CHAPTER 2

Introducing Cisco RESTful and SOAP-based APIs

Overview

Cisco Prime Service Catalog supports northbound integration via a set of RESTful APIs for submitting service orders or requisitions, and accessing entities defined in the service catalog. The service catalog also has a set of legacy SOAP-based APIs for submitting and managing requisitions.

Cisco offers a set of standard REST (Representational State Transfer) APIs and Java stubs for accessing entities defined in Service Catalog. They are collectively known as nsAPI. nsAPI callers have to first authenticate themselves with a valid service catalog account to establish a session.

Access permissions to service catalog entities are governed by the Role-Based Access Control (RBAC) object-level permissions defined for the user in the Service Catalog application. If you have signed into the Service Catalog application using SSO, the SSO tokens are passed to the API automatically.

Apart from supporting calls from external applications, nsAPI can also be invoked from within the Service Portal module. The portal features support the design and rendering of portlets created using Java, JavaScript, or HTML. Within such portlets, nsAPI can be invoked to retrieve the required entity information, and allow users to update the data for certain types of entities. For more information about the portal module, see *Cisco Prime Service Catalog Designer Guide*.

Using REST-based APIs with HTTP clients

A common way of using the REST APIs (also known as nsAPI) that Cisco Prime Service Catalog supports are through HTTP-based clients. In order for a client to call these APIs, it needs to first authenticate the caller. nsAPI supports two methods of authentication, namely header-based authentication and token-based authentication.

Making an nsAPI call with header-based authentication

A RESTful nsAPI call would only be processed if the caller can authenticate itself. This is similar when a user first login to Prime Service Catalog via the browser at:

http://<serverURL>/RequestCenter

then, upon successful login to Service Catalog, user enters a valid nsAPI REST URL in the browser address bar; for example:

```
http://<serverURL>/RequestCenter/nsapi/definition/categories/id/3
```

The response XML, like the one below, is shown in the browser:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<category>
  <categoryId>3</categoryId>
  <categoryName>Workplace Services</categoryName>
  <description>Services for voice and data communications, desktop, mobile devices, and
application access.</description>
  <topDescriptionEnabled>false</topDescriptionEnabled>
  <topDescription />
  <middleDescriptionEnabled>false</middleDescriptionEnabled>
  <middleDescription />
  <bottomDescriptionEnabled>false</bottomDescriptionEnabled>
  <bottomDescription />
  <catalogTypeId>1</catalogTypeId>
  <catalogType>Consumer Services Catalog</catalogType>
  <isRoot>false</isRoot>
  <associatedServices>
    <associatedService>
      <description>Order a new or refurbished laptop. Manager approval
required.</description>
      <id>20</id>
      <name>New Laptop</name>
      <status>Active</status>
    </associatedService>
    <associatedService>
      <description>Order a new iPhone or Blackberry, configured and maintained under
corporate policy.</description>
      <id>22</id>
      <name>New Mobile Device</name>
      <status>Active</status>
    </associatedService>
  </associatedServices>
  <includedCategories>
    <includedCategory>
      <id>8</id>
      <name>Email</name>
    </includedCategory>
    <includedCategory>
      <id>9</id>
      <name>Laptops</name>
    </includedCategory>
  </includedCategories>
  <categoryURLSc>
    <a
href='/RequestCenter/myservices/navigate.do?categoryid=3&amp;query=catalog&amp;layout=popu
p_p' onclick="return GB_showFullScreen('Category', this.href)">Workplace Services</a>
  </categoryURLSc>
  <categoryURLOnlySc>/RequestCenter/myservices/navigate.do?categoryid
=3&amp;query=catalog</categoryURLOnlySc>
</category>
```

If a REST API request was executed before logging into the application, the URL would return the following error:

```
HTTP Error 401 Unauthorized
```

Authenticating a RESTful nsAPI call

The HTTP Header must include the following parameters:

username=<username>

password=<password>

For all nsAPI and RAPI requests, the password is considered as encrypted if the **HTTP Header-acceptEncryptedPassword=true**.

The password is encrypted in the following situations:

- If the value of Accept Encrypted Password is enabled (set to On) in the Administration > Settings page and set to false in the HTTP Header.
- If the value of Accept Encrypted Password is disabled (set to Off) in the Administration > Settings page and set to true in the HTTP Header.

Note You can hide the encrypted value of the password attribute and the cleartext value of the cloudpassword attribute by setting the value for the parameter nsapi.directory.person.hide.secure.information to true in the newscale.properties file.

Install the patch 11.1.1_Patch_v4 or later, on Prime Service Catalog, to use this feature. For more information on installing the patch, refer to the README available with the patch. You can download the patch and the README from the following location on www.cisco.com:
Downloads Home > Products > Cloud and Systems Management > Service Catalog > Prime Service Catalog > Prime Service Catalog 11.1 > Prime Service Catalog Patches.

Upon successful authentication, a JSessionID cookie is returned in the HTTP response. Subsequent invocations of nsAPI should include the same JSessionID cookie in the request to retain the session without having to authenticate again.

During authentication the response code indicates if the user account is locked upon password expiry or unsuccessful password attempts. To unlock the user account, contact the system administrator.

Note The response code for user authentication also indicates if the user account is in its grace period and the date by which the user password needs to be updated.

For more information about password policies, see Enforcing Password Policies in Cisco Prime Service Catalog Administration and Operations Guide.

API session becomes inactive if session times out or if nsapi logout is called by any user. The URL for log out using nsapi is:

```
RequestCenter/nsapi/authentication/logout
```

The following API is used to check if the session is still valid. The client interface returns a HTTP code 200 if session is valid and a HTTP 401 code otherwise.

```
http://<ServerURL>:8088/RequestCenter/nsapi/authentication/session
```

The response XML, like the one below, is shown in the browser:

```
<nsapi-response utid="235c3faa2bbade2ebc85afc2b7f29bd3">User is
authenticated.</nsapi-response>
```

For more information about configuring session time out on the Service Catalog application, see the Site Administration chapter of *Cisco Prime Service Catalog Administration and Operations Guide*.

Note If any nsAPIs are directly called with credentials (without calling nsAPI login) then the session should be automatically terminated after the response is sent. If nsAPI login is explicitly called then the nsAPI session does not terminate automatically unless nsAPI logout is called manually or the session times out based on the new administration setting for the API timeout (Administration Settings > API Timeout).

Note To enforce Cross-site request forgery (CSRF) security vulnerability in nsAPI, set the value of session.token.validation to 1 in newscale.properties file. Furthermore if you provide the authentication token for a session, you need not provide user ID and password for subsequent nsapi calls.

Using Token-based Authentication

Prime Service Catalog supports token-based authentication mechanism. Instead of authenticating with username and password for each request (call), you can authenticate once and obtain a time-bound token in return. After you obtain the token (using GET request), you can use it for subsequent RESTful calls without supplying the username and password.

Obtaining Token ID

Request the token through the following nsAPI GET call. Ensure that the HTTP Header has a valid username and password:

http://<ServerURL/RequestCenter/nsapi/authentication/token?persistent=true

Sample Response:

<sessiontoken utid="P_D887A7375EC640D725A1D042FBFBEAFE"/>

Setting Token Validation Parameter Value

You must set the value of *session.token.validation* parameter in the newscale.properties file as follows:

session.token.validation=1

In addition, set the time-out interval for the token validity in the **API Session Timeout** field under **Administration > Settings** tab.

Setting Token Validation for nsAPI Calls

Note Install the patch 11.1.1_Patch_v2 or later, on Prime Service Catalog, to use this feature. For more information on installing the patch, refer to the README available with the patch. You can download the patch and the README from the following location on www.cisco.com:
Downloads Home > Products > Cloud and Systems Management > Service Catalog > Prime Service Catalog > Prime Service Catalog 11.1 > Prime Service Catalog Patches.

You can enable token based validation for nsapi calls from Service Catalog. This is an optional configuration. The validation can be controlled by setting the value for the parameter *session.token.nsapisc.validation* to 1 in the newscale.properties file.

If you are upgrading and have existing nsAPI calls from external systems that are affected by this token based validation, you may consider setting this property value to 0.

Note You must restart Service Catalog server every time you modify the newscale.properties file.

Indicating Token Expiry

The following error is returned to the nsAPI caller when the token it uses has expired:

<nsapi-error-response errorcode="AUTH_0014"> Token Invalid</nsapi-error-response>

Using the Token

The token ID obtained from REST call must be added as the HTTP Header in subsequent REST-based nsApi calls (username and password would not be required).

utid=<token_id>

Supported Entities

The entities supported by nsAPI come under the following categories:

Table 2-1 Supported Entities Table

Entity Group	Entity Type
Definitional Data	Categories Services Agents Billing Rates Policies
Directory Data	Organizational Units Persons Groups Accounts
Transactional Data	Agreements Requisitions Requisition Entries Authorizations Tasks
Service Item and Standards	Service Item Details All Service Items Standards Billing History Policy Alerts
Portal Designer Data	Custom content tables

Supported Operations

The following types of operations are supported by the nsAPI:

- HTTP GET operations for data retrieval of all entities
- HTTP POST and PUT operations for creating or updating people, accounts, agreements, billing rates, policies and service item instances.
- HTTP POST operations for taking task actions and granting/revoking service item permissions.
- HTTP DELETE operations for deleting accounts, agreements, billing rates, policies and service item instances.

Request and Response Format

All nsAPIs support the use of "application/xml" for request content-type. The following entities also support the application/json content-type:

- Service Items
- Accounts, Agreements
- Billing Rates, Billing History
- Policies and Policy Alerts

The default response content-type will be the same as request content-type but can be overridden with the query parameter 'responseType' (xml or json).

Example:

Submit a new requisition/cart by adding service(s) to it

Method: POST

REST URL:

/RequestCenter/nsapi/transaction/requisitions

Payload:

```
{
  "requisition": {
    "customerLoginName": "admin",
    "billToOU": "H_OU",
    "services":[{
          "name": "TestServiceRest"
          "quantity": "1",
          "version": "0",
          "dictionaries": [{
              "name": "TestNonGrid",
              "data": {
                  "FullName": "AAB",
                  "HireDate": "02/20/1978",
"MultiSelect":["MS3","MS4"],
}
},{
              "name": "TestG",
              "data":
                [
                  {"city":"Bangalore", "country": "India"},
                  {"city":"Mysore", "country":"India"}
                ]
```

```
            }]
        }]
    }
}
```

Success Code: 201

Response Error Code: Refer to REST/Web Services Error Messages table.

Submitting and Managing Service Orders or Requisitions

This section illustrates how a RESTful client places an order with Prime Service Catalog, reading its status for review, and then cancelling it. There are other supported APIs with a service order or requisitions as well as requisition entries, and such APIs are documented in the reference section of this guide.

Using RESTful APIs for service orders

Submitting a service order or requisition

When writing a RESTful client to submit a service order or requisition in Cisco Prime Service Catalog, you need to perform the following steps:

1. Authenticate the client using a valid user account that has the RBAC permission to place an order on the service.

   Issue a command

   http://<ServerURL>/RequestCenter/nsapi/authentication/token?persistent=true

   Supply username and password in the HTTP header

   Assuming the call is successful, you will receive an authentication token:
   <sessiontoken utid="P_D887A7375EC640D725A1D042FBFBEAFE"/>

2. Construct the API payload in JSON format. Typically these are data that you would have filled into the order form if the order was submitted on the UI.

   Assuming you know which Prime Service Catalog service you want to submit a request for, which form field valuess you need to supply, construct the payload for the service request or requisition as follows:

   Payload:

```
{
    "requisition": {
        "services": [
            {
                "version": "0",
                "dictionaries": [
                    {
                        "data": {
                            "Name": "DockerTr9"
                        },
                        "name": "Application_Information"
                    }
                ],
                "name": "Docker1",
                "quantity": "1"
            }
        ],
```

```
        "customerLoginName": "combo1user1",
        "billToOU": "UCSD::uc1::Fenced_Group1"
    }
}
```

Use the "customerLoginName" field to indicate the service is ordered for another user.

Note You can order on behalf of another user based on your permissions defined in the RBAC configuration.

Supply the authentication token (sessiontoken utid=<token_id>) that you receive in the previous step in the HTTP header.

If the order is successfully submitted, Prime Service Catalog returns a Success Code: 201.

The response payload is the following:

```
{
  "RequisitionSubmit": {
     "id": 96,
     "customer": "combo1user1 combo1user1",
     "initiator": "combo1user1 combo1user1",
     "startedDateRaw": 1444267887000,
     "startedDate": "10/07/2015 6:31 PM",
     "status": "Ordered"
  }
}
```

Response Error Code: Refer to REST/Web Services Error Messages table.

3. Since the order has now been placed, the RESTful API client would thus utilize the requisition ID that it received in the previous step to monitor the order status.
   The order may take minutes to days to complete, depending on the service orchestration and delivery process.The nsAPI client would utilize the same utid=<token_id> token received previously, or authenticate to obtain a new token.

   Issue the RESTful call to obtain the order status

   HTTP Method: GET

   http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitionId>

   Alternatively, the response payload you got at the time of placing the order also contains a URL that you can use for checking the requisition status.

In case you want to cancel the order/request placed in custom portal, you can invoke the cancellation in the custom portal.

In this case, the underlying RESTful API client of the portal issues a corresponding request to Prime Service Catalog:

HTTP Method: DELETE

http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/<reqEntryId>?

Payload is not needed

Success Code: 200

Response Error Code: Refer to REST/Web Services Error Messages table.

Legacy SOAP-based RAPI for service orders

This section documents the use of web services for Service Catalog. These include web services which implement the SOAP-based version of Requisition API (RAPI 2), an API which allows an external system to create and manage service requests within Service Catalog. The web services include additional requests, to allow the management of delivery and authorization tasks within a service request; and to review the contents of the Service Catalog.

WSDLs

To validate any request developed, the SOAP-based web services WSDLs must be available. The WSDLs can be found at:

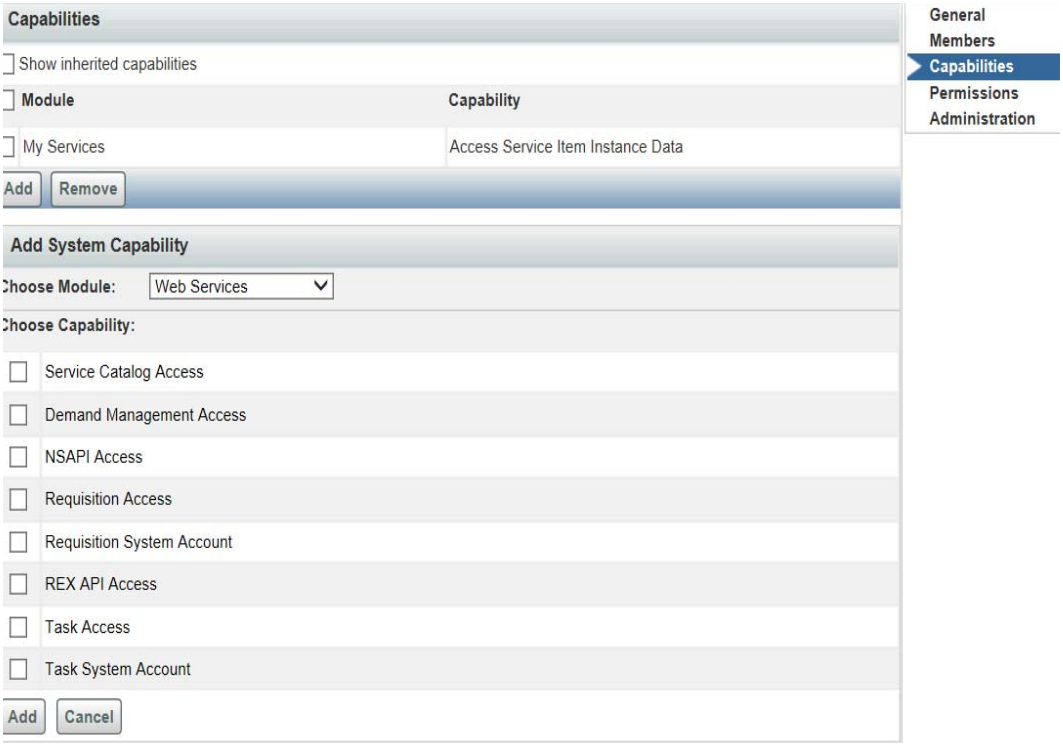
http://<ServerName>/RequestCenter/webservices/wsdl/

Available WSDLs are summarized in the table below.

WSDL	Contents
AuthenticationService.wsdl	A request to authenticate the specified user to Service Catalog.
RequisitionService.wsdl	Requests to submit a requisition, cancel a requisition, or get its status.
ServiceCatalog.wsdl	For internal use only.
ServiceManagerTaskService.wsdl	Requests to approve or reject an authorization or to signify a review has been performed.

Configuring Roles and Capabilities

The web services can be accessed by users who have a role which includes appropriate capabilities for the Web Services module. No prebuilt roles include these capabilities, so administrators will need to use Organization Designer to create one or more custom roles. Once the role is created, you can add Web Services capabilities.



The web services capabilities are:

- **Service Catalog Access:** users having this capability can access the Service catalog for web services.
- **Demand Management Access:** users having this capability can access the Demand Management web service for themselves.
- **NSAPI Access:** users having this capability can access NSAPI web service.
- **Requisition Access**: users having this capability alone can access the RequisitionService web service requests for themselves. The authenticated user and the initiator will have to be the same. If not, an appropriate fault response is thrown.

- **Requisition System Account**: users having this capability can access the RequisitionService web service requests for themselves as well as anybody else. The authenticated user and the initiator can be different.
- **REX API Access**: user having this capability can access the Catalog Deployer Functions.
- **Task Access**: users having this capability alone can access the ServiceManagerTaskService web service requests for themselves. This is a required capability.
- **Task System Account**: users having this capability can access the ServiceManagerTaskService web service requests for themselves as well as anybody else. The authenticated user and the initiator can be different.

Generating WebServices Client Code

The client for the web services can be coded with tools like CXF or Axis from Apache.

Generating Client Code using Axis 2

Detailed instructions and user guide for generating web service client using Axis 2 can be found in the Apache website.

Here are the high-level steps for creating the axis2 client using soapUI:

Step 1 Download the Axis 2 library.

Step 2 Set the Axis 2 library location in the soapUI Preferences menu.

Step 3 Generate the client code by going to **Tools > Axis 2 Artifacts**.

When generating the client code, you should choose **adb**, the Axis default binding, as the databinding method. You should also generate a test case option.

The client code is generated. Method stubs are created in the test case. You will need to populate the objects properly.

Generating Client Code using Apache CXF

Instructions for generating web service client using CXF can also be found in the Apache website.

Here are the steps needed to create a CXF client using soapUI:

Step 1 Download the Apache CXF library.

Step 2 Set the CXF library location in soapUI Preferences menu.

Step 3 Generate the client code by going to **Tools > Apache CXF**.

The client code is generated. The class of interest is:

```
RequisitionServicePortType_RequisitionServiceHttpPort_Client.java
```

This class has a main method and all the operations defined in the WSDL can be invoked from here. The code for invoking these operations will already be present. You must populate the various variables needed. Method stubs are created. All that is needed is to populate the objects properly.

Web Services for Request Management

The operations that can be performed via RAPI 2 request management are summarized in the table below:

Request	Description
addComment	Add a comment to an open requisition. See Adding Comments to a Requisition
cancelRequisition	Cancel an open requisition, including all service requests in the requisition. See Cancelling a Requisition
cancelRequisitionEntry	Cancel a service request. If this is the last service request in the requisition, cancel the requisition. See Cancelling a Requisition
getOpenRequisitions	Get a list of all open requisitions. See Getting a List of Requisitions
getRequisitions	Get a list of open requisitions, optionally restricting the contents of the list. See Getting a List of Requisitions
getRequisitionStatus	Get the status of the specified requisition. See Getting the Requisition Status
getServiceDefinition	Get the definition of the service for which a requisition is to be entered. See getServiceDefinition Response
submitRequisition	Submit a new requisition. See Sample submitRequisition Request

Note RAPI 2 needs the OOB Permission to Submit the request rather the global OOB setting.For information about adding OOB Permission, see Organization Design chapter of Cisco Prime Service Catalog Designer Guide.

Authenticating Web Services

Any web service exposed by Service Catalog needs to be authenticated. Unauthenticated web service calls need to be intercepted and stopped.

Authentication via web services in Service Catalog can be done in the following ways:

- Authenticate per session
- Authenticate per request

Unauthenticated users cannot make any successful web services call. If the global setting “Enable Web Services” is turned off, no web service in Service Catalog is accessible. By default, this setting is turned off.

Authenticate per Session

In this approach the user first makes an Authentication web service call and authenticates the user. The server then establishes a session for this user. As long as this session is valid, this user can make additional web service calls. The authenticate per session request is included in the AuthenticationService WSDL.

The authenticate request has the following format:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:aut="http://authentication.api.newscale.com">
  <soapenv:Header/>
  <soapenv:Body>
   <aut:authenticate>
     <aut:userName>?</aut:userName>
     <aut:password>?</aut:password>
   </aut:authenticate>
  </soapenv:Body>
</soapenv:Envelope>
```

During authentication the error code indicates if the user account is locked upon password expiry or unsuccessful password attempts. To unlock the user account contact the system administrator.

Note The error code for user authentication also indicates if the user account is in its grace period and the date by which the user password needs to be updated. For more information about error codes, see REST/Web Services Error Messages.

Authenticate per Request

In this approach, there is no separate call to the authentication web service. The user sends the authentication information in the SOAP header as part of each web service call. The Authentication handler for the web service in Service Catalog checks whether the user is authenticated. If no session has been established for this particular user, this handler retrieves the authentication information from the SOAP header. If the authentication information is present, this handler tries to authenticate the user. If the authentication information is missing or invalid, this handler throws an exception to the client with appropriate error code and error message.

Encryption

The password specified in the SOAP header may be configured to accept encrypted format only. To enforce encrypted passwords, enable the Accept Encrypted Password setting in the Administration module. An encryption utility is available for users with the Site Administrator role to obtain the encrypted value of a password. To access this utility, open the browser page:

http://<server>:<port>/RequestCenter/EncryptedPassword.jsp

Authenticating mechanism for Web Services

Each web service exposed in Service Catalog has an associated system capability. The authentication handler also checks to see whether the specified user can access (or execute) the web service. If the user has the appropriate system capability, the user is allowed to proceed further. Otherwise, an exception is thrown to the client with the proper error code and message.

Interaction of SOAP Authentication with Directory Integration

If Directory Integration is not enabled, the user specified must exist in the personnel directory before the SOAP request is issued.

If Directory Integration is enabled and the Login event includes an Import Person operation, an external directory is consulted to retrieve the person's profile, and that information is inserted into the personnel directory. In taking this approach, the directory information must include a role granting appropriate web services capabilities, or such a role must have been previously assigned to the business unit (or service teams) of which the user is a member. Consequently, it is recommended that prospective SOAP accounts be prepopulated in the database and assigned appropriate privileges before these accounts submit requests.

If Directory Integration is enabled and the Login event is configured to do only Single Sign-on (SSO), there is an option to bypass the directory events altogether and fall back to simple authentication against the personnel directory. By default, when there is a SOAP request to the web server and SSO is successfull, the SSO user becomes the web service session user. For this to happen, the SOAP header should not contain any user credentials. However, if any overriding credentials are specified in the SOAP request header, the credentials are used to authenticate against the personnel directory instead of the external directory. In other words, the presence of user credentials in the SOAP header controls whether the authentication should be local or external.

If Directory Integration for the Login event includes the External Authentication step (with or without SSO coupled with it), the authentication always is against the Directory datasource.

Getting the Service Definition

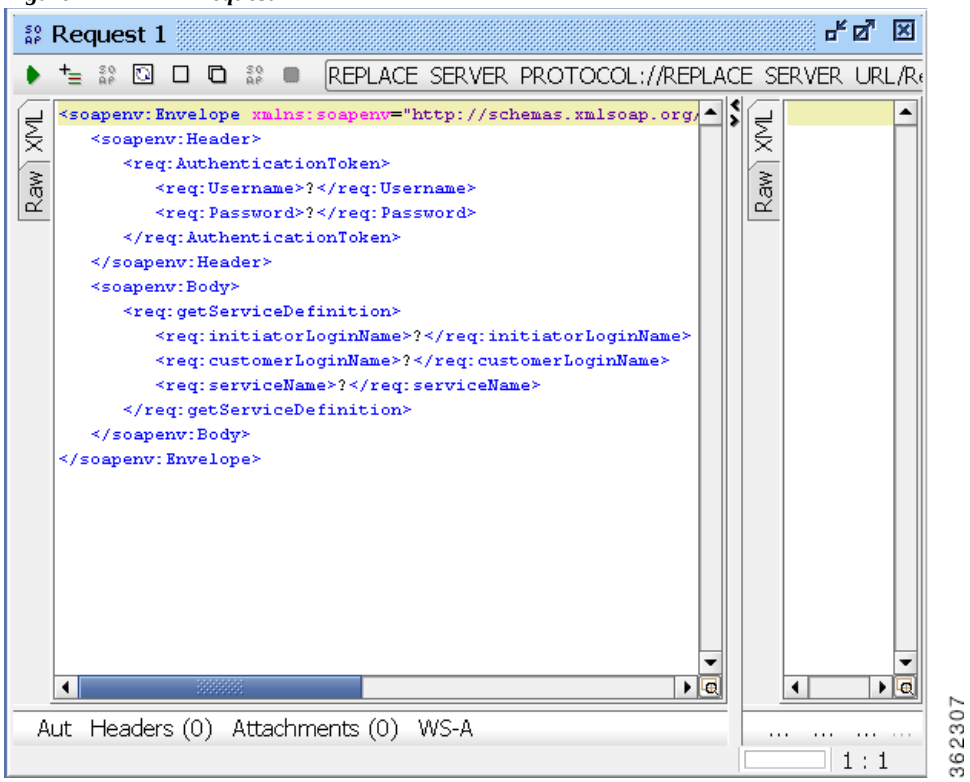
The getServiceDefinition request returns metadata describing the specified service. This metadata is required to submit a request. The use of this operation for services that include grid dictionaries is not supported in this release. An error is returned when the operation is invoked against such services.

getServiceDefinition Request

The request specifies the name of the service whose definition is needed.

In soapUI, right-click the sample request (Request1) under the getServiceDefinition node, then click **Show Request Editor**. The request appears, as it was generated. A question mark (?) indicates all XML elements where a value is expected.

Figure 2-1 Request



You must supply an endpoint for the SOAP request. If you consult the properties of this request (and the menu bar above), you see that the Endpoint has not yet been defined. Replace this with the endpoint for the RAPI 2 services:

http://<ServerName>/RequestCenter/services/**RequisitionService**

where **RequisitionService** is the wsdl name.

You can then copy the request, using the **Creates a copy of this request** icon ( ) in the menu bar of the Request Editor, leaving the prototype request for reference. In your copy, replace the question marks, supplying authentication criteria, as well as the initiator and customer login names and the name of the service you are interested in:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:req="http://requisition.api.newscale.com">
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
     <req:Password>admin</req:Password>
   </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:getServiceDefinition>
     <req:initiatorLoginName>admin</req:initiatorLoginName>
     <req:customerLoginName>mthurston</req:customerLoginName>
     <req:serviceName>New Standard Laptop Computer</req:serviceName>
   </req:getServiceDefinition>
  </soapenv:Body>
</soapenv:Envelope>
```

getServiceDefinition Response

To submit the getServiceDefinition request, click the Submit request to specified URL button ( ) at the top left of the Request Editor window. The response appears within the Request Editor, to the right of the request.

The response to getServiceDefinition returns the metadata that describes the service, as summarized in the table below:

Table 2-2 Response to getServiceDefinition

XML Element (with document hierarchy)	Description
Service	
name	Name of the service
pricingmodel	
quantity	Quantity of service to be ordered
version	The version number of the service
Dictionaries	
Dictionary	Each service contains one or more dictionaries
name	Name of the dictionary
readable	True if the is dictionary readable as per the Access Control in the Service Designer Active form component for the ordering moment; false otherwise
writable	True if the dictionary editable as per the Access Control in the Service Designer Active form component for the ordering moment; false otherwise
Fields >	
DictionaryField	Each dictionary contains one or more fields
canSelectMultiple	Can multiple values be selected for this field?
defaultValue	The default value of the field
fieldDataType	The data type of the field (numeric, date, and so on)
fieldName	The name of the field
inputType	The html input type of the field
label	The label of the field
mandatory	True if the field is mandatory; false otherwise
maxLength	Maximum length of the field
selectableValues	Selectable values for the field

Each dictionary is described, as well as each field within the dictionary. The access control specified for the dictionary in the ordering moment is critical for writing a well-formed submitRequisition request. Only those dictionaries which are readable or writeable by the customer in the ordering moment are included in the response and need to be included in the submitRequisition request.

```
<name>Customer_Information</name>
<readable>true</readable>
<writable>true</writable>
```

```
</Dictionary>
```

The complete getServiceDefinitionResponse for the “New Standard Laptop Computer” is given in the Sample Requests and Responses.

Submitting a Requisition

It is not required to perform a getServiceDefinition request before sending a submitRequisition request. However, the getServiceDefinition returns information that is critical to formulating a valid submitRequisition message for the current version of the service.

- The current version of the service is required. The version number is incremented whenever the service definition itself or any of the included Active Form Components or dictionaries is updated.
- The getServiceDefinition request specifies which fields are mandatory; the submit request must include data for all mandatory fields.
- All mandatory dictionaries and fields must be listed in the submit request. The dictionaries, or the fields within the respective dictionaries, may appear in any order
- Form rules that are configured to be triggered on the browser side do not take effect in web services. If there are select lists or default values that need to be populated by form rules, those rules should be associated with the After Submission event so that they get executed before validations and workflow commence.
- The getServiceDefinition request also returns default values assigned to any fields, included resolved lightweight namespaces for Customer and Initiator information. These values are typically mandatory and need to be supplied in the submitRequisition request.
- The getServiceDefinition request can be used to submit a request for a service whose definition includes fields with options (single-select, multi-select, and radio buttons) when those options are defined using the Active Form Component's Display Options (HTML Representation) pages. When the options are specified via a data retrieval rule, the service request can be submitted; however, it is the responsibility of the submitting program to ensure that the value for the field is a valid option.

The submitRequisition request basically bypasses the ordering moment which occurs when a request is submitted via My Services. No conditional rules, data retrieval rules, or ISF is executed in conjunction with the submitted request. Therefore, if these facilities are used to provide values for dictionary fields or to perform validations, an alternate means must be found of providing these values. The use of this operation for services that include grid dictionaries is not supported in this release. An error is returned when the operation is invoked against such services.

submitRequisition Request

Any dictionary viewable or editable in the ordering moment must be included as a <section> node in the submitrequisition request. All mandatory fields and their values must be specified. No value need be included for the optional fields (but be sure to remove the question marks inserted by soapUI). The order of the dictionaries and the order of the fields does not have to match the order in the service definition but the fields have to appear under the correct dictionary node.

Table 2-3 submitRequisition Request

XML Element (and document hierarchy)	Description
initiatorLoginName	The initiator’s login name
customerLoginName	The customer’s login name
serviceRequests > ServiceRequest	There can be multiple service requests.
name	The name of the service

Table 2-3 submitRequisition Request

quantity	Quantity of services to be ordered
version	The version of the service
Sections > Section	Each service can have multiple dictionaries (sections).
name	The name of the dictionary
Fields > Field	Each dictionary can have multiple fields.
name	The name of the field
value > string	The value to be set for this field

For example, XML setting the value of the ZipCode field in the dictionary RC_ServiceLocation to be "07201" would look like this:

```
<req:Section>
.   .   .
    <req:fields>
.   .   .
        <req:Field>
            <req:name>ZipCode</req:name>
            <req:value>
                <req:string 07201/>
            </req:value>
        </req:Field>
    </req:fields>
    <req:name>RC_ServiceLocation</req:name>
</req:Section>
```

submitRequisition Response

If the request to submit the requisition succeeds, the response with include the requisition ID of the created request, as well as several other attributes of the request.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:submitRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
     <ns1:submitRequisitionResult ns1:customer="admin admin"
         ns1:dueDate="2009-05-08T16:14:26.267-07:00"
         ns1:requisitionId="186"
         ns1:initiator="admin admin"
         ns1:startedDate="2009-04-30T18:14:26.110-07:00"
         ns1:status="Ongoing"/>
   </ns1:submitRequisitionResponse>
  </soap:Body>
</soap:Envelope>
```

The attributes of the submitRequisitionResult response are summarized in the table below:

Table 2-4 submitRequisition Response

XML Element	Description
submitRequisitionResponse > submitRequisitionResult	The response will contain as many entries as there are services in the service request
customer	The customer name for the requisition
dueDate	The due date for the requisition

Table 2-4 submitRequisition Response

requisitionId	The requisition ID for the requisition
initiator	The initiator for the requisition
startedDate	The date the requisition was started
status	The status of the requisition

If the request fails, an error message is returned. Possible errors are shown in Appendix B: RAPI Error Messages. The error message is always in the format of a "SOAP fault", as shown in the sample below:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <soap:Fault>
     <faultcode>soap:Server</faultcode>
     <faultstring>The version specified in the request does not match the version in the
database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</faultstring>
     <detail>
      <RequisitionFault xmlns="http://requisition.api.newscale.com">
        <errorCode>REQ_0018</errorCode>
        <errorMessage>The version specified in the request does not match the version in
the database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</errorMessage>
      </RequisitionFault>
     </detail>
   </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Getting a List of Requisitions

The getRequisitions and getOpenRequisitions operations return information about open requisitions. They differ in the arguments that can be included in the request.

These operations might be useful in managing requisitions. For example, a list of open requisitions might be returned, and those of a particular type (for a particular service) whose past due date exceeds some user-defined threshold may be noted.

getOpenRequisitions Request

getOpenRequisitions returns all open requisitions, up to a specific maximum number of requisitions. The requisitions are returned in descending order by Requisition ID. This request is supported only for backward compatibility of certain retired Service Catalog integration points and should not be used in web services.

getRequisitions Request

getRequisitions returns all requisitions, up to a specific maximum number of requisitions. It also allows you to specify the view type and status of the requisitions to be returned. This request is supported only for backward compatibility of certain retired Service Catalog integration points and should not be used in web services.

Getting the Requisition Status

The getRequisitionStatus operation returns information on the authorizations and task plan status for the specified requisition. The level of detail is similar to that shown to the My Services user, when he/she views the delivery plan:

Figure 2-2 getRequisitionStatus

getRequisitionStatus Request

The request returns information on the current status of a requisition.

Table 2-5 getRequisitionStatus

XML Element (and document hierarchy)	Description
loginUserName	The name of the user requesting the information. This user must have privileges to view the requisition.
requisitionid	The id of the requisition to be interrogated.

GetRequisitionStatus Response

If the request to get the requisition succeeds, the response with include information about the requisition.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
  </soap:Body>
</soap:Envelope>
```

The attributes of the get*RequisitionsResult response are summarized in the table below:

Table 2-6 GetRequisitionStatus Response

XML Element	Description
getRequisitionStatusResponse > get*RequisitionsResult	
RequisitionEntryStatuses > RequisitionEntryStatus	One status block for each service in the request
itemNumber	Sequence assigned to the service within the requisition
quantity	Number of services order
requisitionEntryId	Requisition Entry ID for the service
serviceName	Name of the service
status	Current status of the requisition entry

Table 2-6 GetRequisitionStatus Response

requisitionStepStatuses > RequisitionStepStatus	One StepStatus for each moment configured in the delivery plan for the service
dueDate	Date the current authorization, review or task is due
name	Moment in the delivery plan; for example “Service Group Authorization” or “Delivery project for <service name>”
stepStatus	Status of the task; for example, “In Progress”, “Pending” or “Completed”

Adding Comments to a Requisition

The addComments operation adds a user comment to the specified requisition.

addComments Request

The request adds the specified comment to the specified requisition. The user specified must have permission to access the requisition.

Table 2-7 addComments Request

XML Element (and document hierarchy)	Description
loginUserName	The name of the user adding the comment
requisitionid	The id of the requisition to be affected
commentText	The text of the user comment

Cancelling a Requisition

The cancelRequisition operation is used to cancel the specified service request. All services that comprise the requisition are canceled.

The cancelrequisitionentry operation cancels the specified service (requisition entry) within a service request. If this is the only (or last) service in the requisition, the requisition is canceled. Otherwise, its status remains unchanged.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
     <req:Password>admin</req:Password>
   </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:cancelRequisition>
     <req:loginUserName>ltierstein</req:loginUserName>
     <req:requisitionId>99</req:requisitionId>
   </req:cancelRequisition>
  </soapenv:Body>
</soapenv:Envelope>
The response is shown below (with formatting added for clarity):
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:cancelRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
     <ns1:cancelRequisitionResult
```

```
        ns1:closedDate="2009-06-02T12:04:32.837-07:00"
        ns1:customer="Leslie Tierstein"
        ns1:dueDate="2009-04-03T15:00:00-07:00"
        ns1:id="99"
        ns1:initiator="Leslie Tierstein"
        ns1:startedDate="2009-04-03T09:55:54.843-07:00"
        ns1:status="Cancelled"/>
    </ns1:cancelRequisitionResponse>
  </soap:Body>
</soap:Envelope>
```

Web Services for Task Management

Overview

The operations that can be performed via task management web services are summarized in the table below:

Table 2-8 Web Services for Task Management

Request	Description
approveTask	Approve an authorization/approval.
getAuthorizations	Retrieve authorizations
getAuthorizationsForUser	Retrieve authorizations for a specified user
getMyAuthorizations	Retrieve authorizations assigned to the specified person
rejectSelectedReqEntry	Reject the specified service (requisition entry)
rejectTask	Reject an authorization/approval
reviewTask	Mark a “review” task as reviewed

Getting a List of Authorizations

The getAuthorizations and getMyAuthorizations operations return information about authorizations that are “In Progress”. They differ in the arguments that can be included in the request.

Unlike the Requisition Service operations, which have provisions for separate Web Services Administrative users (specified in the SOAP Header) and the Service Catalog user to which the operation applies, these Task Service operations allow the specification of only one user, in the SOAP header. Therefore, the user whose authorizations are to be retrieved or processed must have the Task Access capability of the Web Services module. To do this:

- Create a role which includes that capability. Since the ability to perform authorizations is included in the My Services Professional role, create a child of that role:
- Assign that role (either in addition to or instead of My Services Professional) to people whose authorizations need to be reviewed or processed via web services:

getMyAuthorizations Request

getMyAuthorizations returns all open requisitions, up to a specific maximum number of requisitions for the person whose Service Catalog credentials are specified in the SOAP header. The requisitions are returned in descending order by Requisition ID. This request is supported only for use in the JSR168-compliant Authorizations portlet and should not be used in web services.

getAuthorizations Request

getAuthorizations returns all authorizations, up to a specific maximum number of authorizations, starting with a specified authorization in the list. It also allows you to specify the view type and status of the requisitions to be returned. This request is supported only for use in the JSR168-compliant Authorizations portlet and should not be used in web services.

getAuthorizationsForUser Request (internal only and unsupported)

This is similar to the getAuthorizations described above, but adds a userLoginName parameter you can use to specify the user for whom you want to get authorizations.

Sample getAuthorizationsForUser SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com">   <soapenv:Header>
      <smt:AuthenticationToken>
         <smt:Username>admin</smt:Username>
         <smt:Password>admin</smt:Password>
      </smt:AuthenticationToken>
   </soapenv:Header>
   <soapenv:Body>
      <smt:getAuthorizationsForUser>
         <smt:userLoginName>qreviewer</smt:userLoginName>
         <smt:startRow>0</smt:startRow>
         <smt:numberOfRows>5</smt:numberOfRows>
         <smt:status>1</smt:status>
         <smt:viewType>2</smt:viewType>
      </smt:getAuthorizationsForUser>
   </soapenv:Body>
</soapenv:Envelope>
```

Useful Parameters for getAuthorizations and getAuthorizationsForUser Requests

Table 2-9 Parameters and Values for getAuthorizations and getAuthorizationsForUser Request

Parameter	Values
Status	Ongoing –1
	Cancelled – 2
	Approved – 3
	Rejected – 4
	Reviewed – 5
	All – 6
ViewType	My Authorizations – 1
	My Assigned and Unassigned – 2

Approving or Rejecting an Authorization

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com">
  <soapenv:Header>
   <smt:AuthenticationToken>
     <!--Optional:-->
     <smt:Username>admin</smt:Username>
     <!--Optional:-->
     <smt:Password>admin</smt:Password>
   </smt:AuthenticationToken>
```

```
  </soapenv:Header>
  <soapenv:Body>
   <smt:approveTask>
     <smt:approverLoginName>maria</smt:approverLoginName>
     <smt:taskID>281</smt:taskID>
   </smt:approveTask>
  </soapenv:Body>
</soapenv:Envelope>
```

The response is shown below (with formatting added for clarity):

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:approveTaskResponse xmlns:ns1="http://smtask.api.newscale.com">
     <ns1:approveTaskResult
       ns1:actionID="5"
       ns1:requisitionId="103"
       ns1:status="approved"
       ns1:taskName="Computer Memory - Upgrade - APPROVAL NEEDED"/>
   </ns1:approveTaskResponse>
  </soap:Body>
</soap:Envelope>
```

Sample Requests and Responses

getServiceDefinition Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:getServiceDefinitionResponse xmlns:ns1=>
     <ns1:getServiceDefinitionResult>
      <dictionaries>
        <Dictionary>
         <fields>
           <DictionaryField>
            <canSelectMultiple>false</canSelectMultiple>
            <defaultValue>
              <string/>
            </defaultValue>
            <fieldDataType>Text</fieldDataType>
            <fieldName>ModelNumber</fieldName>
            <inputType>text</inputType>
            <label>Model Number</label>
            <mandatory>false</mandatory>
            <maxLength>50</maxLength>
            <selectableValues>
              <string/>
            </selectableValues>
           </DictionaryField>
           <DictionaryField>
            <canSelectMultiple>false</canSelectMultiple>
            <defaultValue>
              <string/>
            </defaultValue>
            <fieldDataType>Text</fieldDataType>
            <fieldName>AssetTag</fieldName>
            <inputType>text</inputType>
```

```
    <label>Asset Tag</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
 </fields>
 <name>NewLaptop</name>
 <readable>true</readable>
 <writable>true</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>First_Name</fieldName>
    <inputType>text</inputType>
    <label>First Name</label>
    <mandatory>false</mandatory>
    <maxLength>100</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Last_Name</fieldName>
    <inputType>text</inputType>
    <label>Last Name</label>
    <mandatory>false</mandatory>
    <maxLength>100</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>admin</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Login_ID</fieldName>
    <inputType>hidden</inputType>
    <label>Login ID</label>
    <mandatory>false</mandatory>
    <maxLength>200</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
```

```
 <fieldDataType>Text</fieldDataType>
 <fieldName>Personal_Identification</fieldName>
 <inputType>text</inputType>
 <label>Personal_Identification</label>
 <mandatory>false</mandatory>
 <maxLength>510</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>ed @cisco.com</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Email_Address</fieldName>
 <inputType>text</inputType>
 <label>Email Address</label>
 <mandatory>false</mandatory>
 <maxLength>1024</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>Site Administration</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Home_Organizational_Unit</fieldName>
 <inputType>text</inputType>
 <label>Department</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Company_State</fieldName>
 <inputType>text</inputType>
 <label>State</label>
 <mandatory>false</mandatory>
 <maxLength>100</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Supervisor</fieldName>
 <inputType>hidden</inputType>
 <label>Supervisor</label>
```

```
      <mandatory>false</mandatory>
      <maxLength>100</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Supervisor_Email</fieldName>
      <inputType>hidden</inputType>
      <label>Supervisor Email</label>
      <mandatory>false</mandatory>
      <maxLength>1024</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Custom_1</fieldName>
      <inputType>text</inputType>
      <label>Custom_1</label>
      <mandatory>false</mandatory>
      <maxLength>200</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Custom_2</fieldName>
      <inputType>text</inputType>
      <label>Custom_2</label>
      <mandatory>false</mandatory>
      <maxLength>200</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
     </DictionaryField>
   </fields>
   <name>Customer_Information</name>
   <readable>true</readable>
   <writable>true</writable>
  </Dictionary>
  <Dictionary>
   <fields>
     <DictionaryField>
      <canSelectMultiple>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
```

```
 <fieldName>First_Name</fieldName>
 <inputType>text</inputType>
 <label>First Name</label>
 <mandatory>false</mandatory>
 <maxLength>100</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>admin</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Last_Name</fieldName>
 <inputType>text</inputType>
 <label>Last Name</label>
 <mandatory>false</mandatory>
 <maxLength>100</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>admin</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Login_ID</fieldName>
 <inputType>text</inputType>
 <label>Login ID</label>
 <mandatory>false</mandatory>
 <maxLength>200</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Personal_Identification</fieldName>
 <inputType>hidden</inputType>
 <label>Personal Identification</label>
 <mandatory>false</mandatory>
 <maxLength>510</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string>ed @cisco.com</string>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Email_Address</fieldName>
 <inputType>text</inputType>
 <label>Email Address</label>
 <mandatory>false</mandatory>
```

```
    <maxLength>1024</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>Site Administration</string>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Home_Organizational_Unit</fieldName>
    <inputType>text</inputType>
    <label>Department</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
 </fields>
 <name>Initiator_Information</name>
 <readable>false</readable>
 <writable>false</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string>Yes</string>
    </defaultValue>
    <fieldDataType>Boolean</fieldDataType>
    <fieldName>PerformWork</fieldName>
    <inputType>radio</inputType>
    <label>Will work be performed at the customer location?</label>
    <mandatory>false</mandatory>
    <maxLength>0</maxLength>
    <selectableValues>
      <string>Yes</string>
      <string>No</string>
    </selectableValues>
   </DictionaryField>
 </fields>
 <name>RC_PerformWork</name>
 <readable>true</readable>
 <writable>true</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Street1</fieldName>
    <inputType>text</inputType>
    <label>Street</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
```

```
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Street2</fieldName>
 <inputType>hidden</inputType>
 <label>Street2</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Floor</fieldName>
 <inputType>hidden</inputType>
 <label>Floor</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>OfficeCubeRoom</fieldName>
 <inputType>text</inputType>
 <label>OfficeCubeRoom</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Building</fieldName>
 <inputType>hidden</inputType>
 <label>Building</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
```

```
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>City</fieldName>
    <inputType>text</inputType>
    <label>City</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
  <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>State</fieldName>
    <inputType>text</inputType>
    <label>State</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
  <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>PostalCode</fieldName>
    <inputType>text</inputType>
    <label>Zip Code</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
  <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Country</fieldName>
    <inputType>hidden</inputType>
    <label>Country</label>
    <mandatory>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
  <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>MailStop</fieldName>
```

```
 <inputType>hidden</inputType>
 <label>MailStop</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>Region</fieldName>
 <inputType>hidden</inputType>
 <label>Region</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>District</fieldName>
 <inputType>hidden</inputType>
 <label>District</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>LocationName</fieldName>
 <inputType>hidden</inputType>
 <label>LocationName</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
 <selectableValues>
   <string/>
 </selectableValues>
</DictionaryField>
<DictionaryField>
 <canSelectMultiple>false</canSelectMultiple>
 <defaultValue>
   <string/>
 </defaultValue>
 <fieldDataType>Text</fieldDataType>
 <fieldName>LocationCode</fieldName>
 <inputType>hidden</inputType>
 <label>LocationCode</label>
 <mandatory>false</mandatory>
 <maxLength>50</maxLength>
```

```
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
 </fields>
 <name>RC_RequestorLocation</name>
 <readable>true</readable>
 <writable>true</writable>
</Dictionary>
<Dictionary>
 <fields>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Street</fieldName>
    <inputType>text</inputType>
    <label>Street</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>OfficeCubeRoom</fieldName>
    <inputType>text</inputType>
    <label>OfficeCubeRoom</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>State</fieldName>
    <inputType>text</inputType>
    <label>State</label>
    <mandatory>false</mandatory>
    <maxLength>40</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
   </DictionaryField>
   <DictionaryField>
    <canSelectMultiple>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>City</fieldName>
    <inputType>text</inputType>
```

```
        <label>City</label>
        <mandatory>false</mandatory>
        <maxLength>40</maxLength>
        <selectableValues>
          <string/>
        </selectableValues>
       </DictionaryField>
       <DictionaryField>
        <canSelectMultiple>false</canSelectMultiple>
        <defaultValue>
          <string/>
        </defaultValue>
        <fieldDataType>Text</fieldDataType>
        <fieldName>BuildingName</fieldName>
        <inputType>hidden</inputType>
        <label>BuildingName</label>
        <mandatory>false</mandatory>
        <maxLength>40</maxLength>
        <selectableValues>
          <string/>
        </selectableValues>
       </DictionaryField>
       <DictionaryField>
        <canSelectMultiple>false</canSelectMultiple>
        <defaultValue>
          <string/>
        </defaultValue>
        <fieldDataType>Text</fieldDataType>
        <fieldName>Floor</fieldName>
        <inputType>hidden</inputType>
        <label>Floor</label>
        <mandatory>false</mandatory>
        <maxLength>40</maxLength>
        <selectableValues>
          <string/>
        </selectableValues>
       </DictionaryField>
       <DictionaryField>
        <canSelectMultiple>false</canSelectMultiple>
        <defaultValue>
          <string/>
        </defaultValue>
        <fieldDataType>Text</fieldDataType>
        <fieldName>ZipCode</fieldName>
        <inputType>text</inputType>
        <label>Zip Code</label>
        <mandatory>false</mandatory>
        <maxLength>15</maxLength>
        <selectableValues>
          <string/>
        </selectableValues>
       </DictionaryField>
     </fields>
     <name>RC_ServiceLocation</name>
     <readable>true</readable>
     <writable>true</writable>
    </Dictionary>
  </dictionaries>
  <estimatedpriceperunit>1500.0</estimatedpriceperunit>
  <name>New Standard Laptop Computer</name>
  <pricingmodel>0</pricingmodel>
  <quantity>0</quantity>
  <serviceId>5</serviceId>
  <version>32</version>
```

```
    </ns1:getServiceDefinitionResult>
   </ns1:getServiceDefinitionResponse>
  </soap:Body>
</soap:Envelope>
```

Sample submitRequisition Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
   <req:AuthenticationToken>
     <req:Username>admin</req:Username>
     <req:Password>admin</req:Password>
   </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
   <req:submitRequisition>
     <req:initiatorLoginName>admin</req:initiatorLoginName>
     <req:customerLoginName>admin</req:customerLoginName>
     <req:serviceRequests>
      <req:ServiceRequest>
        <req:name>New Standard Laptop Computer</req:name>
        <req:quantity>1</req:quantity>
        <req:sections>
         <req:Section>
           <req:fields>
            <req:Field>
              <req:name>ModelNumber</req:name>
              <req:value>
               <req:string>T60</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>AssetTag</req:name>
              <req:value>
               <req:string>ABC123</req:string>
              </req:value>
            </req:Field>
           </req:fields>
           <req:name>NewLaptop</req:name>
         </req:Section>
         <req:Section>
           <req:fields>
            <req:Field>
              <req:name>First_Name</req:name>
              <req:value>
               <req:string>admin</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>Last_Name</req:name>
              <req:value>
               <req:string>admin</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>Login_ID</req:name>
              <req:value>
               <req:string>admin</req:string>
              </req:value>
            </req:Field>
            <req:Field>
              <req:name>Personal_Identification</req:name>
              <req:value>
```

```
        <req:string />
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Email_Address</req:name>
       <req:value>
        <req:string>training3@cisco.com</req:string>
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Home_Organizational_Unit</req:name>
       <req:value>
        <req:string>Site Administration</req:string>
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Company_State</req:name>
       <req:value>
        <req:string />
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Supervisor</req:name>
       <req:value>
        <req:string />
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Supervisor_Email</req:name>
       <req:value>
        <req:string />
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Custom_1</req:name>
       <req:value>
        <req:string />
       </req:value>
     </req:Field>
     <req:Field>
       <req:name>Custom_2</req:name>
       <req:value>
        <req:string />
       </req:value>
     </req:Field>
   </req:fields>
   <req:name>Customer_Information</req:name>
 </req:Section>
 <req:Section>
   <req:fields>
    <req:Field>
      <req:name>First_Name</req:name>
      <req:value>
       <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Last_Name</req:name>
      <req:value>
       <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Login_ID</req:name>
```

```
            <req:value>
             <req:string>admin</req:string>
            </req:value>
           </req:Field>
           <req:Field>
            <req:name>Personal_Identification</req:name>
            <req:value>
             <req:string />
            </req:value>
           </req:Field>
           <req:Field>
            <req:name>Email_Address</req:name>
            <req:value>
             <req:string>training3@cisco.com</req:string>
            </req:value>
           </req:Field>
           <req:Field>
            <req:name>Home_Organizational_Unit</req:name>
            <req:value>
             <req:string>Site Administration</req:string>
            </req:value>
           </req:Field>
          </req:fields>
          <req:name>Initiator_Information</req:name>
         </req:Section>
        </req:sections>
        <req:version>32</req:version>
      </req:ServiceRequest>
     </req:serviceRequests>
   </req:submitRequisition>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample getMyAuthorizations Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
   <ns1:getMyAuthorizationsResponse xmlns:ns1="http://smtask.api.newscale.com">
     <ns1:getMyAuthorizationsResult>
      <ns1:Activity>
        <activityFormId xmlns="http://smtask.api.newscale.com">2</activityFormId>
        <activityTypeId xmlns="http://smtask.api.newscale.com">2</activityTypeId>
        <actualDuration xmlns="http://smtask.api.newscale.com">0.0</actualDuration>
        <agentId xmlns="http://smtask.api.newscale.com">0</agentId>
        <clientOrganizationalUnit xmlns="http://smtask.api.newscale.com">
         <authorizationStructure>0</authorizationStructure>
         <billable>true</billable>
         <costCenterCode xsi:nil="true"/>
         <description xsi:nil="true"/>
         <GUID>3C921968-6474-45B2-8D65-A1822E52782F</GUID>
         <id>6</id>
         <localeId>1</localeId>
         <managerId>0</managerId>
         <managerName xsi:nil="true"/>
         <name>Field Sales</name>
         <organizationalUnitTypeId>2</organizationalUnitTypeId>
         <parentId>0</parentId>
         <parentName xsi:nil="true"/>
         <parentOrganizationalUnitGuid xsi:nil="true"/>
         <placeId>0</placeId>
         <placeName xsi:nil="true"/>
```

```
            <recordStateId>1</recordStateId>
            <tenantId>1</tenantId>
          </clientOrganizationalUnit>
          <clientOuId xmlns="http://smtask.api.newscale.com">6</clientOuId>
          <creatorObjectId xmlns="http://smtask.api.newscale.com">57</creatorObjectId>
          <creatorObjectInstId
xmlns="http://smtask.api.newscale.com">9</creatorObjectInstId>
          <customer xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <customerId xmlns="http://smtask.api.newscale.com">12</customerId>
          <customerName xmlns="http://smtask.api.newscale.com">Terry Training</customerName>
          <customerRoleId xmlns="http://smtask.api.newscale.com">0</customerRoleId>
          <customerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <defActivityId xmlns="http://smtask.api.newscale.com">0</defActivityId>
          <depth xmlns="http://smtask.api.newscale.com">0</depth>
          <displayOrder xmlns="http://smtask.api.newscale.com">0</displayOrder>
          <dueOn xmlns="http://smtask.api.newscale.com">2009-06-03T23:00:00-07:00</dueOn>
          <dueOnTz xmlns="http://smtask.api.newscale.com">149</dueOnTz>
          <effort xmlns="http://smtask.api.newscale.com">0.5</effort>
          <escalationLevel xmlns="http://smtask.api.newscale.com">0</escalationLevel>
          <expectedDuration xmlns="http://smtask.api.newscale.com">8.0</expectedDuration>
          <flagId xmlns="http://smtask.api.newscale.com">0</flagId>
          <formURL xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <group xmlns="http://smtask.api.newscale.com">0</group>
          <hasChildren xmlns="http://smtask.api.newscale.com">false</hasChildren>
          <icon xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <id xmlns="http://smtask.api.newscale.com">422</id>
          <instructions xmlns="http://smtask.api.newscale.com"/>
          <isBusy xmlns="http://smtask.api.newscale.com">0</isBusy>
          <isLast xmlns="http://smtask.api.newscale.com">false</isLast>
          <lastChannelId xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <listCount xmlns="http://smtask.api.newscale.com">-1</listCount>
          <nextActionId xmlns="http://smtask.api.newscale.com">5</nextActionId>
          <overbookTime xmlns="http://smtask.api.newscale.com">0</overbookTime>
          <parentId xmlns="http://smtask.api.newscale.com">0</parentId>
          <performerActualDuration
xmlns="http://smtask.api.newscale.com">0.0</performerActualDuration>
          <performerId xmlns="http://smtask.api.newscale.com">11</performerId>
          <performerName xmlns="http://smtask.api.newscale.com">Jared
Roberts</performerName>
          <performerOfficeId xmlns="http://smtask.api.newscale.com">0</performerOfficeId>
          <performerRoleId xmlns="http://smtask.api.newscale.com">331</performerRoleId>
          <performerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <performerSharable
xmlns="http://smtask.api.newscale.com">false</performerSharable>
          <performerShared xmlns="http://smtask.api.newscale.com">false</performerShared>
          <priority xmlns="http://smtask.api.newscale.com">0</priority>
          <priorityName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <processId xmlns="http://smtask.api.newscale.com">0</processId>
          <projectActivityId xmlns="http://smtask.api.newscale.com">0</projectActivityId>
          <reqId xmlns="http://smtask.api.newscale.com">170</reqId>
          <retryCount xmlns="http://smtask.api.newscale.com">0</retryCount>
          <scheduledStart
xmlns="http://smtask.api.newscale.com">2009-06-03T15:00:00-07:00</scheduledStart>
          <startedOn
xmlns="http://smtask.api.newscale.com">2009-06-03T12:09:41.443-07:00</startedOn>
          <startedOnTz xmlns="http://smtask.api.newscale.com">149</startedOnTz>
          <stateId xmlns="http://smtask.api.newscale.com">6</stateId>
          <stateName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <stepId xmlns="http://smtask.api.newscale.com">4</stepId>
          <stepLogicName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <subject xmlns="http://smtask.api.newscale.com">Computer Memory - Upgrade -
APPROVAL NEEDED</subject>
          <taskUrl xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <ticketId xmlns="http://smtask.api.newscale.com">175</ticketId>
```

```
        <ticketObjectId xmlns="http://smtask.api.newscale.com">37</ticketObjectId>
        <totalCost xmlns="http://smtask.api.newscale.com">0.0</totalCost>
        <waiting xmlns="http://smtask.api.newscale.com">0</waiting>
        <WDDXCheckList xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
      </ns1:Activity>
    </ns1:getMyAuthorizationsResult>
  </ns1:getMyAuthorizationsResponse>
 </soap:Body>
</soap:Envelope>
```

REST/Web Services Error Messages

In the error messages below, the symbol of a number enclosed in curly brackets (for example, '{0}') is replaced in the actual error message with the name or identifier of the object that caused the error.

AUTH_0001	The user has not been authenticated yet or the session has timed out.
AUTH_0002	Authentication failed for user '{0}'.
AUTH_0003	Service Catalog is configured for SSO but some configuration problems are preventing the SSO from working correctly.
AUTH_0004	The password must be encrypted properly.
AUTH_0005	The user name header is invalid. It is either not present or empty. Please send a valid header.
AUTH_0006	Access to web services has been turned off.
AUTH_0007	User does not have access to this web service.
AUTH_0008	The SOAP header structure is invalid or the session has timed out.
AUTH_0009	The remote user name is not valid. Please try again with a valid name.
AUTH_0010	User does not have permission to view version information.
AUTH_0011	Your account is locked. Please contact system administrator.
AUTH_0012	Your password expires on <date>. Failure to reset your password before this date will result in suspension of your account. or Your password has expired and your account is suspended.
INFRA_0001	Cannot submit the requisition as the Service Catalog Business Engine queue is not available and the Administration setting for asynchronous submission has been turned on. Please try later or contact your administrator.
REQ_0001	Initiator '{0}' is not found in the system.
REQ_0002	Customer '{0}' is not found in the system.
REQ_0003	Service '{0}' is not found in the system.
REQ_0004	User '{0}' is not found in the system.
REQ_0005	Requisition ID '{0}' is not found in the system.
REQ_0006	Cannot cancel requisition entry '{0}' as the specified requisition id '{1}' does not match.
REQ_0007	Cannot cancel requisition entry '{0}' as the specified service '{1}' does not match.
REQ_0008	User does not have permission to cancel this requisition. Only the requisition owner can cancel a requisition.
REQ_0009	This requisition has already been cancelled.

REQ_0010	The customer '{0}' does not have permission to order the service '{1}'.
REQ_0011	The service '{0}' is not orderable.
REQ_0012	User does not have permission to add comments to this requisition.
REQ_0013	Service form mandatory field: '{0}' has not been filled.
REQ_0014	Service form field: '{0}' exceeds maximum length allowed.
REQ_0015	Please enter a valid number in the field: '{0}'.
REQ_0016	Service form field: '{0}' should have only one value.
REQ_0017	User does not have permission to access this requisition.
REQ_0018	The version specified in the request does not match the version in the database for service '{0}'. Please use the latest service definition.
REQ_0019	The initiator '{0}' does not have Order on Behalf permission for this customer '{1}'.
REQ_0020	The authenticated user '{0}' does not have web service RAPI system account capability.
REQ_0021	The value you passed for this field: '{0}' does not exist in the option list.
REQ_0022	The values for this field: '{0}' have more data than designed values.
REQ_0023	The value you passed for this field '{0}' does not exist in the option list.
REQ_0024	Service form field: '{0}' contains invalid date format.
REQ_0025	Service form field: '{0}' contains invalid date time format
REQ_0026	Service form field: '{0}', you provided the login as '{1}' does not exist in the system. Please input the login name.
REQ_0027	You cannot cancel this requisition ID '{0}'. The requisition is closed.
REQ_0028	The Requisition ID '{0}' is past the point of no return. You cannot cancel this requisition.
REQ_0029	You cannot cancel this requisition entry Id '{0}'. The requisition entry is closed.
REQ_0030	The status of service '{0}' is inactive.
REQ_0031	You cannot cancel this requisition Id '{0}'. The requisition entry is already cancelled.
REQ_0032	The value you passed for this field '{0}' does not exist in the option list.
REQ_0033	The value you passed for this field '{0}' does not exist in the option list.
REQ_0034	The values for this Field '{0}' have more data than designed values.
REQ_0035	The dictionary name '{0}' does not exist for this service. Please correct the field name.
REQ_0036	The dictionary field '{0}' does not exist for this dictionary '{0}'. Please correct the field name.
REQ_0037	User does not have permission to cancel this requisition entry.
REQ_0038	Number of fields does not match for this dictionary: {0}’.
REQ_0039	Number of dictionaries does not match for this service:'{0}'.
REQ_0040	The user: '{0}' does not exist in DB the service catalog database. Tried importing user from LDAP, but LDAP lookup is set to false.

REQ_0041	OOB event is not configured/enabled. LDAP search for this user:'{0}' was not performed
REQ_0042	Problem while importing data from LDAP.
REQ_0043	The user: '{0}' does not exist in the service catalog database and the LDAP System
REQ_0044	Service form person look up event is not configured or enabled. LDAP search for this user:'{0}' was not performed.
REQ_0045	The event operations are not properly configured to import the person from LDAP
REQ_0046	Please enter only monetary values without currency symbols in this field: '{0. Use "." as the decimal separator.
REQ_0047	Please enter only letters and numbers in this field: '{0}'.
REQ_0048	Please enter only letters and numbers in this field: '{0}'.
REQ_0049	The dictionary: '{0}' is repeated more than once in the request. A given dictionary can appear only once in a request.
REQ_0050	The dictionary field: '{0}.{1}' is repeated more than once in the request. A given dictionary field can appear only once in the request.
REQ_0051	The service '{0}' contains one or more grid dictionaries. This API does not accept submission of requisition with grid dictionary values
REQ_0052	The service form could not be submitted because of following error: {0}
REQ_0053	The service form could not be submitted because of the following error: missing mandatory field {0}
REQ_0054	The requisition could not be submitted as it contains one or more services that can only be submitted individually based on the ordering mode
REQ_0055	Last Date the process was executed
REQ_0056	You have reached the maximum '{1}' records allowed for dictionary '{0}'.
REQ_0057	Requisition ID: '{0}' already submitted.
REQ_0058	Unsubmitted requisition not found.
REQ_0059	Ordering mode option is disabled for the service: {0}
REQ_0060	The requisition cannot be cancelled or deleted.
REQ_0061	Multiple unsubmitted requisitions found.
REQ_0062	Requisition {0} entry not found
REQ_0063	Requisition {0} entry cannot be cancelled/ or deleted
REQ_0064	Access denied for the requisition operation
REQ_0065	No requisition entries found for the requisition.
REQ_0100	Runtime Exception occurred. This can be caused by a legitimate Business Engine workflow exception (for example, the task is not allowed to be cancelled as defined in Service Designer). User should check the task definition.
REQ_0101	Runtime error occurred while reading the service form data.
REQ_0101	Runtime error occurred while reading the service form data.
TASK_0005	The task '{0}' cannot be rejected.
TASK_0008	The task '{0}' doesn't exist in the system.

TASK_0001	The user '{0}' doesn't have permission to approve the task '{1}'.
TASK_0002	The user '{0}' doesn't have permission to reject the task '{1}'.
TASK_0003	The user '{0}' doesn't have permission to review the task '{1}'.
TASK_0004	The task '{0}' is not an approval task.
TASK_0005	The task '{0}' cannot be rejected.
TASK_0006	The task '{0}' is not a review task.
TASK_0008	The task '{0}' does not exist in the system.
TASK_0009	The requisition entry id '{0}' for the specified task id '{1}' does not match.
TASK_0010	The task '{0}' does not contain more than one requisition entries.
TASK_0011	The task '{0}' does not have financial or OU authorization.
TASK_0012	The user'{0}' does not have permission to reject partial requisition entry for this task '{1}'.
TASK_0013	The task '{0}' has already been rejected.