xsd:string	The communication service whose use is being tracked
timeStamp	xsd:dateTime	The time at which the event was triggered (in milliseconds from midnight 1 January 1970
origAddr	xsd:string	The address of the originating party.
destAddr	xsd:string	The address of the destination party.
spAccountId	xsd:string	The ID of the Service Provider that generated the CDR.
appAccountId	xsd:string	The ID of the Application Account that generated the CDR.
completionStatus	tns1:CdrCompletionStatus	Completion status of the CDR. See "CdrCompletionStatus".
info	xsd:string	Additional info provided by the service capability module.
additionalProperties	impl:ArrayOf_tns1_Property	Application defined data. See "Property".

CdrCompletionStatus

Table C–92 lists the enumeration that defines the completion status of a CDR.

Table C–92 CdrCompletionStatus

Element name	Data Type	Description
COMPLETED	xsd:string	The operation generating the CDR succeeded.
FAILED	xsd:string	The operation generating the CDR failed
PARTIAL	xsd:string	The operation generating the CDR succeeded partially. May be supported, depending on the communication service.
COMPLETED_NOTIFICATION_FAILED	xsd:string	The CDR is completed, but the notification was not delivered to the application.
POLICY_DENIED	xsd:string	Policy denied the operation.

Property

Table C–93 describes the array of name-value pairs.

Table C–93 Property

Element name	Data Type	Description
Name	xsd:string	Name of the property, with the value defined in Value. Unique within the array.
Value	xsd:string	The data associated with Name.

Interface: OpStatisticsUtil

The Operator Statistics Utility Web Service allows the Operator to retrieve the statistics generated in Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_op/services/OpStatisticsUtil`

where the value for *host* and *port* depend on the Services Gatekeeper deployment.

OpStatisticsUtil Error Messages

This interface can generate these error messages

- "getStatistics failed!"
- "listStatisticTypes failed"

listStatisticTypes

Lists the statistics types registered in Services Gatekeeper. This operation takes no input.

Method Signature

```
listStatisticTypes()
```

Input Parameters

This method takes no input.

Return Parameters

Returns the descriptions of the available statistics types. See "StatisticTypeDescriptor".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

getStatistics

Retrieve statistics matching the given criteria.

Method Signature

```
getStatistics(statisticType, fromDate, toDate, spAccountId, appAccountId)
```

Input Parameters

Table C–94 describes the input parameters.

Table C–94 getStatistics Input Parameters

Parameter Name	Type	Description
statisticType	xsd:int	Number representing the type of statistics to retrieve. Use null to not filter on this parameter
fromDate	xsd:dateTime	From date and time. Use null to not filter on this parameter.
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
spAccountId	xsd:string	Input parameter. ID of the Service Provider Account to filter the result on. Use null to not filter on this parameter
appAccountId	xsd:string	ID of the Application Account to filter the result on. Use null to not filter on this parameter.

Return Parameters

Returns statistics. See "StatisticsInfo".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Exceptions

This exception is thrown by this interface.

CommonException

This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Data Types

These complex data types are used by this interface.

StatisticsInfo

Table C–95 describes the data structure defining a statistics record. All services that produce statistics do not use all fields, and they use the fields slightly differently, depending on the type of the service.

Table C–95 StatisticsInfo

Element name	Data Type	Description
statisticsType	xsd:string	The statistics type.
timeStampStart	xsd:dateTime	The starting time of the interval during which the statistics were gathered.
timeStampEnd	xsd:dateTime	The end time of the interval during which the statistics were gathered.
numberOfTransactions	xsd:int	The number of transactions during the interval.
spAccountId	xsd:string	The ID of the Service Provider that generated the statistics.
appAccountId	xsd:string	The ID of the Application Account that generated the statistics.

StatisticTypeDescriptor

Table C–96 defines the type of statistics that can be generated.

Table C–96 StatisticTypeDescriptor

Element name	Data Type	Description
transactionTypeName	xsd:string	Name of a transaction type that statistics can be generated for.
transactionTypeId	xsd:int	Numeric ID of the transaction type.

Interface: OpAlarmUtil

The Operator Alarm Utility Web Service allows the Operator to retrieve alarms from Services Gatekeeper.

The endpoint for this interface is:

`http://host:port/prm_op/services/OpAlarmUtil`

where the value of *host* and *port* depend on the Services Gatekeeper deployment.

countAlarms

Counts the number of alarms of a certain type of a given severity for a specified time interval.

Method Signature

```
countAlarms(alarmId, severity, fromDate, toDate)
```

Input Parameters

Table C–97 describes the input parameters.

Table C–97 countAlarms Input Parameters

Parameter Name	Type	Description
alarmId	xsd:int	The ID of the type of alarm.
severity	tns1:AlarmSeverity	Severity of the alarm. See "AlarmSeverity". Use null to not filter on this parameter
fromDate	xsd:dateTime	Start date and time. Use null to not filter on this parameter.
toDate	xsd:dateTime	End date and time. Use null to not filter on this parameter.

Return Parameters

Returns the number of alarms.

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

listAlarms

Retrieves all alarms matching the given criteria.

Method Signature

`listAlarms(alarmId, severity, fromDate, toDate, startIndex, maxEntries)`

Input Parameters

Table C–98 describes the input parameters.

Table C–98 listAlarms Input Parameters

Parameter Name	Type	Description
alarmId	xsd:int	The ID of the type of alarm.
severity	tns1:AlarmSeverity	Severity of the alarm. See "AlarmSeverity". Use null to not filter on this parameter.
fromDate	xsd:dateTime	From date and time.Use null to not filter on this parameter.
toDate	xsd:dateTime	To date and time. Use null to not filter on this parameter.
startIndex	xsd:long	Which entry, in the overall result set, to start the result list on (cursor).
maxEntries	xsd:int	The maximum number of alarms returned.

Return Parameters

Returns the list of alarms. See "AlarmInfo".

Exceptions

The possible exceptions are:

- **ACCESS_DENIED**: Exceptions of this type are raised when the operation is not permitted. The user does not have the appropriate privilege level to perform the operation.
- **CommonException**: This exception is raised when the login session has expired (BC only) or there are communication problems with the underlying platform.

Data Types

These complex data types are used by this interface.

AlarmInfo

Table C–99 describes the data structure defining an alarm.

Table C–99 AlarmInfo

Element name	Data Type	Description
alarmInstanceId	xsd:long	The ID of the emitted alarm. Unique identifier for an emitted alarm.
alarmId	xsd:int	The identifier for the alarm type.
source	xsd:string	Specifies the name of the software module that raised the alarm and the IP address of the server the service is installed in.
severity	tns1:AlarmSeverity	Specifies the alarm's severity level. See "AlarmSeverity".
timeStamp	xsd:dateTime	Specifies the time and date the alarm was raised.
info	xsd:string	Alarm information provided by the software module the raised the alarm.
additional_info	xsd:string	Additional information depending on context.

AlarmSeverity

Table C–100 defines the severity of an alarm.

Table C–100 AlarmSeverity

Element name	Data Type	Description
WARNING	xsd:string	Severity level is Warning.
MINOR	xsd:string	Severity level is Minor.
MAJOR	xsd:string	Severity level is Major.
CRITICAL	xsd:string	Severity level is Critical.