Cisco Prime Service Catalog 12.1 Integration Guide

31 August, 2017

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco Prime Service Catalog 12.1 Integration Guide

Getting Started

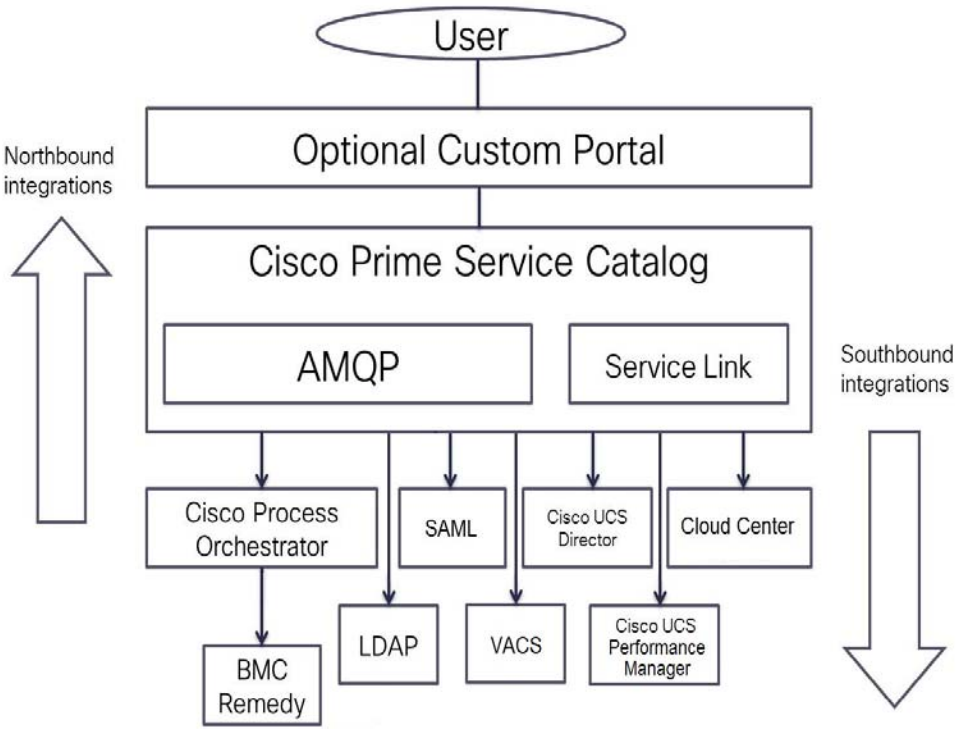
Cisco Prime Service Catalog provides employees with a self-service portal, a service catalog, and a lifecycle management software for enterprise IT use cases (licenses sold per user) and cloud computing use cases (licenses sold per server). These services are designed in Cisco Prime Service Catalog by service designers and made available in the Service Catalog portal for end user to order.

The services may be a simple service like ordering a laptop where the IT support provides the laptop to the employee or a complex service such as provisioning an application stack that requires automation and orchestration of various tasks.

To be able to perform services that require automation or orchestration Prime Service Catalog integrates with external systems. The external systems could be external directories, domain managers, BMC Remedy SRM, Cisco UCS Director, and so on.

Any application that is making a call to request services or manages service items is at the northbound of the integration design and the application that receives the request, performs the required action, and sends a response is at the southbound of the integration design architecture. For example a company's custom portal that makes an API call to Service Catalog for service request is at the northbound of Prime Service Catalog whereas the other applications that receives an API call from Prime Service Catalog to perform the required operation is at the southbound of Prime Service Catalog.

Figure 1-1Integration Scenario Example



During a service request/ delivery workflow Prime Service Catalog integrates either with Cisco Process Orchestrator which in turn interacts with other applications to be able to complete a service request. It could also directly interact with the external application based on the integration design. ie., either the Prime Service Catalog interacts with an external application by making a southbound call directly to the application or through Cisco Process Orchestrator.

There are also a few implementations where the Service Catalog is integrated with the company's portal. This means that the services are defined by the company in its portal and is designed and delivered using Prime Service Catalog. Therefore the end user submits service request using the company's portal and the portal interacts with Prime Service Catalog for the service delivery.

Consider the following scenario for Prime Service Catalog Integration:

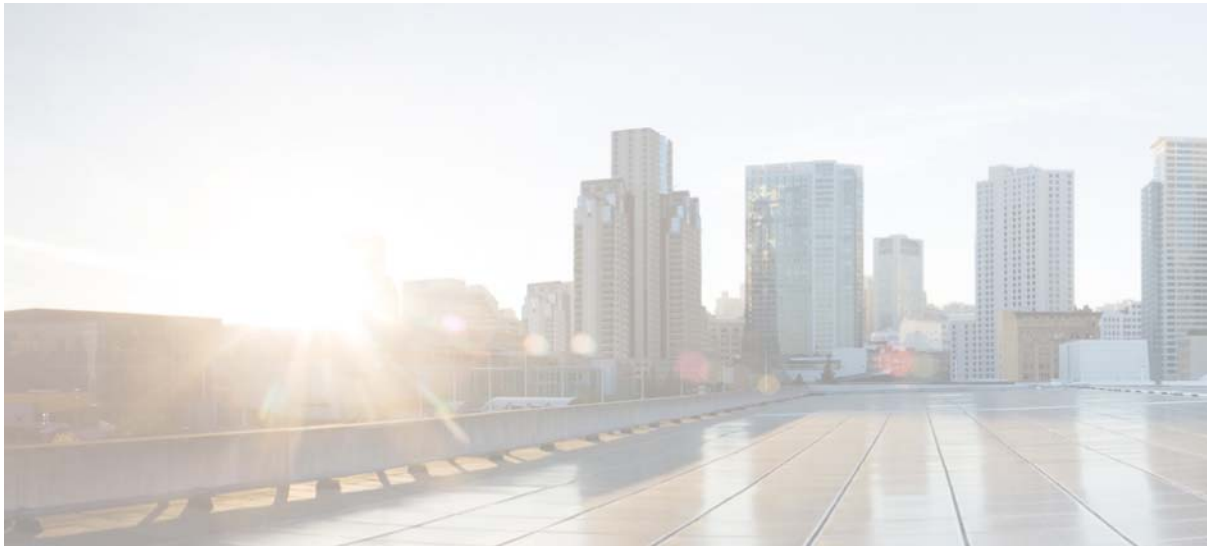
Scenario: A user needs to add himself to a organization group.

Workflow:

1. Service Designer designs a service where the service form has the following details:

   User ID, Group name(drop-down list), User Name, and the Request Name “Join Group”.
2. User logs in to Service Catalog and requests the service” Join group”
3. User fill the details in the service form and clicks submit
4. Prime Service Catalog gets the user details and service form attributes from the service form and sends it to service link.
5. Service link makes an API call to active directory add the user to the group.
6. Active directory completes the request and returns HTTP response 200 and the flow is complete.

Note Service Link module of Prime Service Catalog enables the integrations of Prime Service Catalog with external systems.



PART 1

Northbound Integration

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*.

New, Changed, and Deprecated APIs

This table provides information about the new and changed API information for this release:

Table 2-1 New and changed API Information Table

API Description	REST URL	Status	Location
Grants/revokes permissions to People, OU, or Role	http://<ServerURL>/RequestCenter/nsapi/directory/entity/permissions	New	Table 3-17 Permissions API Table, page 3-152
Get all ObjectOperationIds and it's details for given ObjectId	http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/id/{entitytype}/permissions	New	Table 3-17 Permissions API Table, page 3-152

Table 2-1 New and changed API Information Table

API Description	REST URL	Status	Location
Grants/revokes permission to Service Group, Active Form Group, or DictionaryGroup for the specified Role ID	http://<ServerURL>/RequestCenter/nsapi/directory/roles/id/{roleID}/permissions/	Enhancement	Table 3-17 Permissions API Table, page 3-152
Gets list of entities	http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids	New	Table 3-17 Permissions API Table, page 3-152
Gets all the modules	http://<ServerURL>/RequestCenter/nsapi/directory/module/ids	New	Table 3-17 Permissions API Table, page 3-152
Gets assignment IDs for the specified OU	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/1/permissions	New	Table 3-15 Organizational Unit API Table, page 3-134
Gets the assignment IDs for the specified user	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/5/permissions	New	Table 3-14 Person API Table, page 3-119
Filters the permissions for specified object type for person	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/5/permissions?objectId=310	New	Table 3-14 Person API Table, page 3-119
Filters the permissions for specified object type for OU	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/5/permissions?objectId=13	New	Table 3-15 Organizational Unit API Table, page 3-134
Gets the assignable instances for the given object ID of the person only when ServiceGroup has assign right	http://<ServerURL>/RequestCenter/nsapi/directory/entity/1/assignableinstances	New	Table 3-17 Permissions API Table, page 3-152
Gets available instances and its details for the given object ID of the person.	http://<ServerURL>/RequestCenter/nsapi/directory/entity/31/availableinstances	New	Table 3-17 Permissions API Table, page 3-152
Gets the list of assignment types for the assignment	http://<ServerURL>/RequestCenter/nsapi/directory/entity/assignmenttypes	New	Table 3-17 Permissions API Table, page 3-152
Assigns/unassigns roles to multiple people/OU/Group	http://<ServerURL>/RequestCenter/nsapi/directory/entity/roles	New	Table 3-17 Permissions API Table, page 3-152
Creates the Custom SG and SIG for the given name in the payload	http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions	New	Table 3-7 Services API Table, page 3-75
Updates the SG name and SIG Name	http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions	New	Table 3-7 Services API Table, page 3-75
Gets the assignable roles	http://<ServerURL>/RequestCenter/nsapi/directory/people/assignableroles?type=1	New	Table 3-14 Person API Table, page 3-119
Gets the solution type of logged in user	http://<ServerURL>/RequestCenter/nsapi/directory/people/solutiontypes	New	Table 3-14 Person API Table, page 3-119

Table 2-1 New and changed API Information Table

API Description	REST URL	Status	Location
Gets the operation action IDs	http://<ServerURL>/RequestCenter/nsapi/directory/entity/operationactionids	New	Table 3-17 Permissions API Table, page 3-152
Gets the info whether the Role for the given name exists or not	http://<ServerURL>/RequestCenter/nsapi/directory/role/name/{name}/roleexists	New	Table 3-17 Permissions API Table, page 3-152
Gets the integration types	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getintegrationtypes	New	Table 3-6 Environment API Table, page 3-64
Gets the type of getCloudIntegrationsList	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getCloudIntegrationsList?type=2	Enhancement	Table 3-6 Environment API Table, page 3-64
Deletes the custom service group and Service Item Group (if SIG associated to SG)	http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions?id=23	New	Table 3-7 Services API Table, page 3-75
Gets the user list of the custom group's SOA role	http://<ServerURL>/RequestCenter/nsapi/ucsd/v1/soauserslist/custom/26	New	Table 3-6 Environment API Table, page 3-64
Transfers or shares the SOA role to specified users	http://<ServerURL>/RequestCenter/nsapi/directory/v1/transfersoa	Enhancement	Table 3-6 Environment API Table, page 3-64
Gets all the Roles	http://<ServerURL>/RequestCenter/nsapi/directory/roles	New	Table 3-17 Permissions API Table, page 3-152
Get all the capabilities for a module	http://<ServerURL>/RequestCenter/nsapi/directory/modules/id/7/capabilities	New	Table 3-17 Permissions API Table, page 3-152
Get objects on which permissions can be applied	http://<ServerURL>/RequestCenter/nsapi/directory/entity/objectId/{objectId}/operationId/{operationId}/objectinstances	New	Table 3-17 Permissions API Table, page 3-152
Creates a new role	http://<ServerURL>/RequestCenter/nsapi/directory/v2/role	New	Table 3-14 Person API Table, page 3-119
Gets all Sub Roles of the specified role	http://<ServerURL>/RequestCenter/nsapi/directory/subroles/id/{roleId}	New	Table 3-14 Person API Table, page 3-119
Gets Capabilities of a specified role	http://<ServerURL>/RequestCenter/nsapi/directory/roles/id/{roleId}/capabilities?showInherited=true	New	Table 3-14 Person API Table, page 3-119
Gets assigned members of role	http://<ServerURL>/RequestCenter/nsapi/directory/v2/roles/id/{roleId}/members	New	Table 3-14 Person API Table, page 3-119
Edits a role	http://<ServerURL>/RequestCenter/nsapi/directory/v2/role	New	Table 3-14 Person API Table, page 3-119
Clones a role	http://<ServerURL>/RequestCenter/nsapi/directory/v2/role?cloneof={roleId}	New	Table 3-17 Permissions API Table, page 3-152
Deletes the specified role	http://<ServerURL>/RequestCenter/nsapi/directory/v2/role?roleId={roleId}	New	Table 3-17 Permissions API Table, page 3-152
Gets information of all the services belonging to the specified connection	http://<ServerURL>/RequestCenter/nsapi/definition/connection/{connectionid}/servicelist	New	Table 3-7 Services API Table, page 3-75

Table 2-1 New and changed API Information Table

API Description	REST URL	Status	Location
Fetches information of all services which belongs to service group	http://<ServerURL>/RequestCenter/nsapi/definition/servicegroup/{servicegroupid}/servicelist	New	Table 3-7 Services API Table, page 3-75
Fetches all the information of the project tenants	http://<ServerURL>/RequestCenter/nsapi/directory/v1/projecttenant	New	Table 3-34 Tenant Management API Table, page 3-225
Assigns services to tenants	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceassignment?action=”grant”	New	Table 3-7 Services API Table, page 3-75
Revokes services to tenants	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceassignment?action=”revoke”	New	Table 3-7 Services API Table, page 3-75
Gets assigned services for person with the filters for name, description, connection service group	http://<ServerURL>RequestCenter/nsapi/directory/v2/tenant/{tenantID}/assignedservices?	New	Table 3-34 Tenant Management API Table, page 3-225
Makes the service orderable for the tenant users	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceorder?action="grant"	New	Table 3-7 Services API Table, page 3-75
Makes services unorderable for tenant users	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceorder?action="revoke"	New	Table 3-7 Services API Table, page 3-75
Submits requisition of a transaction	http://<ServerURL>/RequestCenter/nsapi/transaction/v1/requisition	New	Table 3-20 Requisitions API Table, page 3-170
Gets service items for All or specific connection	http://<ServerURL>/RequestCenter/nsapi/serviceitem/SiSubmitSI	New	Table 3-28 All Service Items API Table, page 3-204
Gets list of connections	http://<ServerURL>/RequestCenter/nsapi/definition/v1/services/connection	New	Table 3-7 Services API Table, page 3-75
Gets specific assets of the particular connection	http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/shortName/{ConnectioinShortName}/assets/{assetType}	New	Table 3-7 Services API Table, page 3-75
Gets list of all the assets for the specified connection	http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/shortName/{ConnectioinShortName}	New	Table 3-7 Services API Table, page 3-75
Gets all the service assets for all the connections	http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets	New	Table 3-7 Services API Table, page 3-75
Gets list of specific asset for all connections	http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/assets/{AssetType}	New	Table 3-7 Services API Table, page 3-75
Gets all the SOA users list for the specified connection	http://<ServerURL>/RequestCenter/nsapi/ucsd/v1/soauserslist/shortname/{ConnectioinShortName}	New	Table 3-14 Person API Table, page 3-119
Deletes SOA Role of the connection	http://<ServerURL>/RequestCenter/nsapi/directory/v1/soarole/roleid/{roleID}/personId/{personID}	New	Table 3-14 Person API Table, page 3-119

Table 2-1 New and changed API Information Table

API Description	REST URL	Status	Location
Gets all the tenants matching the filter	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant?	New	Table 3-34 Tenant Management API Table, page 3-225
Gets all the tenant admins of the specified tenant ID	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{TenantID}/admins?	New	Table 3-34 Tenant Management API Table, page 3-225
Creates sub tenants/sub teams by the tenant admin	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant	New	Table 3-34 Tenant Management API Table, page 3-225
Adds users to the team	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}/users?action=add	New	Table 3-34 Tenant Management API Table, page 3-225
Removes users from the team	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}/users?action=remove	New	Table 3-34 Tenant Management API Table, page 3-225
Activates and deactivate team	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}?activate={true or false}	New	Table 3-34 Tenant Management API Table, page 3-225
Gets all the requisitions ordered by the User for all teams the user belongs to	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/RequisitionViewName=[MyTeams]	New	Table 3-34 Tenant Management API Table, page 3-225
Gets all the requisitions for the tenant if the logged in user belongs to the tenant, or has admin privileges on the tenant	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/RequisitionViewName=[MyTeams]?teamId={teamId}}	New	Table 3-34 Tenant Management API Table, page 3-225

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.

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.

Enabling HSTS on IIS and Apache Server

HTTP Strict Transport Security (HSTS) is a security enhancement specified by a web application through the use of a special response header. HSTS allows the web servers to declare that web browsers must only communicate using the secure HTTPS, and never HTTP protocol.

Note: You must restart the webservers after adding the configurations for the changes to take effect.

To Configure on HSTS on Apache Server

Add the below configuration in the httpd.conf file located at /etc/httpd/conf/httpd.conf.

```
<VirtualHost {your Ip Address}:443>
    Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains;"
</VirtualHost>
<VirtualHost *:80>
  [...]
  <IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
   </IfModule>
</VirtualHost>
```

To Configure on HSTS on IIS Server

Add the below configuration in the web.config file located at C:\inetpub\wwwroot\web.config.

```
<httpProtocol>
            <customHeaders>
                        <add name="Strict-Transport-Security" value="max-age=15552001;
includeSubDomains; preload" />
            </customHeaders>
        </httpProtocol>
        <rewrite>
            <rules>
                <rule name="http to https" stopProcessing="true">
                    <match url="(.*)" />
                    <conditions>
                        <add input="{HTTPS}" pattern="^OFF$" />
                    </conditions>
                                <action type="Redirect" url="https://{HTTP_HOST}/{R:1}"
redirectType="SeeOther" />
                </rule>
            </rules>
        </rewrite
```

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

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 2-2 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 values 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:

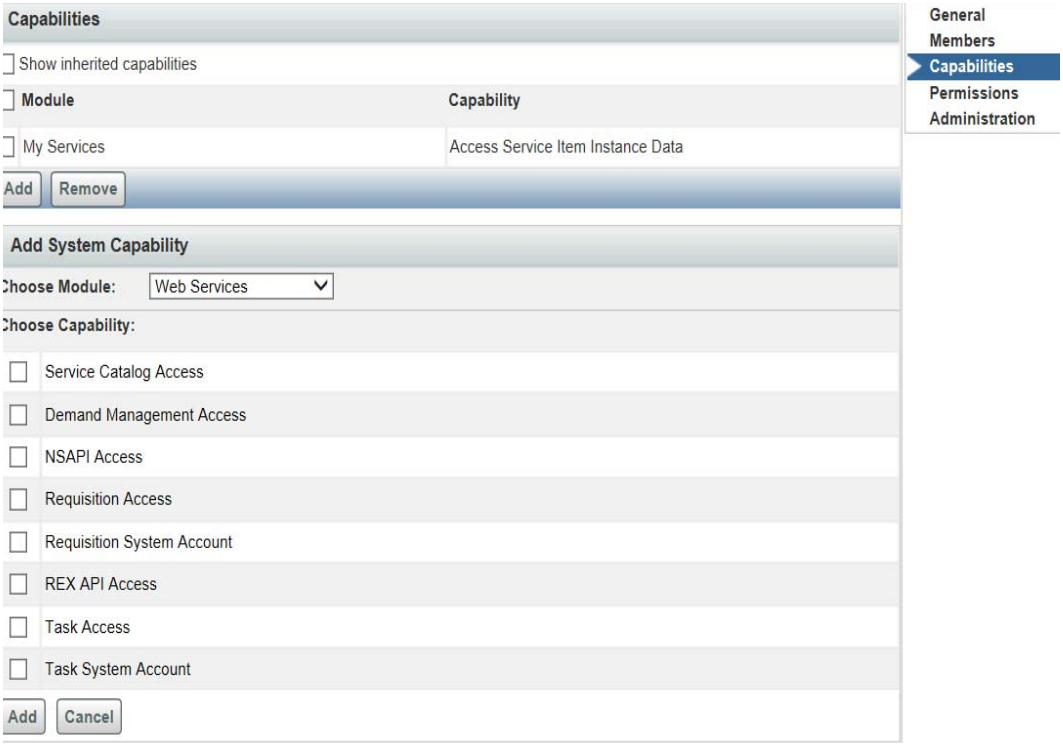
http://<ServerName>/RequestCenter/webservices/wsdl/

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

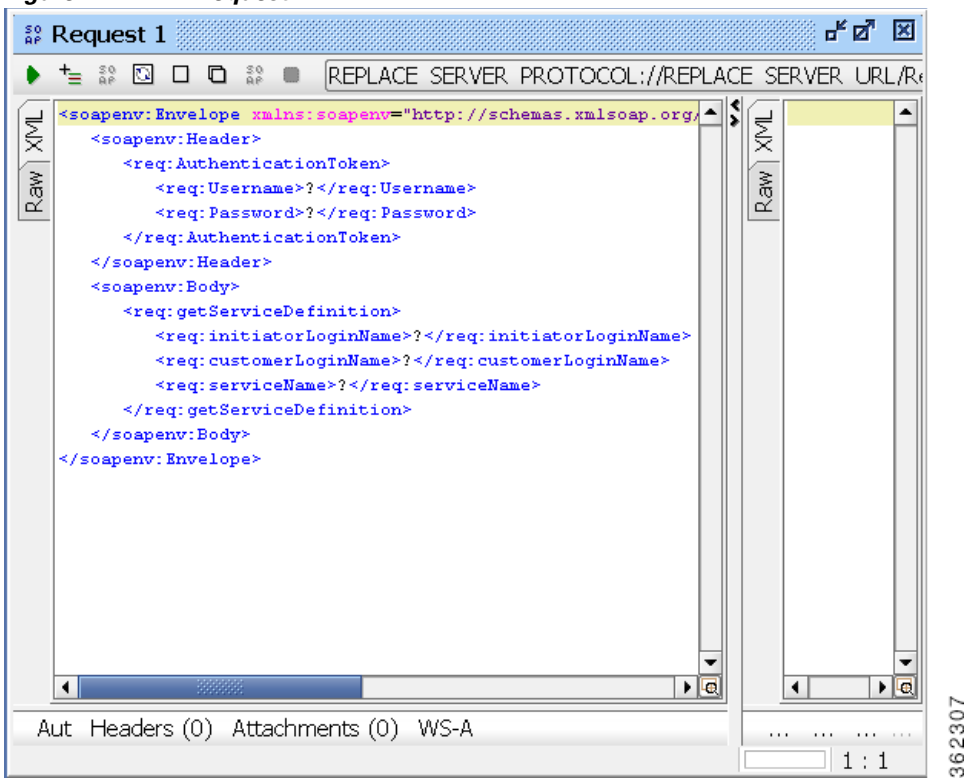
The getServiceDefinition request returns metadata describing the specified service. This metadata is required to submit a request. The use of this operation for services that include grid dictionaries is not supported in this release. An error is returned when the operation is invoked against such services.

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 2-3 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 2-4 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
quantity	Quantity of services to be ordered
version	The version of the service

Table 2-4 submitRequisition Request

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 2-5 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
requisitionId	The requisition ID for the requisition
initiator	The initiator for the requisition

Table 2-5 submitRequisition Response

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 2-6 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 2-7 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
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 2-8 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 2-9 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 2-10 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.

References and Examples for Prime Service Catalog RESTful APIs

Overview

This chapter 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.

REST URL Syntax and Convention

- The REST URL follows this convention:

  ```
  http(s)://<serverURL>/RequestCenter/nsapi/<entityGroup>/<entityType>/<filters>?sortBy=
  <columnName>&sortDir=<sortOrder>?startRow=<x>&recordSize=<y>
  ```

  where elements enclosed in angle brackets (<>) indicate that an appropriate parameter or parameter value must be substituted.
- Filters, sorting and paging controls and actions are passed as optional parameters in the REST URLs. Filters may include one or multiple expressions as explained in the "Supported Filters" section. When filters are absent, all instances found for the entity are returned.
- Multiple filters can sometimes be combined. In this case, second and subsequent instances of the parameter are optional, and the syntax diagram indicates this by enclosing the parameter syntax within square brackets ([]). The optional parameters must be enclosed in separators (|).
- Unless otherwise noted, all URLs are case-sensitive.
- Versioning is built into the nsAPI. For the base version 1.0, the REST URLs do not have the version number as a parameter. However, this may change in the future releases.
- HTTP status codes and error messages are returned in the REST responses to show whether the requested operation succeeded or failed.
- nsAPI Java binding packages are named com.newscale.nsapi.*.

Supported Filters

nsAPI supports a variety of filters for GET operations based on the entity type:

Table 3-1 Supported Filters Table

Entity Type	Available Filters/Syntax	REST URL Examples
All Entities	Id /id/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/id/16
	Name – exact match /name/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/name/Create%20Custom%20VM
	Name – wildcard search ?name=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?name=Create%20Custom*
Standards, Service Items, Custom Content	Any table column, with up to three filter expressions comprised of the following elements: • Comparison Operators: =, >, <, >=, <= • Relational Operators: AND, OR (case-insensitive, order precedence is not supported) • Separator: | /<columnName1><operator1><value1>[|<AND|OR>|]<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] *Date field values should be in mm-dd-yyyy format.*	http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|AND|Custom2=64 *which can also be written as:* http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|Custom2=64 http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux|OR|Custom2=64 http://serverURL/RequestCenter/nsapi/serviceitems/serviceitemsubscription/SubmitDate>=03-01-2011 http://serverURL/RequestCenter/nsapi/serviceitems/serviceitemsubscription/AccountName=tenantaccount1 http://serverURL/RequestCenter/nsapi/customcontent/UcAnnouncementObj/Category=Corporate
Service Items	View Name ?ViewName=<value> *where possible View Names are:* • *My ServiceItems (as seen in My Services)* • *Manage ServiceItems (as seen in Service Item Manager)*	http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=My%20ServiceItems http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=Manage%20ServiceItems
Organizational Units	OU Type ?type=<all/businessUnit/serviceTeam>	http://serverURL/RequestCenter/nsapi/directory/organizationalunits?type=serviceTeam

Table 3-1 Supported Filters Table

Entity Type	Available Filters/Syntax	REST URL Examples
Person	Login Name ?loginname/<value>	http://serverURL/RequestCenter/nsapi/directory/people?loginname/dsmith
	OU Name ?ouname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?ouname=Operations
	Group Name ?groupname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?groupname=Approvers
	Role Name ?rolename=<value> *Inherited roles are not available for filtering.*	http://serverURL/RequestCenter/nsapi/directory/people?rolename=Service%20Performer
Categories	Category Type ?catalogType=<serviceCatalog/offeringCatalog>	http://serverURL/RequestCenter/nsapi/definition/categories?catalogType=serviceCatalog
	User Context ?userContext=<true/false>	http://serverURL/RequestCenter/nsapi/definition/categories?userContext=true
Services	Category Name ?categoryName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?categoryName=Manage%20Physical%20Servers
	Keyword ?keywordName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?keywordName=server
Requisitions and Authorizations	View Name, Status /ViewName=<value1>[| Status=<value2>] *The view names available for filtering correspond to the list of views in the My Services Requisitions and Authorizations tabs.* *The status filter must be used in conjunction with the view name filter and cannot be used on its own.*	http://serverURL/RequestCenter/nsapi/transaction/requisitions/ViewName=Ordered%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self| Status=Approved

Table 3-1 Supported Filters Table

Entity Type	Available Filters/Syntax	REST URL Examples
Tasks	View Name ?viewName=<value> *The view names available for filtering correspond to the list of system-defined views in Service Manager. User-defined views are not available for filtering.*	http://serverURL/RequestCenter/nsapi/transaction/tasks?viewName=AvailableWork
Tasks – for a specific requisition entry	Task Type ?taskType=<value> where possible Task Types are: all, delivery, authorization Task Status (Skipped) ?showSkippedTasks=<false|true>	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=1234?taskType=delivery&showSkippedTasks=true

For POST operations, the following filters are supported:

Table 3-2 Supported Filters Table for POST Operations

Entity Type	Available Filters	REST URL Examples
Person	Login Name or Id *The filter is implicit in the operation and the person identifier is obtained from the request XML.*	http://serverURL/RequestCenter/nsapi/directory/people/update *The same URL is used for both create and update actions. No person attribute needs to be passed in the URL.*
Tasks	Id /<value>/<done |approve |reject | review> *The "|" indicates that exactly one of the possible options must be chosen.*	http://serverURL/RequestCenter/nsapi/transaction/tasks/215/approve

Wildcard Search Entries

For wildcard search in entity name filters (for example, ?name=<value>), leading wildcard characters such as "*" or "%" in the search string are ignored. If these characters are located in the middle or at the end of the string, they are applied in wildcard matching.

To enable the use of a leading wildcard character in name search, locate the property ContainsQueryInFnS
in the newscale.properties file (under RequestCenter.war/WEB-INF/classes/config) and set the value to true:

```
ContainsQueryInFnS=true
```

This property also controls full wildcard search support in the Service Manager and Service Link modules.

Note Such search operations may negatively impact system performance and are generally not recommended in production environments.

Associated entities are nested entities that are fetched in conjunction with the primary entity —for example, the OU of which a person is a member. Associated entity name filters do not support wildcard search. Searches on those filters return only results with an exact match, and are case-sensitive. In other words, wildcard characters in the search string are treated as literals during matching. For example,

```
/nsapi/directory/people?ouname=star*OU
```

returns people in the organization unit literally named “star*OU”.

```
/nsapi/directory/people?groupname=starGroup*
```

returns people in the group literally named “starGroup*”.

Sorting and Paging Controls

Sorting

Sorting controls are available for operations that return more than one row of data. As with filters, sorting controls are specified as parameters in the REST URL:

```
?sortBy=<columnName>&sortDir=<sortOrder>
```

- sortBy – The field name by which sorting is to be done.
- sortDir – The direction of sorting. Possible values are **asc** (ascending) and **desc** (descending).

When no sorting parameter is passed in the REST URL, the default sort field and sort order are assumed, as shown in the table below.

Table 3-3 Sort Parameters

Entity Type	Default Sort Field and Sort Order
Categories	Name (Asc)
Services	Name (Asc)
Offerings	Name (Asc)
Agents	Name (Asc)
People	First Name (Asc)
Organizational Units	Name (Asc)
Groups	Name (Asc)
Accounts	Name (Asc)
All Service Items	Row Id* (Asc)
Service Item Details	Row Id* (Asc)
Standards	Row Id* (Asc)
Custom Content	Row Id* (Asc)

Table 3-3 Sort Parameters

Requisitions	Submit Date (Asc), Requisition Id (Asc)
Requisition Entries	Requisition Entry ID (Asc)
Tasks	Task Name (Asc), Task Id (Asc)
Authorizations	Requisition Id (Asc), Task Id (Asc)
Agreements	Agreement Name (Asc)

Row Id is the physical order in which the records are inserted into the service item, standard and custom content tables. This corresponds to the column named PrimaryID in those tables.

Service Items, Standards and Custom Content entities support sorting for all columns. Other entities support sorting for specific fields only. For more information, see API Reference and Examples.

Examples

```
/nsapi/directory/people?sortBy=lastName&sortDir=asc
```

returns person records listed in ascending order by their last name.

```
/nsapi/directory/people?sortBy=login&sortDir=desc
```

returns person records listed in descending order by their login name.

Paging

Paging controls are available for operations that return more than one row of data. They are also specified as parameters in the REST URL:

```
?startRow=<x>&recordSize=<y>
```

- startRow – The starting row from which to fetch the records. The default value is 1.
- recordSize – Number of records to be fetched at a time. The default value is set in the Portal Designer common settings. The maximum number of records allowed is 50.

The above parameters and the total number of records returned are shown as attributes in the root tag of the REST XML response, as shown in the example below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<categories totalCount="11" recordSize="10" startRow="1">
  <category>
    <categoryId>1</categoryId>
    <categoryName><b>Consumer Services</b></categoryName>
    <description />
    .  .  .
```

If you specify a startRow greater than the number of records which would be retrieved, an HTTP 500 error is returned. If you specify a recordSize greater than the number of records which would be retrieved, all rows are returned.

Examples

In the following examples, assume there are 60 services defined in Service Designer and the maximum number of records specified for the nsAPI setting in Portal Designer is 30.

Example 1:

```
/nsapi/definition/servicedefs
```

returns the first 30 services. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="1">
  .  .  .
</services>
```

Example 2:

```
/nsapi/definition/servicedefs?startRow=4
```

returns 30 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="4">
  .  .  .
</services>
```

Example 3:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=5
```

returns 5 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="5" startRow="4">
  .  .  .
</services>
```

Example 4:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=35
```

returns 30 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="4">
.  .  .
</services>
```

Nested Entities

Certain entities contain a nested structure. The retrieval of only the first-level children or associated entities are supported by nsAPI. Parent entities and their child entities are summarized in the table.

Table 3-4 Nested Entities

Parent Entity	Child Entity (Entities)
Categories	Subcategories, Included Services, and Offerings for a category
Services	Categories, Keywords, and Bundled Services associated with a service
Person	OUs, Groups, Roles, Addresses, Contacts, Preferences, and Delegates associated with a person

For category, service or person searches that return more than one row of data in the result set, the associated entities are not fetched for performance reasons. The associated entities are available only in GET by Id or Name operations on the individual entities.

Examples

```
/nsapi/directory/people/loginname/<value>
```

returns the groups of which the person is a member:

```
<person>
  .  .  .
  <associatedGroups>
    <associatedGroup>
      <id>2</id>
      <name>group2</name>
    </associatedGroup>
    <associatedGroup>
    <id>5</id>
      <name>group5</name>
    </associatedGroup>
  </associatedGroups>
</person>
```

```
/nsapi/directory/people/id/<value>
```

returns the groups to which the person belongs.

```
/nsapi/directory/people
```

does **not** return the groups the person belongs to.

```
/nsapi/directory/people?ouname=<value>
```

does **not** return the groups the person belongs to.

Using nsAPI with JavaScript Portlets

nsAPI calls can be invoked in JavaScript portlets developed on the Portal Designer module. Using JavaScript, REST URL, AJAX, or Ext JS components, a portlet can be created to retrieve the data for the desired entities and rendered in a grid format.

Render Data in Ext JS Grid

Here is an example of using Ext JS to render a list of categories:

```
getUrl = "/RequestCenter/nsapi/definition/categories";

var proxy = new Ext.data.HttpProxy({
    url: getUrl,
    method: 'GET'
});

defaultPageSize = 5;
fieldList = [ "categoryId",
              "categoryName",
              "description",
              "topDescriptionEnabled",
              "topDescription",
              "topDescriptionURL",
```

```
                "middleDescriptionEnabled",
                "middleDescription",
                "middleDescriptionURL",
                "bottomDescriptionEnabled",
                "bottomDescription",
                "bottomDescriptionURL",
                "catalogTypeId",
                "catalogType",
                "isRoot",
                "descriptionURL",
                "categoryURL"]

displayList = [
                        {id: 'id', header: 'Id', width: 50,sortable: false,  dataIndex:
'categoryId'},
                        {header: 'Name',sortable: true, dataIndex: 'categoryName'},
                        {header: 'Description', dataIndex: 'description',sortable: false},
                        {header: 'TD Enabled', dataIndex:
'topDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Top Description', dataIndex:
'topDescription',hidden:true,sortable: false},
                        {header: 'Top Description URL', dataIndex:
'topDescriptionURL',hidden:true,sortable: false},
                        {header: 'Middle DescriptionEnabled', dataIndex:
'middleDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Middle Description', dataIndex:
'middleDescription',hidden:true,sortable: false},
                        {header: 'Middle Description URL', dataIndex:
'middleDescriptionURL',hidden:true,sortable: false},
                        {header: 'Bottom Description Enabled', dataIndex:
'bottomDescriptionEnabled',hidden:true,sortable: false},
                        {header: 'Bottom Description', dataIndex:
'bottomDescription',hidden:true,sortable: false},
                        {header: 'Bottom Description URL', dataIndex:
'bottomDescriptionURL',hidden:true,sortable: false},
                        {header: 'Catalog Type Id', dataIndex:
'catalogTypeId',hidden:true,sortable: false},
                        {header: 'Catalog Type', dataIndex:
'catalogType',hidden:true,sortable: false},
                        {header: 'isRoot', dataIndex: 'isRoot',hidden:true,sortable:
false},
                        {header: 'descriptionURL', dataIndex:
'descriptionURL',hidden:true,sortable: false},
                        {header: 'URL', dataIndex: 'categoryURL',sortable: false}
        ];


var store = new Ext.data.XmlStore({
    autoDestroy: true,
    proxy: proxy,
    root : "categories",
    record: 'category',
    idPath: 'rowId',
    totalProperty: '@totalCount',
    autoLoad: true,
    paramNames: {
      start: 'startRow',
      limit: 'recordSize',
      catName : 'categoryName'
    },
    fields: fieldList
});

var sortchange = function(obj,direction) {
```

```
    store.setBaseParam("sortBy","categoryName");
  store.setBaseParam("sortDir",direction.direction);
  store.load();
}


var grid =  new Ext.grid.GridPanel({
            renderTo :  '#divName#',
            store : store,
            autoWidth : true,
            height : 300,
            title : 'Category List',
            colModel: new Ext.grid.ColumnModel({
            defaults: {
               width: 120,
               sortable: true
            },
            columns:displayList,
             }),
            bbar : [new Ext.PagingToolbar({
                              store : store,
                              displayInfo : true,
                              pageSize : defaultPageSize,
                              params:{
                                       startRow: 1,
                                       recordSize: defaultPageSize
                                     }
               })]

   });

grid.on("sortchange",sortchange);
store.setBaseParam("recordSize",5);
store.load();
```

Get Logged-In User

Namespace variables for the currently logged-in user are available to be used in the JavaScript portlet; for example:

```
Ext.onReady(function() {
  /* Demonstrate JavaScript to get Logged-In user details */
  alert('PersonId: ' +nsAPP_CurrentUserId);
  alert('Login name: ' +nsAPP_CurrentUserLoginName);
  alert('First name: ' +nsAPP_CurrentUserFirstName);
  alert('Last name: ' +nsAPP_CurrentUserLastName);
  alert('HomeOUId: ' +nsAPP_CurrentUserHomeOuId);
}
```

Using nsAPI with JSR Portlets

Authentication

When accessed through JSR portlets, the nsAPI Java client can be used to invoke the login and logout operations.

```
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .
```

```
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username",   "password" ); // Login
by username, password.
.  .   .
.  .   .
nsApiClient.logout(); // Logout

.  .   .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter",
sessionId); // Login using current session id.
.  .   .
.  .   .
nsApiClient.logout();
```

Get Logged-In User

Here is an example of Spring-based JSR Portlet Controller to get the details of the logged-in user:

```
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import org.springframework.ui.Model;
import org.springframework.stereotype.Controller;
import com.newscale.nsapi.directory.person.Person;
…
@Controller
public class MyJSRController {
private NSApiClient nsApiClient = getNSApiClient();
public NSApiClient getNsApiClient() {
return nsApiClient;
  }
public void setNsApiClient(NSApiClient nsApiClient) {
this.nsApiClient = nsApiClient;
}
@RequestMapping("VIEW")
@RenderMapping("NORMAL")
public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
 nsApiClient.login("http://<AppServer host>:<port>/RequestCenter",
request.getPortletSession().getId());
// Get Currently Logged-in user from nsAPI client
Person persons  = nsApiClient.getDirectory().getCurrentUser();
// Get user info
long personId = persons.getPersonId());
long homeOUId = persons.getHomeOrganizationalUnitId());
String firstName =persons.getFirstName());
String lastName =persons.getLastName());
String username =persons.getLogin());
  }
}
```

Get Operations

Here are some sample code snippets that illustrate the methods for fetching entities. For further details on these methods, see the Javadoc for the individual entity classes.

- Get person by Id

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
```

```
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

Person person = nsApiClient.getDirectory().getPersonById(123);

/* This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/id/123
*/
.  .  .
nsApiClient.logout();
```

- Get person by Name

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");
Person person = nsApiClient.getDirectory().getPersonByLoginName("jsmith");
/*
This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/loginname/jsmith
*/

.  .  .
nsApiClient.logout();
```

- Get all people

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
import java.util.List;
import org.apache.commons.collections.map.MultiValueMap;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

PersonList persons = nsApiClient.getDirectory().getPeople(null);

/*
 This is the equivalent of REST URL
    http://<serverURL>/RequestCenter/nsapi/directory/people
*/
.  .  .
nsApiClient.logout();
```

- Get person with query filter

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
```

```
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

MultiValueMap filterMap = new MultiValueMap();//this can be used to specify multiple
filter criteria
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

/*
 This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people?name=John
*/
.  .   .
nsApiClient.logout();
```

- Get person with multiple query filters

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
.  .   .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

MultiValueMap filterMap = new MultiValueMap();
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTBY, "lastName");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTDIR, "asc");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

// This is the equivalent of REST URL
//    http://<serverURL>/RequestCenter/nsapi/directory/people?name
//    =John&sortBy=lastName&sortDir=asc
// search for people by the name John,
// sort the result set by Last Name in ascending order
.  .   .
nsApiClient.logout();
```

Post Operations

Here are some sample code snippets that illustrate the methods for creating/updating a person and taking actions on tasks. For further details on these methods, see the Javadoc for the individual entity classes.

- Update person

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapi.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
```

```
import com.newscale.nsapiclient.NSApiClient;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");
Person person = NSApiClient.getDirectory().getPersonById(123);
person.setLastName("Smith");
Person persons = NSApiClient.getDirectory().updatePerson(person);
/*
This is the equivalent of posting XML of person 123 with last name changed to "Smith" to
the REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/update
*/
.  .  .
nsApiClient.logout();
```

- Complete delivery task

```
package com.newscale.nsapiclient.transaction;
import com.newscale.nsapiclient.transaction.task.TaskAction;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.transaction.Transaction;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
.  .  .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter","username","password");

// This is the equivalent of REST URL
//    http://<serverURL>/RequestCenter/nsapi/tasks/10/done

NSApiClient.getTransaction().completeTask(10);
nsApiClient.logout();
```

API Reference and Examples

The javadoc for nsAPI can be located in the Image folder of the product installer. Examples for invoking the nsAPI through REST URL and Java client are provided for each of the supported entities.

Definitional Data

Categories

Table 3-5 Categories API Table

Area	Examples
Core API	**Get all categories** By default only categories of type “Service Catalog” are returned. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories?catalogType=serviceCatalog Java Example: CategoryList categories = NSApiClient.getDefinition().getCategories(null);
	Gets all consumer service categories Returns all categories of “Consumer Services” type. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories Java Example: `CategoryList categories = NSApiClient.getDefinition().getCategories("serviceCatalog");`
	Get consumer service category by Name Returns a service catalog category with the name specified. Nested entities (subcategories and included services) are fetched. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName> Java Example: `Category categories = NSApiClient.getDefinition().getCategoryByName("<categoryName>");`
	Gets consumer service category by Id Returns a service catalog category with the Id specified. Nested entities (subcategories and included services) are fetched. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/categories/id/<categoryId> Java Example: `Category categories = NSApiClient. getDefinition().getCategoriesById(<categoryId>);`

Table 3-5 Categories API Table

Area	Examples

	Gets service offering category by Name
	Nested entities (subcategories and included offerings) are included. Returns a service offering category with the name specified.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName>?catalogType=offeringCatalog
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put(QUERYPARAM_CATALOG_TYPE, "offeringCatalog");`
	`Category categories =`
	`NSApiClient.getDefinition().getCategoryByName("<categoryName>");`

Filters	**Category Name Filters**
	Search is case-sensitive.
	StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.
	Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard is applied.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/categories?name=<wildcardValue>&catalogType=serviceCatalog
	`Java Example:`
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put(QUERYPARAM_CATALOG_TYPE, "serviceCatalog");`
	`paramsMap.put(QUERYPARAM_NAME, "<wildcardValue>");`
	`CategoryList categories =`
	`NSApiClient.getDefinition().getCategories(paramsMap);`

	Service User Access Filter
	Returns all the Categories for the service immediately under it or under its hierarchy for the logged in user with permissions to order.
	REST URL:
	http://<ServiceUrl>/RequestCenter/nsapi/definition/categories?userContext=<true or false>
	`Java Example:`
	`MultiValueMap paramMap = new MultiValueMap();`
	`paramMap.put(QUERYPARAM_USERCONTEXT, "true");`
	`NSApiClient.getDefinition().getCategories(paramMap);`

Table 3-5 Categories API Table

Area	Examples
Sort Column(s)	CategoryName
Response XML	<categories totalCount="x" recordSize="y" startRow="z"> <category> . . . <associatedServices> . . . <associatedService> <description> </description> <id></id> <imageURL></imageURL> <name> </name> <status> </status> </associatedService> . . . </associatedServices> . . . </category> </categories>

Environment

Table 3-6 Environment API Table

Area	Examples
CoreAPI	Fetch AMQP Environment details using Public GUI ID (Public GUI ID: Go to Administration > Settings > specify Public/Private Key) Returns all the AMQP Environment details. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/messagebroker/service/<serviceName>?publicKeyGUID=... http://<ServerURL>/RequestCenter/nsapi/messagebroker/overview?publicKeyGUID=... **Status Code**: 200 OK **Status Code**: 400 Bad request Refer to **Integrating with AMQP** section for details on Encrypt Credentials using Public Key GUID.
	Fetch cloud connection details by Provider type, short name Returns Cloud type (such as, UCSD/ICFD/PUPPET) connection information for the specified short name including the SSL Certificate in Base64 encoded format. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/connections/provider/{providerName}/shortName/{shortName} {Provider name} can be UCSD/ICFD/PUPPET and {psc} is the short name identifier used in the connection. If the connection incident is https, then API returns root certificate used for the connection. SSL Certificate (Base 64 format) in the connection is also returned. **Status Code**: 200 OK **Status Code**: 400 Bad request Sample Response: { "connections": { "connection": [{ "primaryID": 2, "userName": "admin", "protocol": "https", "hostName": "172.21.38.151", "description": "...", "url": "https://172.21.38.151:443/app/api/rest", "port": 443,

Table 3-6 Environment API Table

Area	Examples
	Fetch cloud connection details by Provider type, short name using Public GUI ID If the external systems Public Key GUID is passed as a Query parameter, the credentials (password and token) are encrypted (using PO Secure String Format) in the response with Public Key associated to the GUID. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/connections/provider/{providerName}/shortName/{shortName}?publicKeyGUID=.. When Public Key GUID is not passed or incorrect - **Status Code**: 400 Bad request is returned.

Table 3-6 Environment API Table

Area	Examples
	Create AMQP Server Details Creates the SSL and non-SSL connections with the AMQP server and update the AMQP Server details in the web interface. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/saveConnection For connection using SSL certificate: `{` `  "id": "row-1",` `  "shortName": "am2",` `  "description": "amqp_server2",` `  "protocol": "SSL",` `  "hostName": "10.78.0.247",` `  "port": "5671",` `  "un": "pscuser",` `  "up": "!@^_cHNjdXNlcg==_^@!",` `  "pollerEnabled": "0",` `  "provider": "AMQP",` `  "exportUser": "0",` ` "cert": "-----BEGIN CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAh NeVRlc3RDQTAeFw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG 9zdG5hbWUpMQ8wDQYD\nVQQKDAZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBA QCq8v2yJT7tv+nOwFSo\nEE1c0oVg3skd86JN7jVJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw +a+0ext2WRHgQLTMPvVO\nA86jwuPd/bhUxXg8jeEfE4V/1Seci9Xz+5VxqCybCNOzJQ12/vLXvIJ K43U/+1GdXnpWxFaF0yd0\nht3iUy6mfUAHfNMI2SOfJwbbdUaMyD0/Krsiu+X+vFQBUDmM7Y0pri ItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvkTET3tIVsKDi1Fn9Id KQuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCMAAwCwYD VR0PBAQDAgUgMBMGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4ku u/dpLeFWNUhb5Uyk6AqbTRAyD\nlq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94g q8luSem1Z8lQzUCOCE6rsaznrw\nm9jJO7gXA5SSmy7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEP REqnshs6f6bQSm8tzSNETy3OyHL\npUo7YVTBCfJMQ/e2nZxCJSDuEL6QaIL4kmEYeu8j/1RplBAo fMkDfDe+yMx2MJYl+MopVggGexpa\nMqybCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNVenCpGCtgGIc upRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----END CERTIFICATE-----",` ` "ovCert": "",` ` "skipValidateCertificate": "0",` ` "virtualHost": "psc",` ` "publicKey": "none",` ` "secureStringFormat": "Bytes",` ` "serverDownNotification": "none",` ` "recoveryInterval": "5",` ` "inboundQueue": "psc_inbound_queue",` ` "messageType": "XML",` ` "basicAuthentication": "0",` ` "isTriggeredByPoller": false` `}`

Table 3-6 Environment API Table

Area	Examples

	For connection without SSL certificate:
	<pre>{
	  "id": "row-1",
	  "shortName": "am1",
	  "description": "amqp_server1",
	  "protocol": "TCP",
	  "hostName": "10.78.0.247",
	  "port": "5672",
	  "un": "admin",
	  "up": "!@^_Y2lzY28xMjM=_^@!",
	  "pollerEnabled": "0",
	  "provider": "AMQP",
	  "exportUser": "0",
	  "cert": "",
	  "ovCert": "",
	  "skipValidateCertificate": "0",
	  "virtualHost": "default",
	  "publicKey": "none",
	  "secureStringFormat": "Bytes",
	  "serverDownNotification": "none",
	  "recoveryInterval": "5",
	  "inboundQueue": "psc_inbound_queue",
	  "messageType": "XML",
	  "basicAuthentication": "0",
	  "isTriggeredByPoller": false
	}</pre>
	Note Password value is to be passed as a base64 encoded value prefixed by !@^_ and suffixed by _^@!. Base64 encoded value can be generated using javascript btoa method.
	
	Note ShortName can have max 3 characters.

	Delete AMQP Connection
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/deleteConnection
	<pre>{
	  "shortName": "am3",
	  "provider": "AMQP"
	  "protocol": "TCP"
	}</pre>

Table 3-6 Environment API Table

Area	Examples

	Gets all the Integration types
	GET REST URL:
	http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getintegrationtypes
	Sample Response:
	{
	"List": [
	{
	"id": "1",
	"integrationType": "Internal"
	},
	{
	"id": "2",
	"integrationType": "Custom"
	}
	]
	}

	Filters the Integration based on type (Internal or Custom) GET REST URL: http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getCloudIntegrationsList?type=1 Where, type=1 indicates *Internal* and type=2 indicates *Custom* To filter based on the internal integration: http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getCloudIntegrationsList?startRow=1&recordsPerPage=10&type=1&name=&integrationTypes=[CLIQR,UCSD,UCSPM,PO,AMQP,WEBCONN,SSO,GUACAMOLE]

Table 3-6 Environment API Table

Area	Examples

Sample Response:


```
{
    "Result": {
        "recordsReturned": 20,
        "totalRecords": 1,
        "startIndex": 0,
        "records": [
            {
                "authenticationScheme": "",
                "baseTemplate": "Cloud Center Base Template Service",
                "baseTemplateID": 121,
                "certficateData": "",
                "clientId": "null",
                "core": "",
                "description": "CLN",
                "environmentId": "null",
                "exportUser": 0,
                "governanceModeEnabled": 0,
                "hostName": "172.21.37.62",
                "msp": 0,
                "password": "AD46967871AFB825",
                "pollerEnabled": 0,
                "port": 443,
                "previousBaseTemplateID": 0,
                "primaryID": 1,
                "protocol": "https",
                "providerType": "CLIQR",
                "realm": "",
                "serviceGroupID": 14,
                "shortName": "CLN",
                "skipValidateCertificate": 1,
                "status": "Active",
                "system": "Cisco CloudCenter",
                "token": "",
                "url": "https://172.21.37.62:443/v1/users",
                "userName": "cliqradmin",
                "version": ""
            }
        ]
    },
    "Roles": {
        "IA": false,
        "SOA": [],
        "createSGAccess": false,
        "siteAdmin": true
    }
}
```

Table 3-6 Environment API Table

Area	Examples

	Gets the User list for the custom group's SOA role
	GET REST URL:
	http://<ServerURL>/RequestCenter/nsapi/ucsd/v1/soauserslist/custom/{serviceGroupId}
	Sample Response:
	`{`
	`"ConnectionDetails": {`
	`"soaRoleID": "567"`
	`},`
	`"List": [`
	`{`
	`"OwnerName": "user2 IA",`
	`"Type": "Service Operations Administrator",`
	`"OwnerID": "11",`
	`"isLoggedinUser": "false"`
	`},`
	`{`
	`"OwnerName": "s1user Platform",`
	`"Type": "Service Operations Administrator",`
	`"OwnerID": "7",`
	`"isLoggedinUser": "false"`
	`},`

	Shares/transfers SOA role to users of a custom group
	PUT REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/v1/transfersoa
	Sample Payload:
	`For Custom:`
	`{"Group_ID":"22","Opp_Type":"share","Person_ID": ["11"],"Provider_Type":"CUSTOM"}`
	
	`For Internal:`
	`{Short_Name: "CC1", Opp_Type: "share", Person_ID: ["11"], Provider_Type: "CLIQR"}`

Localization

Table 3-1 Localization API Table

Area	Examples
CoreAPI	**Get all languages along with the locale details** Returns all languages along with the locale details. REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/locales Sample response: [{ "active": 1, "defaultCurrency": "REAL", "defaultDateSep": "/", "defaultLongDate": "dd 'de' MMMM 'de' yyyy", "defaultShortDate": "dd/MM/yy", "defaultTime": "HH:mm:ss", "localeCode": "BRPT", "localeID": 9, "name": "Brazilian-Portuguese" }, { "active": 1, "defaultCurrency": "YEN", "defaultDateSep": "/", "defaultLongDate": "MMMM dd, yyyy", "defaultShortDate": "MM/dd/yyyy", "defaultTime": "h:mm aa", "localeCode": "CNZH", "localeID": 6, "name": "Chinese (Simplified)" }, ----- -----]

Table 3-1 Localization API Table

Area	Examples
	Get locales specific javascripts strings Returns javascript strings specific to the locales passed in the input. Javascript string for the English locale is returned by default. If an unknown locale is passed in the request, the request, returns the response for the known locales and shows no error response for the unknown locale. REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings?locales=KRKO, ALSQABC, BRBT, CNZH Sample response: [{ "StringID": "js1", "US English": "src_eng22", "Chinese (Simplified)": "", "Brazilian-Portuguese": "", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "Chinese (Simplified)": "", "Brazilian-Portuguese": "", "Korean": "5" }, -------- ---------]

Table 3-1 Localization API Table

Area	Examples
	Get all available javascript strings for all available languages Returns all available javascript strings for all available languages i.e. there is no limitation on the number of languages. REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings Sample response: [{ "StringID": "js1", "US English": "src_eng22", "German": "", "Spanish": "", "French": "js string french", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "German": "", "Spanish": "", "French": “french js2", "Korean": "5" }, ---------- ----------]

Table 3-1 Localization API Table

Area	Examples
	Post all available javascript strings for all available languages REST URL: http://<ServerURL>/RequestCenter/nsapi/localization/javascriptstrings Sample Request: [{ "StringID": "js1", "US English": "src_eng22", "German": "", "Spanish": "", "French": "js string french", "Korean": "update5" }, { "StringID": "js2", "US English": "src_eng2", "German": "", "Spanish": "", "French": “french js2", "Korean": "5" }, ---------- ----------] Sample Success Response: Create/Update of JavaScript Strings is completed successfully. Sample Error Response: The language name(s) in json content is either misspelled or not available in the product .Please verify 'French (Switzerland)’

Services

Table 3-7 Services API Table

Area	Examples
CoreAPI	**Get all services** Returns all services. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs Java Example: `ServiceList services = NSApiClient.getDefinition().getServices(null);`

	Get all services by service catalog wildcard search
	Returns all services with service name, service description, category, or keyword that matches the search string. Leading wildcard is not supported.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?search={wildcardValue*}
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("search", "<wildcardValue*>");`
	`ServiceList services = NSApiClient.getDefinition().getServices(paramsMap);`

	Get service by Id Nested entities (associated categories and keywords) are fetched for getById and getByName only. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/<serviceId> Java Example: `Service services = NSApiClient.getDefinition().getServiceById(<serviceId>);`

	Get service by Name
	Nested entities (associated categories and keywords) are fetched for getbyId and getByName only.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/name/<serviceName>
	Java Example:
	`Service services =`
	`NSApiClient.getDefinition().getServiceByName("<serviceName>");`

Table 3-7 Services API Table

Area	Examples
	Get all services in a category Returns all services associated with the category REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?categoryName=<categoryName> Java Example: `ServiceList services =NSApiClient.getDefinition().getServiceByCategoryName("<categoryName>");`
	Gets all services by keyword Returns all services associated with the keyword REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?keywordName=<keyword> Java Example: `ServiceList services = NSApiClient.getDefinition().getServiceByKeyword("<keyword>");`

Filters	**Services Name Filters**
	Search is case-sensitive.
	StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.
	Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?name=<wildcardValue>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("name","<wildcardValue>");`
	`ServiceList servicesList = NSApiClient.getDefinition().getServices(paramsMap)`

Sort Column(s)	ServiceName

Response XML	`<services totalCount="x" recordSize="y" startRow="z">`
	`<service>`
	`. . .`
	`<includedServices>`
	`. . .`
	`<includedService>`
	`<id></id>`
	`<name> </name>`
	`</includedService>`
	`. . .`
	`</includedServices>`
	`. . .`
	`</service>`
	`</services>`

Table 3-7 Services API Table

Area	Examples

Categories Extensions and Facets	Create/Update services with categories, facets and extensions
	Use POST for New Service - an error occurs if Service already exists. Creates/updates Categories, Facets and Extensions and associates it to the Service.
	Use PUT to Update existing Service - an error occurs if Service does not exist Creates/updates Categories, Facets and Extensions and associates it to the Service.
	REST URL:
	/RequestCenter/nsapi/cloud/marketplace/service
	Sample Payload:
	{
	"***serviceName***": "TestServiceT225", "***serviceId***": "...", "***templateName***": "TestService", "***templateCategory***" : "TemplateCategory", "***description***": "test", "imageData": "", "imageURL": "$RC_IMAGEPATH$/ABCServer.jpeg", "overview": "<p>Product Details<\/strong>
	Version: 1.23.0-0 on Ubuntu 12.04.4
	", "serviceGroupName": "TestGrp",
	"**categories**": [
	{
	"name": "TestABCImageURL2", "description" : "TestDesc", "imageURL": "$RC_IMAGEPATH$/ABCServer.jpeg"}
	],
	"**facetes**": {
	"facetlogicname1" : {"displayName" : "Facet 1", "values" : ["f1val1", "f1val2", "f1val3", "f1val4"]},
	"facetlogicname2" : {"displayName" : "Facet 2", "values" : ["f2val1", "f2val2"]}
	},
	"**extensions**": {
	"extensionlogicname1" : {"displayName" : "Extension 1", "values" : "extval1"},
	"extensionlogicname2" : {"displayName" : "Extension 2", "values" : "extval2"}
	}
	}
	Sample Response XML:
	{ "JSONObject":
	{
	"serviceName": "TestNewJSON1-TestNew",
	"serviceEndpoint":
	"/RequestCenter/nsapi/definition/servicedefs/name/TestNewJSON1-TestNew"
	}
	}

Table 3-7 Services API Table

Area	Examples

	Unlink Categories from a Service
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkCategories
	Sample Payload:
	{"categories" :["catname1","catname2",...]}
	Status Code: 200 OK
	Status Code: 400 Bad request
	Sample Response:
	{
	status-message:
	{
	code: "...."
	value: "....."
	}
	}

Table 3-7 Services API Table

Area	Examples

	Responses in various scenarios:
	1. If the serviceId passed does not exist:
	Http code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. If any of the Category passed is not associated with Service:
	Http Code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0003"
	errormessage: "Categories could not be unlinked from the Service."
	}
	}
	3. On any general Exception, the JSON Response is similar to:
	Http code: 500
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	4. If the logged in user does not have RBAC permissions on the Service identified by serviceId:
	Http code: 401
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object."
	}
	}

Table 3-7 Services API Table

Area	Examples

	5. On Successful unlinking of Categories:
	Http code: 200
	Http Response:
	{
	status-message:
	{
	code: "Success"
	value: "Categories Unlinking is done Successfully for the Service."
	}
	}

Table 3-7 Services API Table

Area	Examples

	Unlink Facets from a Service
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkFacets
	Payload takes Facet Name (<facetName> is the Name of facet and not the Display name) and its values in the following JSON Array format:
	{
	"<facets>" :["<facet1>","<facet2>",...]
	}
	Responses in various scenarios:
	1. If the serviceId passed does not exist:
	Http code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. If a facet is not associated with the Service:
	Http Code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0005"
	errormessage: "Facets could not be unlinked from the Service."
	}
	}
	3. If the logged in user does not have RBAC permissions on the Service identified by serviceId:
	Http code: 401
	Http Response:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

Table 3-7 Services API Table

Area	Examples

	4. On any general Exception, the JSON Response is similar to:
	Http code: 500
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. On Successful unlinking of facets:
	Http code: 200
	Http Response:
	{
	status-message:
	{
	code: "Success"
	value: "Facets Unlinking is done Successfully for the Service."
	}
	}

Table 3-7 Services API Table

Area	Examples

	Unlink values from a multivalued Facet. This API is applicable only for multivalued facets.
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkMultiValueFacet
	Payload takes Facet Name (<facetName> is the Name of facet and not the Display name) and its values in the following JSON Array format:
	{
	"<facetName>" :["<facetValue1>","<facetValue2>",...]
	}
	Responses in various scenarios:
	1. If the serviceId passed does not exist:
	Http code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. If a facet value is not associated with the Service:
	Http Code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0005"
	errormessage: "Facet values could not be unlinked from the Service."
	}
	}

Table 3-7 Services API Table

Area	Examples

	3. If the logged in user does not have RBAC permissions on the Service identified by serviceId:
	Http code: 401
	Http Response:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}
	4. On any general Exception, the JSON Response is similar to:
	Http code: 500
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. On Successful unlinking of facet values:
	Http code: 200
	Http Response:
	{
	status-message:
	{
	code: "Success"
	value: "Facet Values Unlinking is done Successfully for the Service."
	}
	}

Table 3-7 Services API Table

Area	Examples

	Unlink Service Extension from a Service
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/{serviceId}/delinkServiceExtensions
	Payload takes Service Extension Name (<serviceExtensions> is the logical Name and not the Display name) in the following JSON Array format:
	{
	"serviceExtensions" : ["serExt1","serExt2",...]}
	Responses in various scenarios:
	1. If the serviceId passed does not exist:
	Http code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	2. If a Service Extension is not associated with the Service:
	Http Code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0004"
	errormessage: "Servce Extensions could not be unlinked from the Service."
	}
	}
	3. If the logged in user does not have RBAC permissions on the Service identified by serviceId:
	Http code: 401
	Http Response:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

Table 3-7 Services API Table

Area	Examples

	4. On any general Exception, the JSON Response is similar to:
	Http code: 500
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Some NSAPI Exception Occurred."
	}
	}
	5. On Successful unlinking of Service Extensions:
	Http code: 200
	Http Response:
	{
	status-message:
	{
	code: "Success"
	value: "Service Extensions Unlinking is done Successfully for the Service."
	}
	}

Table 3-7 Services API Table

Area	Examples

	Deleting a Service
	This API takes the Id of the Service to be deleted in the following URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/{id}
	HTTP Method: DELETE
	1. On Successful deletion of a Service:
	Http code: 200
	Http Response:
	{
	status message:
	{
	code: “Success”
	"value": "Service with ServiceID {id} deleted successfully.”
	}
	}
	2. If the serviceId passed does not exist:
	Http code: 404
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0001"
	errormessage: "ServiceId does not exist."
	}
	}
	3. Service has a requisition and cannot be deleted
	Http code: 403
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "SERV_0002"
	errormessage: "Service Cannot be deleted since it already exists amongst requisitions"
	}
	}

Table 3-7 Services API Table

Area	Examples

	4. General Exception
	Http code: 500
	Http Response:
	{
	nsapi-error-response:
	{
	errorcode: "EXC_0001"
	errormessage: "Delete Service Failed"
	}
	}
	5. If the logged in user does not have RBAC permissions on the Service identified by serviceId:
	Http code: 401
	Http Response:
	nsapi-error-response:
	{
	errorcode: "NSAPI_ERROR_004"
	errormessage: "The user does not have sufficient permission to perform the operation on this object"
	}
	}

Table 3-7 Services API Table

Area	Examples

	Copying a service
	Copy a service definition, by reusing the existing active form component (AFC) and dictionary or by cloning a service along with AFCs, dictionaries, email templates, and scripts.
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/service
	Copy a Service:
	To copy a service definition by reusing the existing Active Form Components (AFC) and Dictionaries.
	Sample Input:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "false",`
	`}`
	Clone a Service:
	To copy all Forms and Dictionaries in the service with the Prefix.
	Sample Input:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "true",`
	`"Form Name Prefix": "NewForm",`
	`"Dictionary Name Prefix": "NewDictionary"`
	`}`
	To copy only required forms and AFC's in the service.
	Sample Input:
	`{`
	`"Service ID": "88",`
	`"New Service Name": "service",`
	`"Deep Copy": "true",`
	`"Forms":{`
	`"Old form name":{`
	`"New Form Name": "Deep Copy Form",`
	`"New Form Group Name": "Deep copy form group",`
	`"Dictionaries":{`
	`"Old dictionary name":{`
	`"New Dictionary Name": "Deep_Copy_Dictionary",`
	`"New Dictionary Group Name": "new group"`
	`}`
	`}`
	`}`
	`},`
	`}`

Table 3-7 Services API Table

Area	Examples
	Gets all the services belonging to the specified connection REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/connection/{connectionid}/servicelist Method: GET **Note** Use connection ID and not connection short name in this API.
	Gets services which belongs to the specified service group ID **REST URL:** http://<ServerURL>/RequestCenter/nsapi/definition/servicegroup/{servicegroupid}/servicelist Method: GET
	Assigns services to tenants REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceassignment?action=”grant” Method: PUT Java Example: `{ "ServiceTenantsAssignmentsList": { "serviceTenantsAssignmentAPIDTOs": [ { "serviceId": 134, "tenantIds": [17] } ] } }`

Table 3-7 Services API Table

Area	Examples
	Unassigns services to tenants REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceassignment?action=”revoke” Method: PUT Java Example: `{ "ServiceTenantsAssignmentsList": { "serviceTenantsAssignmentAPIDTOs": [ { "serviceId": 134, "tenantIds": [17] } ] } }`
	Makes services orderable for tenant users **Note** The calling user must be the team admin of the project tenant to use this API. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceorder?action=”grant” Method: PUT Java Example: `{ "TenantServicesAssignmentAPIDTO": { "tenantId" : 4, "serviceIds" : [130] } }`

Table 3-7 Services API Table

Area	Examples

	Makes services unorderable for tenant users
	
	Note The calling user must be the team admin of the project tenant to use this API.
	
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/service/permission/serviceorder?action=”revoke”
	Method: PUT
	Java Example:
	`{`
	`"TenantServicesAssignmentAPIDTO": {`
	`"tenantId" : 3,`
	`"serviceIds" : [130]`
	`}`
	`}`

	Gets orderable services for a tenant REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantId}/services/orderable Method: GET Sample Response:
Core	**Fetches orderable services for a tenant** REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantId}/services/orderable Method: GET
inherited	• true: returns only inherited services • false: returns only services made directly orderable When called without this parameter, the API returns both inherited and direct services. Sample REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/8/services/orderable?inherited=true

Table 3-7 Services API Table

Area	Examples
pagination	• startRow • recordSize Sample REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantId}/services/orderable?recordSize=10 &startRow=1
sorting	• sortBy: Name • sortDir: ASC/DESC Sample REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/8/services/orderable?sortBy=Name&sortDir=DESC
search	• serviceName • description Search is an AND operation and it is not case sensitive. Sample REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/8/services/orderable?serviceName=Ser*1&description=des

Table 3-7 Services API Table

Area	Examples

Sample Response	{
	"orderableServiceProjections": [
	{
	"description": "Des",
	"id": 7,
	"inherited": true,
	"name": "Service1"
	},
	{
	"description": "Des",
	"id": 8,
	"inherited": true,
	"name": "Service2"
	},
	{
	"description": "Des",
	"id": 9,
	"inherited": true,
	"name": "Service3"
	},
	{
	"description": "",
	"id": 10,
	"inherited": false,
	"name": "Service4"
	}
	],
	"recordSize": 20,
	"startRow": 1,
	"totalCount": 4
	}

Table 3-7 Services API Table

Area	Examples

	Creates the Custom SG and SIG for the specified name
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions
	Method: POST
	Sample Payload:
	{
	"Instance":[
	{
	"serviceGroupName":"IphoneSG",
	"serviceItemGroupName":"IphoneSIG"
	}
	]
	}
	Sample Response:
	{
	"nsapi-response":{
	"status-messages":[
	{
	"code":"TA_310",
	"value":"Created Successfully"
	}
	]
	}
	}
	{
	"nsapi-error-response": {
	"status-messages": [
	{
	"code": "TA_311",
	"value": "A Service Item Group with the same name already exists. Please use a unique name for this Service Item Group."
	}
	]
	}
	}

Table 3-7 Services API Table

Area	Examples

	Updates the Service Group name and Service Item Group name
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions
	Method: PUT
	Note If Service Item Group does not exist, then Service Item Group is created and associate to the Service Group.
	Sample Payload:
	{
	group:[
	{
	serviceGroupId:"19",
	serviceGroupName:"dsds1",
	serviceItemGroupName:""
	}
	]
	}

	Get the SG and SIG details for the given Service group
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/servicegroupnamedetails/{servicegroupID}
	Method: GET
	Sample Response:
	{
	"List":[
	{
	"serviceItemGroupId":"",
	"serviceItemGroupName":"",
	"tenantRelevant":false,
	"serviceGroupName":"dsds1"
	}
	]
	}

Table 3-7 Services API Table

Area	Examples
	Deletes the custom service group and Service Item Group (if SIG is associated to SG) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/service/solutions?id={primaryID} Method: DELETE **Note** To obtain the primaryID use GET API http://<ServerURL>/RequestCenter/nsapi/ucsd/discovery/getCloudIntegrationsList?startRow=1&type=2.
	Gets list of connections REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/services/connection Method: GET
	Get specific assets of the specified connection REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/shortName/{ConnectioinShortName}/assets/{assetType} Method: GET
	Gets list of all the assets of the specified connection REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/shortName/{ConnectioinShortName} Method: GET
	Gets all the service assets of all the connections REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets Method: GET
	Gets list of specific asset for all connections REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/serviceassets/assets/{AssetType} Method: GET

Agents

Table 3-2 Agents API Table

Area	Examples
CoreAPI	Stop an Agent Stops a service link agent that is active. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agent name>/stop
	Start an Agent Start a service link agent that is inactive. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agent name>/start
Response xml	`Agent<agent name> <started/stopped> successfully`

Table 3-3 Agents API Table

Area	Examples
CoreAPI	**Get all agents** Returns all agents. GET REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agents Java Example: `AgentList agents = NSApiClient.getDefinition().getAgents(null);`
	Get agent by Id Outbound and inbound properties are fetched for getById and getByName only. GET REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agents/id/<agentId> Java Example: `Agent agents = NSApiClient.getDefinition().getAgentById(<agentId>);`
	Get agent by Name Outbound and inbound properties are fetched for getById and getByName only. GET REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agents/name/<agentName> Java Example: `Agent Agents = NSApiClient.getDefinition().getAgentByName("<agentName>");`

Table 3-3 Agents API Table

Area	Examples
	Get Status of an Agent Retrieves the status of the agent. GET REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agent/<agentName>/status
Filters	**Agent Name Filters** Search is case-sensitive. StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored. Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agents?name=<wild cardValue> Java Example: `AgentList agents = NSApiClient.getDefinition().getAgentsByFilter("<wildcardValue>");`
Sort Column(s)	AgentName
Response XML	`<agents totalCount="x" recordSize="y" startRow="z"> <agent> . . . </agent> </agents>`

Agreements

Table 3-8 Agreements API Table

Area	Examples
Get APIs	**Get agreement by Id** REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/id/<agreementId>
	Get agreement by Name REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/name/<agreementName>
	Get all master agreements Returns all master agreements. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/master
	Get all project agreements Returns all project agreements. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/project
	Get all agreements for account name Returns all agreements for that Account Name (Exact match with ignore case). REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/accountname/{accountName}
	Get agreement for account name Returns agreement for that account name (Account Name is mandatory in this area). REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/name/{agreementname}?&accountName={accountName}
Sort Column (s)	AgreementName

Response XML	`<agreements totalCount="x" recordSize="y" startRow="z">`
	`  <agreement>`
	`  . . .`
	`  </agreement>`
	`</agreements>`

Table 3-9 Usage Scenarios Table

Area	Examples
Other Usage Scenarios	Get all OUs under an account Returns all OUs associated under this account. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/agreements/agreementorgunitlist?&accountName={accountName}

	Create an agreement
	REST URL (HTTP POST):
	http://<ServerURL>/RequestCenter/nsapi/definition/agreements/create
	`<agreement agreementId="123" name="Sample Agreement Name">`
	`parentAgreementName="Sample Parent Agreement Name" >`
	`<description>Sample Agreement Description</description>`
	`<agreementTemplateName>Sample Agreement Template Name</agreementTemplateName>`
	`</agreement>`

Table 3-9 Usage Scenarios Table

Area	Examples
	Delete an agreement REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/definition/agreements/delete <agreement agreementId="123" name="Sample Agreement Name"> </agreement>
	Update an agreement REST URL (HTTP PUT): http://<ServerURL>/RequestCenter/nsapi/definition/agreements/update <agreement agreementId="123" name="Sample Agreement Name" accountName="Sample Account Name" parentAgreementName="Sample Parent Agreement Name"> <description>Sample Agreement Description</description> <agreementAttributes> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 1" aggregateFunction="Count" serviceItemAttribute="" quota="123" /> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 2" aggregateFunction="Count" serviceItemAttribute="" quota="456" /> <agreementAttribute serviceItemTypeName="Sample Service Item Type Name 2" aggregateFunction="Sum" serviceItemAttribute="Sample Attribute Name" quota="789" /> </agreementAttributes> <organizationalUnits> <organizationalUnit> <organizationalUnitName>Sample OU Name</organizationalUnitName> <organizationalUnitType>Business Unit</organizationalUnitType> </organizationalUnit> </organizationalUnits> </agreement>

Billing Rate

Billing Rate Group

Table 3-10 Billing Rate Group API Table

Area	Examples
Core API	Get all billing rate groups Returns all billing rate groups. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups
	Get rate group by Id Returns the billing rate group by ID. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/id/{id}
	Get rate group by name Returns the billing rate group by name. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/name/{name}
	Get rate group by wild card name search Returns the billing rate groups by wild card name search. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups?name={wild card name}
	Create a rate group (One Rate group Object only) REST URL (HTTP POST): http:// <ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/create POST DATA: <rateGroup name=”Sample Rate Group” description=”Sample rate group description”> <rateTables> <rateTable id=”1” name=”First”/> <rateTable id=”2” name=”Second”/> <rateTable id=”3” name=”Third”/> </rateTables> </rateGroup>

Table 3-10 Billing Rate Group API Table

Area	Examples
	Update a rate group (One Rate Group Object only) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/update BODY <rateGroup id=”123” name=”Sample Rate Group” description=”Sample rate group description”> <rateTables> <rateTable id=”1” name=”First”/> <rateTable id=”2” name=”Second”/> <rateTable id=”3” name=”Third”/> </rateTables> </rateGroup>
	Delete a rate group (One Rate Group Object only) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/rategroups/delete BODY: <rateGroup id=”123” name=”Sample Rate Group Name” />

Billing Rate Definition

Table 3-11 Billing Rate Definition API Table

Area	Example
	Get all Billing rate tables by Group Name Returns light weight objects for rate tables with ID, Name, Description, Service Item type ID and Name. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/rategroupname/{groupName}
	Get all Billing rate tables by Group ID Returns light weight objects for rate tables with ID, Name, Description, Service Item type ID and Name. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/rategroupid/{groupID}

Table 3-11 Billing Rate Definition API Table

Area	Example
	Get rate table definition or schema details by name Returns rate table definition or schema details by name. Returns complete rate table definition with attributes and operations information. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/name/{name}
	Get rate table definition or schema details by ID Returns complete rate table definition with attributes and operations information. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/id/{id}
	Get rate table by wild card name search Returns light weight objects for rate tables with ID, Name, Description, Group ID and Name, Service Item type ID and Name. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema?name={wild card name}
	Create a rate table schema (One Rate Table Object only) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/create <rateTable id="" name=”Sample Rate Table” logicName=”SampleRateTable” description=”Sample Rate Table Description”> <rateGroup id=”123” name=”Sample Rate Group Name”/> <serviceItemType id=”123” name=”Sample Service Item Type Name”/> <billingAttributes> <attribute name="Attribute Name 1" billingField=”true” memoField=”false”/> <attribute name="Attribute Name 2" billingField =”false” memoField =”false”/> <attribute name="Attribute Name 3" billingField =”true” memoField =”false”/> </billingAttributes> <billingOperations> <operation name="Operation1" applicable=”false”/> <operation name="Operation2" applicable =”true”/> <operation name="Operation3" applicable =”true”/> </billingOperations> </rateTable>

Table 3-11 Billing Rate Definition API Table

Area	Example
	Update a rate table schema (One Rate Table Object only) Updates a billing rate table schema (One Rate table Object only). REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/update <rateTable id=”123" name=”Sample Rate Table” logicName=”BiSampleRateTable” description=”Sample Rate Table Description”> <rateGroup id=”123” name=”Sample Rate Group Name”/> <serviceItemType id=”123” name=”Sample Service Item Type Name”/> <billingAttributes> <attribute name="Attribute Name 1" billingField=”true” memoField=”false”/> <attribute name="Attribute Name 2" billingField =”false” memoField =”false”/> <attribute name="Attribute Name 3" billingField =”true” memoField =”false”/> </billingAttributes> <billingOperations> <operation name="Operation1" applicable=”false”/> <operation name="Operation2" applicable =”true”/> <operation name="Operation3" applicable =”true”/> </billingOperations> </rateTable
	Delete a rate table schema (One Rate Table Object only) REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/schema/delete <rateTable id=”123" name=”Sample Rate Table” logicName="BiSampleRateTable" />

Billing Rate Data

Table 3-12 Billing Rate Data API Table

Area	Examples
	Get rate table data by rate table name Returns billing rate table data by rate table name. REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/tablename/{tableName}
	Get rate table data by rate table ID Returns billing rate table data by rate table ID REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/tableid/{tableID}

Table 3-12 Billing Rate Data API Table

Area	Examples
	Update rate table data (One Rate table Object only) Wipes out the existing entries in the rate table data and inserts the given input data (works like Replace). REST URL: http://<ServerURL>/RequestCenter/nsapi/definition/billing/ratetables/data/update <rateTableData id=”123” name=”Sample Rate Table” logicName=”BiSampleRateTable”> <rateTableRecord rate=”123.45” rateCode=”RateCode1” unitOfMeasure=”Per Month”> <attributes> <attribute name="Attribute Name 1">Value1</attribute> <attribute name="Attribute Name 2">Value2</attribute> <attribute name="Attribute Name 3">Value3</attribute> </attributes> </rateTableRecord> <rateTableRecord rate=”456.78” rateCode=”RateCode2” unitOfMeasure=”Per Week”> <attributes> <attribute name="Attribute Name 11">Value11</attribute> <attribute name="Attribute Name 12">Value12</attribute> <attribute name="Attribute Name 13">Value13</attribute> </attributes> </rateTableRecord> </rateTableData

Table 3-12 Billing Rate Data API Table

Area	Examples
Filters	• Allowed parameters in the URL: Allowed parameters in the URL are startRow, recordSize, sortBy, sortDir, and responseType (xml or json) • For 'RateGroup' and 'RateTable', retrieval URLs: 'sortBy' and 'sortDir' query parameters can be used to specify on which column the sorting should be done and in which order. If the 'sortBy' is empty or invalid column name, the default sorting will be 'Ascending order' on 'ID'. • For 'RateTableData', retrieval URLs: Default sorting will be 'Descending order' on 'ID'. 'sortBy' and 'sortDir' query parameters are not supported. • Wild card search allows "starts-with" and prevents contains query and leading wild card searches for performance reasons. • For rate table schema create/update, the rate group and service item type goes by name. If ID is present, tries to cross check the ID, name combination and then perform the Update or Delete. • For rate group or rate table Update/Delete, go by ID if ID is present. Otherwise go by name. • In rate table schema retrieval, attributes will be retrieved only if it is selected as either Billing or Memo field. Operations will also be retrieved only if it is selected as applicable. • In rate table schema create, rate group name and service item type name are mandatory and their IDs are optional. If IDs are present, will cross check the ID, name combination before creating the rate table. • In rate table schema update, attributes or operations will be updated (select or unselect) only if they are provided in the input object. Otherwise, leave the existing ones unchanged. • For rate table data update, all existing records in rate table data will be deleted and the records from input XML/JSON will be saved as rate table data. • All validations from billing history portlet UI will be implemented for all Create/Update/Delete.
Sort Column(s)	Sorting is allowed on the two columns that are allowed in the URL, 'id' or 'name'.

Table 3-12 Billing Rate Data API Table

Area	Examples

Response XML	XML for Lightweght Rate Group list:
	`<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>`
	<rateGroups recordSize="2" startRow="1" totalCount="2">
	`<rateGroup id="11" name="Sample Rate Group 1" description="First group" />`
	`<rateGroup id="22" name="Sample Rate Group 2" description="Second group" />`
	`</rateGroups>`
	
	XML for single Rate Group Retrieval
	
	Same as Rate Group update request payload
	
	XML for Light weight Rate Table list retrieved by group ID or Name:
	`<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>`
	`<rateTables recordSize="2" startRow="1" totalCount="2">`
	`<rateTable id="12" name="Small scale Rate Table"`
	`logicName="BiSmallscaleRateTable" description="Rate table for small scale businesses">`
	`<serviceItemType id="6" name="Laptop"/>`
	`</rateTable>`
	`<rateTable id="22" name="Large scale Rate Table"`
	`logicName="BiLargescaleRateTable" description="Rate table for large scale businesses">`
	`<serviceItemType id="4" name="Desktop"/>`
	`</rateTable>`
	`</rateTables>`
	
	XML for single Rate Table schema Retrieval
	
	Same as Rate Group update request payload

	XML for Rate Table data Retrieval: Same as Rate Group update request payload

Order Management

Table 3-13 Order Management API Table

Area	Examples

GET API	To retrieve information of a requisition
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/reqinfo/reqid/{reqId}
	Sample output:
	[
	{
	"Status":"Cancelled",
	"OrgUnit":"<s ID=\"847\"/>",
	"CustomerPhone":null,
	"Customer":"Customer1",
	"Initiator":"Initiator1",
	"CustomerEmail":null,
	"CreatedDate":"5/26/16 7:46 AM",
	"ClosedDate":"05/26/2016",
	"ReqNo":"1",
	"SubmittedDate":"05/26/2016"
	}
	]

	To retrieve information of the services for a requisition
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/servicenfo/reqid/{reqId}
	Sample output:
	[
	{
	"Status":"<s ID=\"68\" />",
	"StdDuration":null,
	"UnitCost":"0.0",
	"Qty":"1",
	"Subtotal":"0.0",
	"Name":"TestsService",
	"ServiceLevelDesc":null
	},
	{
	"TotalCost":"0.0"
	}
	]

Table 3-13 Order Management API Table

Area	Examples

	To retrieve all the system and user comments for a requisition
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/comments/reqid/{reqId}
	[
	[
	{
	"CommentText":"Comment",
	"Name":"admin admin",
	"On":"Mon Jun 27 03:29:00 PDT 2016"
	}
	],
	[
	{
	"CommentText":"admin admin cancelled the requisition 20.",
	"Name":"admin admin",
	"On":"Tue Jun 14 09:04:49 PDT 2016"
	}
	]
	]

	To get details of all the attachments for a requisition
	REST URL:
	URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachmentslist/reqid/{reqId}
	Sample output:
	[
	{
	"UploadDate":"05/26/2016",
	"DocSize":"881304",
	"DocumentId":"1",
	"DocumentName":"file.txt"
	}
	]

	To download an attachment using the document id for a requisition REST URL: URL: http://<ServerURL>/RequestCenter/nsapi/definition/v1/ordermgmt/attachment/viewdocid/{document Id} Sample output: downloads the file

Table 3-13 Order Management API Table

Area	Examples

	Gets all the service Ids that the logged in user can order
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids
	Sample output:
	{
	"Clone Container": 1015,
	"Cassandra Cluster App (v2.1.1)": 1086,
	"Add Cloud Center User": 1081,
	"API_Service": 1091
	}

	Gets all the services that the logged in user can order on behalf of the customer whose login id is given
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/definition/v1/orderableserviceids/customerlogin/{customerlogin}
	Sample Output:
	{
	"1Service:: 2
	}

Policies

Area	Examples
CoreAPI	Get policies by service item type. Returns policies by Service Item Type Name. REST URL: /RequestCenter/nsapi/serviceitem/definition/{ serviceItemTypeName }/policies XML Response: <policies totalCount="2" startRow="1" recordSize="2"> <policy policyType="quota" accountName="bangalore" policyEnabled="true" executionOrder="2" dataTypeLogicName="SiLaptop" policyTemplateLogicName="quotacheck" policyId="34" name="Laptop Quota policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="quotacheck-serviceitemattribute" paramValue="Quota"/> <paramValue policyParamLogicName="quotacheck-quotathreshold" paramValue="90"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="orderservice" executionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="orderservice-servicename" actionParamValue="Apple iPhone"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-initiatorname" actionParamValue="admin"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-customername" actionParamValue="admin"/> <actionParamValue policyActionTemplateParamLogicName="orderservice-billtoOU" actionParamValue="Site Administration"/> <actionParamValue

Area	Examples
	policyActionTemplateParamLogic-Name="orderservice-sections"> <dictionary name="TabletComputer-Dict"> <fields/> </dictionary> </actionParamValue> </actionParamValues> </policyAction> </policyActions> </policy> <policy policyType="update" accountName="bangalore" policyEnabled="true" executionOrder="1" dataTypeLogicName="SiLaptop" policyTemplateLogic-Name="updatecheckchange" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-ser-viceitemattribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" execu-tionOrder="1"> <actionParamValues> <actionParamValue policyActionTemplatePa-ramLogicName="errormessage" actionParam-Value="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> </policies>
	Get policies by ServiceitemtypeName and PolicyID Returns polices by **serviceItemTypeName** and policy ID. REST URL: /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName }**/policies/id/{policyId}

Area	Examples
	Get policies by ServiceitemtypeName, PolicyName and PolciyType. Returns a polices by **serviceItemTypeName** , policy name, and policy type. **REST URL:** /RequestCenter/nsapi/serviceitem/definition/{ **serviceItemTypeName}**/policies/name/{name}?policyType={ capacity, quota, time} **Example:** http://<ServerURL>/RequestCenter/nsapi/serviceitem/definition/Laptop/policies/name/Laptop Policy?policyType=update **Note:** The allowed input values for "policyType" query parameter are: capacity/quota/time/update
XML Response:	<policy policyType="update" accountName="bangalore" policyEnabled="true" execution-Order="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheck-change" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitem-attribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOr-der="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy>
Error Response	<nsapi-error-response> No policy found for this name (Laptop ABCD Policy) and for policyType (update). </nsapi-error-response>

Area	Examples
POST	Create Policy for given ServiceItemTypename. POST URL: /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/create INPUT XML: <policy policyType="update" accountName="bangalore" policyEnabled="true" execution-Order="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheck-change" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitem-attribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOr-der="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> Response XML: <nsapi-response> <status-message code="POLICY_SUCCESS_003"> <value>Policy created successfully. </value> </status-message> </nsapi-response>

Area	Examples
PUT	**Update a Policy for given ServiceItemTypename**. **PUT URL:** /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/update INPUT XML: <policy policyType="update" accountName="bangalore" policyEnabled="true" execution-Order="1" dataTypeLogicName="SiLaptop" policyTemplateLogicName="updatecheck-change" policyId="33" name="Laptop Policy"> <description> </description> <policyParamValues> <paramValue policyParamLogicName="updatecheckchange-serviceitem-attribute" paramValue="Price"/> </policyParamValues> <policyActions> <policyAction policyActionTemplateName="policyalert" executionOr-der="1"> <actionParamValues> <actionParamValue policyActionTemplateParamLogicName="errormessage" actionParamValue="Policy alert when Price is updated for a Laptop"/> </actionParamValues> </policyAction> </policyActions> </policy> Response XML: <nsapi-response> <status-message code="POLICY_SUCCESS_003"> <value>Policy updated successfully. </value> </status-message> </nsapi-response>

Area	Examples
DELETE	**Delete a Policy for given ServiceItemTypename by PolicyId or PolicyName.** **DELETE URL:** /RequestCenter/nsapi/serviceitem/definition/**{serviceItemTypeName}**/policies/delete **Input XML:** To delete by policy name, the input should be: <policy policyType="quota" name="Laptop Quota policy Fourth"> </policy> To delete by policy ID, the input should be: <policy policyId="38"> </policy> **Response:** <nsapi-response> <status-message code="POLICY_SUCCESS_002"> <value>Policy deleted successfully. </value> </status-message> </nsapi-response>

Directory Data

Person

Table 3-14 Person API Table

Area	Examples
CoreAPI	**Get all people** Returns all people. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people Java Example: PersonList person = NSApiClient.getDirectory().getPeople(null);
	Get person by Id Returns the person with the specified Person Id. Nested entities (OUs, Groups, Roles, Addresses, Contact, and Preferences) are fetched. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people/id/<personId> Java Example: `Person persons = NSApiClient.getDirectory().getPersonById(<personId>);`

Table 3-14 Person API Table

Area	Examples
	Get person by LoginName Returns the person with the LoginName specified. Nested entities (OUs, Groups, Roles, Addresses, Contact, and Preferences) are fetched. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/people/loginname/<loginName> Java Example: `Person persons = NSApiClient.getDirectory().getPersonByLoginName("<loginName>");`

Table 3-14 Person API Table

Area	Examples
	Get logged-in user Returns the person who is currently logged in. REST URL: http://<serverURL>/RequestCenter/nsapi/directory/people/currentuser Java Example: `Person person = NSApiClient.getDirectory().getCurrentUser();`

	Get permissions for an object of a person
	Returns the permissions for particular object type for person using the objectId as query parameter.
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/people/id/{personID}/permissions?objectId={objectID}
	Note To obtain person ID use the get API http://<ServerURL>/RequestCenter/nsapi/directory/people and for object ID use the get API, http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids.
	Java Example:
	{
	"permissionList": {
	"startRow": 1,
	"recordSize": 5,
	"totalCount": 5,
	"permissions": [
	{
	"permissionID": 1948,
	"name": "test3 test",
	"type": "Person",
	"permission": "View Forms",
	"rbacRoleID": 102,
	"shadowObjectID": 2,
	"shadowObjectInstID": 5,
	"objectID": 310,
	"objectInstID": 4,
	"objectOperationID": 165,
	"direct": true
	},

Table 3-14 Person API Table

Area	Examples

	Gets assignment IDs for the specified people
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/people/id/{personID}/permissions
	Note To obtain person ID use the get API http://<ServerURL>/RequestCenter/nsapi/directory/people.
	Java Example:
	{
	"permissionList": {
	"startRow": 1,
	"recordSize": 4,
	"totalCount": 4,
	"permissions": [
	{
	"permissionID": 1860,
	"name": "test3 test",
	"type": "Person",
	"permission": "Design Services and change data in this group",
	"rbacRoleID": 102,
	"shadowObjectID": 2,
	"shadowObjectInstID": 5,
	"objectID": 19,
	"objectInstID": 3,
	"objectOperationID": 45,
	"direct": true
	}

Table 3-14 Person API Table

Area	Examples

	Gets the assignable roles (SOA roles and custom roles which has permission on any of SG/Services of SOA user)
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/people/assignableroles?type={roleType}
	Where, type=1 indicates platform roles and type=2 indicates service roles
	Java Example:
	`{`
	`  "roles": {`
	`  "startRow": 1,`
	`  "totalCount": 10,`
	`  "recordSize": 10,`
	`  "role": [`
	`  {`
	`  "roleId": 1235,`
	`  "roleName": "bypassrole",`
	`  "description": "",`
	`  "statusId": 1,`
	`  "roleTypeId": 4,`
	`  "inherited": false,`
	`  "parentId": 0,`
	`  "userCount": 1,`
	`  "entitlementCount": 5`
	`  },`
	…. .

Table 3-14 Person API Table

Area	Examples

	Gets the solution type of logged in user
	Based on the logged in user, the
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/people/solutiontypes
	Java Example:
	ADMIN USER
	{
	"List": [
	{
	"id": "2",
	"type": "Services"
	},
	{
	"id": "1",
	"type": "Platform"
	}
	]
	}
	Non-Admin User:
	{
	"List": [
	{
	"id": "2",
	"type": "Services"
	}
	]
	}

Table 3-14 Person API Table

Area	Examples

	Create a new role
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/v2/role
	Method: POST
	Sample Payload:
	{
	"roleDetials": {
	"roleName": "BTest",
	"parentRoleID": 0,
	"roleDesc": "New Role Desc"
	},
	"capabilities": [218, 219],
	"permissions": [
	{
	"objectId": 310,
	"objectInstId": 3,
	"operationId":165,
	},
	{
	"objectId": 310,
	"objectInstId": 4,
	"operationId":165,
	},
	{
	"objectId": 4,
	"objectInstId": 0,
	"operationId": 23,
	},
	],
	"objectTye": "People",
	"objectInstances": [5, 6, 7]
	}

Table 3-14 Person API Table

Area	Examples
	Gets the Sub Roles of a specified Role REST URL: http://<serverURL>/RequestCenter/nsapi/directory/subroles/id/{roleId} Method: GET

```
roles startRow="0" recordSize="0" totalCount="0">
    <role>
        <roleId>50</roleId>
        <roleName>customer</roleName>
        <description>Dictionary Participant - Customer</description>
        <statusId>0</statusId>
        <roleTypeId>5</roleTypeId>
        <logicName>customer</logicName>
        <inherited>false</inherited>
        <parentId>1067</parentId>
    </role>
    <role>
        <roleId>51</roleId>
        <roleName>st</roleName>
        <description>Dictionary Participant - Service Team</description>
        <statusId>0</statusId>
        <roleTypeId>5</roleTypeId>
        <logicName>st</logicName>
        <inherited>false</inherited>
        <parentId>1067</parentId>
    </role>
</roles>
```

Table 3-14 Person API Table

Area	Examples
	Gets capabilities of the specified role REST URL: http://<serverURL>/RequestCenter/nsapi/directory/subroles/id/{roleId}/capabilities Method: GET The query parameter *showInherited=true* displays the capabilities inherited from parent roles. **Sample Response:**

```
<roleCapabilityList startRow="0" recordSize="0" totalCount="0">
    <roleCapability>
        <systemCapabilityId>6</systemCapabilityId>
        <moduleId>6</moduleId>
        <name>Access Role Configuration</name>
        <inherited>false</inherited>
        <moduleName>Organization Designer</moduleName>
        <logicName>servicecommunity_role_config</logicName>
    </roleCapability>
    <roleCapability>
        <systemCapabilityId>109</systemCapabilityId>
        <moduleId>27</moduleId>
        <name>Access Service Roles</name>
        <inherited>false</inherited>
        <moduleName>User Management</moduleName>
        <logicName>serviceroles_access</logicName>
    </roleCapability>
</roleCapabilityList>
```

Table 3-14 Person API Table

Area	Examples

	Gets the list of assigned members of the specified role
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/v2/roles/id/{roleId}/members
	Method: GET
	Sample Response:
	`roleMemberList startRow="1" recordSize="2" totalCount="2">`
	`<roleMember>`
	`<assignmentId>28</assignmentId>`
	`<objectId>1</objectId>`
	`<objectInstId>4</objectInstId>`
	`<assignmentTypeId>2</assignmentTypeId>`
	`<memberName>MasterTenant_admin</memberName>`
	`<roleId>93</roleId>`
	`<memberType>Organizational Unit</memberType>`
	`<inherited>false</inherited>`
	`</roleMember>`
	`<roleMember>`
	`<assignmentId>27</assignmentId>`
	`<objectId>1</objectId>`
	`<objectInstId>5</objectInstId>`
	`<assignmentTypeId>2</assignmentTypeId>`
	`<memberName>ProviderBusinessTenant_admin</memberName>`
	`<roleId>93</roleId>`
	`<memberType>Organizational Unit</memberType>`
	`<inherited>false</inherited>`
	`</roleMember>`
	`</roleMemberList>`

Table 3-14 Person API Table

Area	Examples

	Edits the existing role
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/v2/roles/id/{roleId}/members
	Method: PUT
	Sample payload:
	{
	"roleDetials": {
	"roleName": "Arole1",
	"parentRoleID": 0,
	"roleDesc": " Updated",
	"roleID": 1325,
	"status":1
	},
	"addsubroles" : [1,2],
	"delsubroles" : [3,4],
	"addcap" : [7,19],
	"delcap" : [10, 20],
	"addper": [{
	"objectId": -10,
	"objectInstId": 0,
	"operationId": 205
	}],
	"delper": [{
	"objectId": -10,
	"objectInstId": 0,
	"operationId": 205
	}]
	"addmem": [{
	"memberType": 2,
	"objectInstances": [3]
	}],
	"delmem": [
	{
	"memberType": 1,
	"objectInstances": [1017]
	}]
	}

Table 3-14 Person API Table

Area	Examples
	Gets all the SOA users list for the specified connection REST URL: http://<serverURL>/RequestCenter/nsapi/ucsd/v1/soauserslist/shortname/{Connectioin ShortName} Method: GET
	Deletes SOA Role for the specified connection REST URL: http://<serverURL>/RequestCenter/nsapi/directory/v1/soarole/roleid/{roleID}/personId/{personID} Method: DELETE

	Assign or unassign people with roles if login name is known
	REST URL:
	http://<serverURL>/RequestCenter/nsapi/directory/people/{loginname}/roles
	Method: POST
	Sample Payload/body:
	{
	"rolemapping":
	{
	"operation":"assign",
	"roles":["My Role",”Tenant User”],
	}
	}
	Possible values of operation: “assign” or “unassign”
	Success response: 200
	Error response: 400 bad request

Table 3-14 Person API Table

Area	Examples
	Deactivates the person Hard deactivate and soft deactivate is possible. In Soft deactivate, the person is only marked inactive. In hard deactivate, • Deassociate all roles. • Deassociate all teams. • Make user details meaningless. URL: http://<serverURL>/nsapi/directory/user/{personID}/deactivate Method: POST **Note** To hard deactivate use filterParameter, isHardDeactivate=true. Sample Payload: {"status-message":{"code":"TA_170","value":"Successfully deactivated person"}}

Table 3-14 Person API Table

Area	Examples

Special Conditions	**Create/Update Person**
	Obtain person XML from the response of any of the GET person REST URLs mentioned above.
	POST person XML to the update person REST URL to create or modify a person.
	If the person exists (identified by login or personId in the XML), an update operation is performed; otherwise a new person is created.
	In the create operation, the following five elements are required:
	• firstName
	• lastName
	• homeOrganizationalUnitName
	• e-mail
	• login
	In the create operation, the password of the person is the set to the login name.
	Changes to the Home OU in update operation replace the Home OU of the Person. Other associated OUs, Groups, and Roles are ignored in both the create and update operations.
	For Home OU, TimeZone, Locale, Supervisor, Authorization Delegate, Login Module, Contact, and Address attributes, the following rules apply:
	• If an id element is sent in the XML but not found in the database, an exception with proper message is thrown.
	• If the id element is not found, the Name element sent in the XML is used instead. An exception is thrown if the name is not found in the database.
	• If neither id nor name element is in the XML and the attribute is optional, the attribute is ignored in the create/update operation (HomeOU is mandatory).
	• If the create/update operation fails due to incorrect or missing data sent in the XML, the HTTP status code returns “422 Unprocessable Entity”.
	POST REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/update
	Content Type:text/plain
	Java Example:
	`Person person = NSApiClient.getDirectory().getPersonById(<personId>);`
	`person.setLastName("<lastName>");`
	`Person persons = NSApiClient.getDirectory().updatePerson(person);`

Table 3-14 Person API Table

Area	Examples

Filters	**OU Name Filters**
	Search is case-sensitive and uses exact match .
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?ouname=<ouName>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("ouname", "<ouName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

	Group Name Filters
	Search is case-sensitive and uses exact match.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?groupname=<groupName>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();s`
	`paramsMap.put("groupname", "<groupName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

	Role Name Filters
	Search is case-sensitive and uses exact match.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people?rolename=<roleName>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("rolename", "<roleName>");`
	`PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);`

Sort Column(s)	First Name, Last Name, Login Name

XML Response	`<people totalCount="x" recordSize="y" startRow="z">`
	`  <person>`
	`  .  .  .`
	`  <contacts>`
	`  .  .  .`
	`  <contact>`
	`  <contactId></contactId>`
	`  <contactType></contactType>`
	`  <contactTypeId></contactTypeId>`
	`  <value></value>`
	`  </contact>`
	`  .  .  .`
	`  </contacts>`
	`  .  .  .`
	`  </person>`
	</people>

Organizational Unit

Table 3-15 Organizational Unit API Table

Area	Examples
CoreAPI	**Get all organizational units** Returns all organizational units. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits Java Example: `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnits(null);`
	Get organizational unit by Id Nested entities (suborganizational units) are fetched for getById and getByName only. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/<ouId> Java Example: `OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitById(ouId);`
	Get organizational unit by Name Nested entities (suborganizational units) are fetched for getById and getByName only. The Service Team OU is returned if two OUs with the same name but of different types are found. The optional parameter ?type=<BusinessUnit/serviceTeam> may also be specified ("all" is not allowed in the type parameter value). REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/name/<ouName> Java Example: `OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitByName("<ouName>");`
	Get OUs by Type Returns all organizational units of the OU type specified. Possible values: "all", "*BusinessUnit*", "*ServiceTeam*". REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?type=<ouType> Java Example: `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitByType("<ouType>");`

Table 3-15 Organizational Unit API Table

Area	Examples

	Create an Organization Unit of type business unit without assigning to any parent organization
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits
	Method: PUT
	Sample Payload/body:
	{
	"organizationalunit": {
	"description": "Details about Tail-f",
	"isBillable": "true",
	"organizationalUnitName": "Tail-f",
	"organizationalUnitType": "BusinessUnit",
	"status": "Active"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

	Create an Organization Unit of type service team without assigning to any parent organization
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits
	Method: PUT
	Sample Payload:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"isBillable": "false",
	"organizationalUnitName": "Tail-f",
	"organizationalUnitType": "ServiceTeam",
	"status": "Active"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

Table 3-15 Organizational Unit API Table

Area	Examples

	Create an Organization Unit of type business unit with inactive status
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits
	Method: PUT
	Sample Payload:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"isBillable": "true",
	"organizationalUnitName": "Ferrari",
	"organizationalUnitType": "BusinessUnit",
	"status": "Inactive"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

	Create an Organization Unit of type business unit with active status and associate it with parent Organization Unit
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits
	Method: PUT
	Sample Payload:
	{
	"organizationalunit": {
	"description": "Automobile company",
	"isBillable": "true",
	"organizationalUnitName": "Hero Honda",
	"organizationalUnitType": "BusinessUnit",
	"status": "Active",
	"parentId": "8",
	"parentName": "Honda"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

Table 3-15 Organizational Unit API Table

Area	Examples
	Update an Organization Unit and do not associate it with any parent Organization Unit Update operation allows you to update the description, OUName, OUType, and status REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/update Method: POST Content type: application/xml Sample Payload: <organizationalunit> <description>Update OU trhrough API</description> <organizationalUnitId>1088</organizationalUnitId> <isBillable>true</isBillable> <organizationalUnitName>Tailf_Updated</organizationalUnitName> <organizationalUnitType>Business Unit</organizationalUnitType> <status>Active</status> </organizationalunit> Success response: 200 created Error response: 400 bad request

Table 3-15 Organizational Unit API Table

Area	Examples
	Update an Organization Unit that is associated with parent Organization Unit. The update operation allows you to update the description, OUName, OUType, status, and parent organization REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/update Method: POST Content type: application/xml Sample Payload: <organizationalunit> <description>Update and change parent OU through API</description> <organizationalUnitId>1096</organizationalUnitId> <isBillable>true</isBillable> <organizationalUnitName>Hero Honda</organizationalUnitName> <organizationalUnitType>Business Unit</organizationalUnitType> <status>Active</status> <parentId>1088</parentId> <parentName>Tailf_Updated</parentName> </organizationalunit> Success response: 200 created

Table 3-15 Organizational Unit API Table

Area	Examples

	Returns assignment IDs for the specified OU
	Only the permissions configured in Service Designer is returned.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/{ouID}/permissions
	Method: GET
	Note To obtain the list of OUs, use the get API http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits.
	Sample Response:
	{
	"permissionList": {
	"startRow": 1,
	"recordSize": 1,
	"totalCount": 1,
	"permissions": [
	{
	"permissionID": 1733,
	"name": "OU1",
	"type": "Organizational Unit",
	"permission": "Read/Write",
	"rbacRoleID": 91,
	"shadowObjectID": 1,
	"shadowObjectInstID": 2,
	"objectID": 309,
	"objectInstID": 2,
	"objectOperationID": 293,
	"direct": true
	}
	]
	}
	}

Table 3-15 Organizational Unit API Table

Area	Examples

	Query for filtering the permissions for particular object type for OU
	Only the permissions configured in Service Designer is returned.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/{ouID}/permissions?objectId={objectID}
	METHOD: GET
	Note To obtain the list of OUs, use the get API http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits and for objectID, http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids.
	Sample Response:
	{
	"permissionList": {
	"startRow": 1,
	"recordSize": 1,
	"totalCount": 1,
	"permissions": [
	{
	"permissionID": 1733,
	"name": "OU1",
	"type": "Organizational Unit",
	"permission": "Read/Write",
	"rbacRoleID": 91,
	"shadowObjectID": 1,
	"shadowObjectInstID": 2,
	"objectID": 309,
	"objectInstID": 2,
	"objectOperationID": 293,
	"direct": true
	}
	]
	}
	}

Table 3-15 Organizational Unit API Table

Area	Examples

	Assign or unassign Organization Unit with roles if OU name is known
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/name/{OUname}/roles
	Method: POST
	Sample Payload:
	{
	"mapping":
	{
	"operation":"assign",
	"roles":["TestRole"],
	"type":"BusinessUnit"
	}
	}
	
	Possible values of operation: “assign” or “unassign”
	Success response: 200 created
	Error response: 400 bad request

	Assign or unassign Organization Unit with roles if OU ID is known
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/{OUID}/roles
	Method: POST
	Sample Payload:
	{
	"mapping":
	{
	"operation":"unassign",
	"roles":["TestRole"],
	"type":"ServiceTeam"
	}
	}
	Possible values of operation: "assign" / "unassign"

Table 3-15 Organizational Unit API Table

Area	Examples
Filters	**Organizational Unit Name Filters** Search is case-sensitive. StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored. Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed. The option parameter ?type=<businessUnit/serviceTeam> may also be specified with wildcard name search. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?name=<wildcard Value> Java Example: `OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitsByFilter("<wildcardValue>");`
Sort Column(s)	Organizational Unit Name
Response XML	`<organizationalunits totalCount="x" recordSize="y" startRow="z"> <organizationalunit> . . . </organizationalunit> </organizationalunits>`

Permissions

From Prime Service Catalog 12.1 release onwards, you can manage the permissions at every entity and object level from REST APIs. To ensure that you can correctly use of the APIs provided to manage these permissions, you must understand the following terms.

- **Entity** represents Organization Designer elements such as Teams, People, OU, and Role.
- **Object** represents service group, services, form group, active forms, dictionary, and dictionary group.
- **Instances** represents each record of the entity or object. An ID is auto assigned for every instance of the entity or object created in Prime Service Catalog.
- **Assignment ID** represents the IDs that is auto assigned for each permission granted on every instance of entity. These IDs are required when you wish to revoke permission using APIs.

Each of the entities and objects are assigned a hard-coded ID in Prime Service Catalog. However, we recommend that you run the API (http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids) which will list the latest IDs of all the entities and objects. This API is frequently used in conjunction with APIs that would need these ID as inputs to perform specific actions.

The table below lists the entities and objects in the system along with their hard-coded IDs.

Table 3-16 List of Entity IDs and Object IDs

Name	ID
Entity	
Organizational Unit	1
Person	2
Group	4
Queue	5
Object	
Reusable Form	309
Active Form Group	310
Service Item Group	–6
Standard Group	–7
Custom Content Group	–10
Service	13
Service Group	19
Portal Page Group	–1
Portal Page	–2
Portlet	–3
Service Item Definition	–8
Standard Definition	–9
Custom Content Definition	–11
Standard Table Data	–4
Service Item Instance Data	–5
Custom Content Data	–12
Billing	–13
Role	136
Dictionary Group	258
Dictionary	31

Workflow of assigning permissions on objects using nsAPI

Before you grant permission on an object you would need to identify a few input parameters. The below steps explains the workflow to assign permission on an object:

Step 1 Obtain the ID for the required entity or object on which you want to grant permissions. See section Obtaining Entity ID or Object ID, page 3-144.

Step 2 Obtain list of all available permissions on the selected object. See section Obtaining list of permissions on objects, page 3-147.

Step 3 Obtain the list of instance IDs for the selected object. See section Getting list of instance IDs, page 3-147.

Step 4 Grant Permission for the role. See section Granting Permission, page 3-148.

Workflow of Revoking permissions on objects using nsAPI

Step 1 Get Assignment ID for the granted permission. See section Getting Assignment ID, page 3-149.

Step 2 Revoke Permission. See section Revoking Permission, page 3-149.

Obtaining Entity ID or Object ID

To start with, first identify the ID of the object or entity on which you wish to grant permissions.

Execute the below API to get the list of all the objects or entities in the system. Note the ID number of the required entity or object from the result.

Fetches list of all entities

REST URL:

http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids

Method: GET

Sample Response:


```
{
        "List": [
                {
                        "id": 1,
                        "name": "Organizational Unit",
                        "logic_name": "ou"
                },
                {
                        "id": 2,
                        "name": "Person",
                        "logic_name": "person"
                },
                {
                        "id": 4,
                        "name": "Group",
                        "logic_name": "group"
                },
                {
                        "id": 5,
                        "name": "Queue",
                        "logic_name": "queue"
                },
                {
                        "id": 309,
```

```
                "name": "Reusable Form",
                "logic_name": "reusableform"
        },
        {
                "id": 310,
                "name": "Active Form Group",
                "logic_name": "formgroup"
        },
        {
                "id": -6,
                "name": "Service Item Group",
                "logic_name": "serviceitemgroup"
        },
        {
                "id": -7,
                "name": "Standard Group",
                "logic_name": "standardgroup"
        },
        {
                "id": -10,
                "name": "Custom Content Group",
                "logic_name": "customcontentgroup"
        },
        {
                "id": 13,
                "name": "Service",
                "logic_name": "service"
        },
        {
                "id": 19,
                "name": "Service Group",
                "logic_name": "servicegroup"
        },
        {
                "id": -1,
                "name": "Portal Page Group",
                "logic_name": "portalpagegroup"
        },
        {
                "id": -2,
                "name": "Portal Page",
                "logic_name": "portalpage"
        },
        {
                "id": -3,
                "name": "Portlet",
```

```
                "logic_name": "portlet"
        },
        {
                "id": -8,
                "name": "Service Item Definition",
                "logic_name": "serviceitemtype"
        },
        {
                "id": -9,
                "name": "Standard Definition",
                "logic_name": "standarddef"
        },
        {
                "id": -11,
                "name": "Custom Content Definition",
                "logic_name": "customcontentdef"
        },
        {
                "id": -4,
                "name": "Standard Table Data",
                "logic_name": "standardallinstancedata"
        },
        {
                "id": -5,
                "name": "Service Item Instance Data",
            "logic_name": "serviceitemtypeallinstancedata"
        },
        {
                "id": -12,
                "name": "Custom Content Data",
                "logic_name": "customcontent"
        },
        {
                "id": -13,
                "name": "Billing",
                "logic_name": "billing"
        },
        {
                "id": 136,
                "name": "Role",
                "logic_name": "role"
        },
        {
                "id": 258,
                "name": "Dictionary Group",
                "logic_name": "dictionarygroup"
```

```
                                    },
                                    {
                                                "id": 31,
                                                "name": "Dictionary",
                                                "logic_name": "dictionary"
                                    }
                        ]
}
```

Obtaining list of permissions on objects

Next, identify the permissions that are available on the chosen object. The below API displays the permission details of the selected object. Note the objectOperationId, logicName, and onAllObject from the result.

Fetches all ObjectOperationIds and details for the specified ObjectID

REST URL:

http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/id/{entityId}/permissions

Method: GET

Sample Response:

```
{
    "entityAssignablePermissionList":
    {
        "entityPermission":
        [
            {
                "objectId": 31,
                "objectOperationId": 46,
                "logicName": "dictionary_read",
                "displayName": "Read",
                "onAllObject": false
            }
```

Where,

- objectId is the ID for the object or entity, this ID is hard- coded by the system.
- objectOperationId is the ID for the specific permission within the scope of the object.
- logicName is the description of the permission.
- displayName is the permission. This can be read, write, or both.
- onAllObject indicates whether the permission is applicate on all objects or selected objects only

Getting list of instance IDs

Determine on which instance of the object the permission must be granted. The below API Fetches the instance ID for the specified entity ID or object ID. Note the objectInstId from the result.

Fetches the assignable instance for the given entity ID

REST URL:

http://<ServerURL>/RequestCenter/nsapi/directory/entity/{entityID}/assignableinstances

Method: GET

Sample Response:

```
{
  "List": [
    {
      "objectId": 13,
      "objectInstId": "16",
      "name": "ser"
    },
```

Where,

- objectId is the chosen object.
- objectInstId" is the instance ID of the object. This ID is auto generated as and when new instances are created in the system.
- Name is the name of the chosen object.

Granting Permission

Once you have all the required information, use the below API to grant permissions on the object or entity.

Grants permission for the specified Role ID to Service Group, Active Form Group, or DictionaryGroup

REST URL:

http://<ServerURL>/RequestCenter/nsapi/directory/roles/id/{roleID}/permissions/

Method: PUT

Sample Payload:

```
{
"assignmentInstance":
{
"operation":"assign",
"operationId":38,
"assignmentType":2,
"objectId":13,
"objectInstIds":[1]
}
}
```

Sample Response:

```
{
  "status-message": {
    "code": "TA_308",
    "value": "Permissions added successfully"
  }
}
```

Where,

- operation specifies whether the permission is granted or revoked
- operationId specifies the ID for the permission
- assignmentType specifies whether the permission is assigned for all instances (1) or for a specific instance (2)

- objectId specifies the Object ID or entity ID
- objectInstIds Specifies the ID for the instance of the object or entity.

Getting Assignment ID

When permission is granted to a role, each such instance called as assignmentInstance is assigned an ID. This ID is necessary in case you want to revoke the permission. Use the below Get API to obtain the assignmentInstance for which you wish to revoke permission. Note the assignmentIds from the result.

Fetches assignment ID of for a role instance

REST URL:

http://<ServerURL>//RequestCenter/nsapi/directory/roles/id/{roleID}/permissions/

Method: GET

Sample Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rolePermissionList startRow="1" recordSize="4" totalCount="4">
    <rolePermission>
        <assignmentId>1379</assignmentId>
        <name>Read</name>
        <logicName>ou_read</logicName>
        <description>Organizational Unit-ProviderBusinessTenant</description>
        <roleId>93</roleId>
        <objectId>1</objectId>
        <objectInstId>3</objectInstId>
        <assignmentTypeId>2</assignmentTypeId>
        <objectOperationId>5</objectOperationId>
        <inherited>false</inherited>
    </rolePermission>
```

Revoking Permission

Use the below API to revoke permission on the specified assignment ID.

Revokes permission for the specified Role ID to Service Group, Active Form Group, or DictionaryGroup

REST URL:

http://<ServerURL>/RequestCenter/nsapi/directory/roles/id/{roleID}/permissions/

Method: PUT

Sample Payload:

```
{
"assignmentInstance":
{
"operation":"unassign",
"assignmentIds":[1830]
}
}
```

Sample Response:

```
{
  "status-message": {
    "code": "TA_308",
    "value": "Permissions removed successfully "
  }
}
```

Where,

operation is unassign to revoke permissions.

assignmentIds is the assignment ID for which the permission is to be revoked.

Table 3-17 Permissions API Table

Area	Examples

	Fetches all ObjectOperationIds and details for the specified ObjectID
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/id/{entityId}/permissions
	Method: GET
	Sample Response:
	`{`
	`"entityAssignablePermissionList":`
	`{`
	`"entityPermission":`
	`[`
	`{`
	`"objectId": 31,`
	`"objectOperationId": 46,`
	`"logicName": "dictionary_read",`
	`"displayName": "Read",`
	`"onAllObject": false`
	`}`

Table 3-17 Permissions API Table

Area	Examples

CoreAPI	**Grants or revokes permissions to People/OU/Role**
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/entity/permissions
	Method: PUT
	Note To obtain the entity and object IDs use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids.
	Sample Payload:
	To grant permission:
	`{ "entityAssignmentInstance": { "operation":"assign", "operationId":39, "objectId":13, "assignmentType":2, "objectInstId":[2], "entityObjectId":136, -> (1- OU, 2-Person and 136 – for role) "entityInstanceIds":[99,100] (Accordingly provide the OU/People/Role instanceIds } }`
	To revoke permission:
	`{ "entityAssignmentInstance": { "operation":"unassign", "assignmentIds":[1830] } }`
	Sample Response:
	Grant Permission Response:
	`{ "status-message": { "code": "TA_308", "value": "Permissions added successfully" } }`
	Revoke Permission Response:
	`{ "status-message": { "code": "TA_308", "value": "Permissions removed successfully " } }`

Table 3-17 Permissions API Table

Area	Examples
	Fetches list of all the roles REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/roles Method: GET Sample Response: <roles startRow="1" recordSize="20" totalCount="124"> <role> <roleId>107</roleId> <roleName>0 Service Icons</roleName> <description>SOA Role(0 Service Icons)</description> <statusId>1</statusId> <roleTypeId>4</roleTypeId> <logicName>soarole_sg_2</logicName> <inherited>false</inherited> <parentId>0</parentId> </role> <role> <roleId>115</roleId> <roleName>1 Private Cloud IaaS Group</roleName> <description>SOA Role(1 Private Cloud IaaS Group)</description> <statusId>1</statusId> <roleTypeId>4</roleTypeId> <logicName>soarole_sg_3</logicName> <inherited>false</inherited> <parentId>0</parentId> </role>

	Clone the Role
	REST URL:
	http://<ServerURL>//RequestCenter/nsapi/directory/v2/role?cloneof={roleId}
	Method: POST
	Sample Payload:
	{
	"roleDetials": {
	"roleName": "CloneRole",
	"parentRoleID": 0,
	"roleDesc": "Clone Role Desc"
	}
	}

Table 3-17 Permissions API Table

Area	Examples
	Deletes the Role REST URL: http://<ServerURL>//RequestCenter/nsapi/directory/v2/role?roleId={roleId} Method: DELETE

Table 3-17 Permissions API Table

Area	Examples

	Grants/revokes permission for the specified Role ID to Service Group, Active Form Group, or DictionaryGroup
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/roles/id/{roleID}/permissions/
	Method: PUT
	Sample Payload:
	To grant permission:
	{
	"assignmentInstance":
	{
	"operation":"assign",
	"operationId":38,
	"assignmentType":2,
	"objectId":13,
	"objectInstIds":[1]
	}
	}
	To revoke permission:
	{
	"assignmentInstance":
	{
	"operation":"unassign",
	"assignmentIds":[1830]
	}
	}
	Sample Response:
	Grant Response:
	{
	"assignmentInstance":
	{
	"operation":"unassign",
	"assignmentIds":[1830]
	}
	}
	Revoke Response:
	{
	"status-message": {
	"code": "TA_308",
	"value": "Permissions removed successfully "
	}
	}

Table 3-17 Permissions API Table

Area	Examples

	Checks whether Role with specified name exists in the system
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/role/name/{name}/roleexists
	Method: GET
	Sample Payload:
	`If role Exists then Response`
	`{`
	`"exists": "true"`
	`}`
	`Role Not exist Response`
	`{`
	`"exists": "false"`
	`}`

	Fetches list of all entities
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids
	Method: GET
	Note To obtain the entity and object IDs use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids.
	Sample Response:
	`{`
	`"List": [`
	`{`
	`"id": 1,`
	`"name": "Organizational Unit",`
	`"logic_name": "ou"`
	`},`

	Fetches list of modules in Prime Service Catalog
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/module/ids
	Method: GET
	Sample Response:
	`{`
	`"List": [`
	`{`
	`"id": 7,`
	`"name": "Administration",`
	`"logic_name": "administration"`
	`},`

Table 3-17 Permissions API Table

Area	Examples
	Fetches capabilities of the logged in user for a specified module REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/modules/id/{moduleID}/capabilities Method: GET **Note** To obtain the module ID use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/module/ids. Sample Response: <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <roleCapabilityList startRow="1" recordSize="13" totalCount="13"> <roleCapability> <systemCapabilityId>7</systemCapabilityId> <moduleId>7</moduleId> <name>Manage Authorization Structure</name> <inherited>false</inherited> <logicName>administration_authorizationStructure</logicName> </roleCapability> <roleCapability> <systemCapabilityId>19</systemCapabilityId> <moduleId>7</moduleId> <name>Access Automation API</name> <inherited>false</inherited> <logicName>administration_manage_automation_api</logicName> </roleCapability>

Table 3-17 Permissions API Table

Area	Examples
	Get objects on which permissions can be applied REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/entity/objectId/{objectId}/operationId/{operationId}/objectinstances Method: GET **Note** To obtain the object IDs use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids and to obtain operationID use, http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/id/{entityID}/permissions. Sample Response: `{ "List": [{ "objectId": 19, "objectInstId": 88, "objectOperationId": 210, "name": "Service_Automation12-5922" }, { "objectId": 19, "objectInstId": 89, "objectOperationId": 210, "name": "Service_Automation12-7309" }, { "objectId": 19, "objectInstId": 90, "objectOperationId": 210, "name": "Service_Automation12-1258" },`

Table 3-17 Permissions API Table

Area	Examples
	Fetches the assignable instance for the given object ID REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/entity/{objectID}/assignableinstances **Note** To obtain the entity and object IDs use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids. Method: GET Sample Response: `{ "List": [ { "objectId": 13, "objectInstId": "16", "name": "ser" },`
	Fetches the available instance for the given object ID REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/entity/{objectID}/availableinstances Method: GET **Note** To obtain the entity and object IDs use the GET API http://<ServerURL>/RequestCenter/nsapi/directory/entitytype/ids. Sample Response: `{ "List": [ { "objectId": 13, "objectInstId": "27", "name": "Power Off App VM", },`

Table 3-17 Permissions API Table

Area	Examples
	Fetches the list of assignment types REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/entity/assignmenttypes Method: GET Where, assignment type ID= 2 indicates *instance* assignment type ID= 1 indicates *Constrained* assignment type ID= 0 indicates *All Objects* Sample Response: `{ "List": [ { "assignmentType": "0", "name": "All", "logicName": "asstypeall" }, { "assignmentType": "1", "name": "Constrained", "logicName": "asstypeconstrained" }, { "assignmentType": "2", "name": "Instance", "logicName": "asstypeinstance" } ] }`

Table 3-17 Permissions API Table

Area	Examples

	Assigns or unassigns roles to multiple people, OU, or group
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/entity/roles
	Method: PUT
	Sample Payload:
	`{`
	`"entityAssignmentInstance":`
	`{`
	`"operationActionId":"1", or (2)`
	`"entityObjectId":4,`
	`"entityInstanceIds":[163,164],`
	`"roleIds":[2341,3423]`
	`}`
	`}`
	Sample Response:
	`{`
	`{`
	`"status-message": {`
	`"code": "TA_312",`
	`"value": "Roles added successfully"`
	`}`
	`}`
	
	`{`
	`"status-message": {`
	`"code": "TA_312",`
	`"value": "Roles removed successfully "`
	`}`
	`},`

Table 3-17 Permissions API Table

Area	Examples
	Fetches the operation action IDs REST URL: http://<serverURL>/RequestCenter/nsapi/directory/entity/operationactionids Method: GET Java Example: `{ "List":[ { "OperationActionId":"assign", "Value":"1" }, { "OperationActionId":"unassign", "Value":"2" } ] }`

Groups

Table 3-18 Groups API Table

Area	Examples
CoreAPI	**Get all groups** Returns all groups. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups Java Example: `GroupList groups = NSApiClient.getDirectory().getGroups(null);`
	Get group by Id Nested entities (subgroups) are fetched for getById and getByName only. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups/id/<groupId> Java Example: `Group groups = NSApiClient.getDirectory().getGroupsById(<groupId>);`

Table 3-18 Groups API Table

Area	Examples
	Get group by Name Nested entities (subgroups) are fetched for getById and getByName only. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups/name/<groupName> Java Example: `Group groups = NSApiClient.getDirectory().getGroupsByName("<groupName>");`

	Create a group
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups"
	Method: PUT
	Sample Payload/body:
	{
	"group":
	{
	"groupName": "TestGroup_API", "description": "Test Group", "status": "Active"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

	Create a group and associate with parent id
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups
	Method: PUT
	Sample Payload:
	{
	"group":
	{
	"groupName": "TestP", "description": "Test Group", "status": "Active", "parentId": "9", "parentName": "TestGroup"
	}
	}
	Success response: 201 created
	Error response: 400 bad request

Table 3-18 Groups API Table

Area	Examples

	Update a group using group id - update name, description, status
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups
	Method: POST
	Sample Payload:
	{
	"group":
	{
	"groupId":"20", "groupName": "TestP_upd", "description": "Test Group update", "status": "Active"
	}
	}
	Success response: 200 OK
	Error response: 400 bad request

	Update a group using group id and associate with new parent group - update name, description, status and associate with parent group
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups
	Method: POST
	Sample Payload:
	{
	"group":
	{
	"groupId":"20", "groupName": "NagTestP_updat", "description": "Test Group update once again", "status": "Active", "parentId": "21", "parentName": "NagTestP_Update"
	}
	}
	Success response: 200 OK
	Error response: 400 bad request

Table 3-18 Groups API Table

Area	Examples

	Assign or unassign group with roles
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/groups/{groupname}/roles
	Method: POST
	Sample Payload:
	{
	"mapping":
	{
	"operation":"assign",
	"roles":["TestRole"]
	}
	}
	Possible values of operation: “assign” or “unassign”
	Success response: 200 created
	Error response: 400 bad request

Filters	**Group Name Filters** Search is case-sensitive. StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored. Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/groups?name=<wild cardValue> Java Example: `GroupList groups = NSApiClient.getDirectory().getGroupsByFilter("<wildcardValue>");`
Sort Column(s)	GroupName

Response XML	`<groups totalCount="x" startRow="y" recordSize="z">`
	`  <group>`
	`  . . .`
	`  </group>`
	`<groups>`

Accounts

Table 3-19 Accounts API Table

Area	Examples
Core API	Get all Accounts Returns all accounts. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts
	Get account by ID Returns account by ID. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/id/{id}
	Get account by name Returns account by name. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/name/{name}
	Get account by name with wild card Returns account for search. REST URL (HTTP POST): http://<ServerURL>/RequestCenter/nsapi/directory/accounts?name={wild card name}

Table 3-19 Accounts API Table

Area	Examples

Other Usage Scenarios	Create or Update an account (One Account Object only).
	REST URL(HTTP POST):
	http://<ServerURL>/RequestCenter/nsapi/directory/accounts/update
	`<account>`
	`<name>Sample Account Name</name>`
	`<accountId>123</accountId>`
	`<accountType>Project Account</accountType>`
	`<description>Sample Account Description</description>`
	`<billingRateGroup>Sample Rate Group Name</billingRateGroup>`
	`<organizationalUnits>`
	`<organizationalUnit>`
	`<organizationalUnitName>OU Name 1</organizationalUnitName>`
	`<organizationalUnitType>Service Team</organizationalUnitType>`
	`</organizationalUnit>`
	`<organizationalUnit>`
	`<organizationalUnitName>OU Name 2</organizationalUnitName>`
	`<organizationalUnitType>Service Team</organizationalUnitType>`
	`</organizationalUnit>`
	`</organizationalUnits>`
	`<functionalPositions>`
	`<functionalPosition>`
	`<functionalPositionName>Func Pos 1</functionalPositionName>`
	`<ownerLoginName>assigneeLoginName</ownerLoginName>`
	`</functionalPosition>`
	`<functionalPosition>`
	`<functionalPositionName>Func Pos 2</functionalPositionName>`
	`<ownerLoginName />`
	`</functionalPosition>`
	`</functionalPositions>`
	`<customAttribute name="IntegerValue" />`
	`<customAttribute name="LongString">This is Long String</customAttribute>`
	`<customAttribute name="MediumString" />`
	`<customAttribute name="ShortName" displayName="ShortName" value="NewValue"/>`
	`<customAttribute name="ShortString">This is short value</customAttribute>`
	`<customAttribute name="MoneyValue">1234567.89</customAttribute>`
	`<customAttribute name="DoubleValue">12345678</customAttribute>`
	`<customAttribute name="DateValue">31-Jan-2013</customAttribute>`
	`</account>`

	Delete Account (One Account Object only). REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/directory/accounts/delete <account> <name>Sample Account Name</name> <accountId>123</accountId> </account>

Table 3-19 Accounts API Table

Area	Examples
	Get OU Name and Type of OU Returns OU Name, Type of OU, which are not associated to tenant account (Which login user has access to the OUs. RBAC check is in place). REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/tenantorgunitlist
	Get Functional Positions for accounts Returns functional positions for accounts. REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/accounts/functionalpositionlist
Filters	Supports no filters.
Sort Column(s)	Sorting is not allowed.
Response XML	<accounts totalCount="x" startRow="y" recordSize="z"> <account> . . .(same as the request payload for update) </account> </accounts>

Transactional Data

Requisitions

Table 3-20 Requisitions API Table

Area	Examples
CoreAPI	**Get requisitions for the current user** Get requisitions with the default filter, that is: ViewName = Ordered for Self Status = Ongoing REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions Java Example: `RequisitionList requisitions = NSApiClient.getTransaction().getRequisitions(null);`
	Get requisition by Id RBAC checking is applied against logged-in user. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitionId> Java Example: `Requisition requisitions = NSApiClient.getTransaction().getRequisitionsById(<requisitionId>);`
	Retrieve Service Form data Retrieves service form data of all the requisition entries of the of the specified requisition ID. The response is a list containing the service name, form field data, and requisitionEntryId for each requisition. The response can be obtained in XML and JSON format. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitioin_id>/requisitiondata Java Example: `RequisitionEntriesList requisitionEntries = NSApiClient.getTransaction().getRequisitionData(id)`

Table 3-20 Requisitions API Table

Area	Examples
	Submit an existing unsubmitted requisition/cart without adding service(s) to it Method: POST REST URL: /RequestCenter/nsapi/transaction/requisitions Request Payload/Body is not needed Success Code: 201 Response Error Code: Refer to REST/Web Services Error Messages table

Table 3-20 Requisitions API Table

Area	Examples

	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

Table 3-20 Requisitions API Table

Area	Examples

	Add order to an existing unsubmitted requisition/cart
	Method: PUT
	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: 200
	Response Error Code: Refer to REST/Web Services Error Messages table

Table 3-20 Requisitions API Table

Area	Examples
	Cancel an unsubmitted requisition/cart Method: DELETE REST URL: /RequestCenter/nsapi/transaction/requisitions Payload is not needed Success Code: 200 Response Error Code: Refer to REST/Web Services Error Messages table
	Remove/Cancel a specific service from a requisition/cart Method: DELETE REST URL: /RequestCenter/nsapi/transaction/requisitionentries/{reqentry_id} Payload is not needed Success Code: 200 Response Error Code: Refer to REST/Web Services Error Messages table
	Cancel the monitor plan of an open order The cancelMonitorPlan option of cancel requisition nsAPI cancels the monitor plan of an open order. The status of the requisition is set to Delivery Canceled. Method: Delete REST URL: /RequestCenter/nsapi/transaction/requisition/{requisitionID}?cancelMonitorPlan=true Payload is not needed Success Code: 200 Response Error Code: Refer to REST/Web Services Error Messages table **Note** Only a user with Site Administration role has the permission to execute the cancelMonitorPlan option.

Table 3-20 Requisitions API Table

Area	Examples
	Update the Service Request form Updates the form data based on the inputs specified in nsXML POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/updateServiceRequest Sample Payload: <?xml version="1.0" encoding="UTF-8"?> <message channel-id="" is-autocomplete="false" is-published="true"> <task-started task-type="task"> <requisition> <requisition-entry> <data-values> <data-value multi-valued="false"> <name>GridUpdate-1.Name</name> <value>ordernamechanged4</value> </data-value> <data-value multi-valued="false"> <name>GridUpdate-1.Roll</name> <value>orderrollchanged4</value> </data-value> </data-values> <requisition-entry-id>55</requisition-entry-id> </requisition-entry> </requisition> </task-started> </message> Sample Response: Update service request is initiated successfully.

Table 3-20 Requisitions API Table

Area	Examples

Filters	**View and Status Filters**
	The views and statuses available for filtering correspond to those in the My Services Requisitions tabs.
	If Status is not specified, "Ongoing" is used.
	If ViewName is not specified, "Ordered for Self" is used.
	Possible values for ViewName:
	• Ordered for Self – Requisitions for the current user.
	• Ordered for Others – Requisitions submitted by the current user for others (via Order on Behalf).
	• Ordered for my unit – Requisitions for people in the OUs the current user belongs to (the view returns data for other people only if the user has the "See Requisitions for My Business Units" capability).
	Possible values for Status: Ongoing, Preparation, Ordered, Closed, Canceled, Rejected, All.
	If an incorrect value is given, the default values ("Ongoing" and "Ordered for Self") are used.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/ViewName=<viewName>[|AND|Status=<status>]
	Java Example:
	`String filterString = "ViewName=<viewName>|AND|Status=<status>";`
	`RequisitionList requisitons =`
	`NSApiClient.getTransaction().getRequisitionsByFilter(filterString);`
	Note Filters are not supported in PUT and POST methods.

Sort Column(s)	Customer Name, Owner (Initiator) Name, Requisition ID, Service Name, Started Date, Status, Submit Date **Note** Sorting is not supported in PUT and POST methods.

Response XML for GET	`<requisitions totalCount="x" recordSize="y" startRow="z">`
	`<requisition>`
	`. . .`
	`</requisition>`
	`</requisitions>`

Table 3-20 Requisitions API Table

Area	Examples

Response XML for PUT and POST	{ "RequisitionSubmit": {
	"requisitionId": 536,
	"customer": "admin admin",
	"initiator": "admin admin",
	"startedDate": "…",
	"startedDateRaw": …,
	"dueDate": "…"
	"dueDateRaw": …,
	"status": "Ordered",
	"links" : [
	{ "name" : "RequisitionStatus",
	"href" : "http://…"
	},
	{ "name" : "ServiceItems",
	"href" : "http://…"
	}
	]
	}
	}
	Note dueDate and dueDateRaw attributes are available only when asynchronous submission setting is turned off in Administration > Settings module.

Table 3-20 Requisitions API Table

Area	Examples

REST API	**Get Service Form details**
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/v1/orderform?serviceids=<list of service IDs, comma seperated>
	Sample Output:
	{
	"API_Service": {
	"serviceID": "12",
	"quantity": 1,
	"API_AFC-API_Dictionary": {
	"Name": {
	"value": ""
	},
	"City": {
	"value": [],
	"isMultiValued": "true",
	"selectableValues": [
	"Bangalore",
	"Mysore"
	]
	},
	"Range": {
	"value": " ",
	"selectableValues": [
	"100-1000",
	"1000-2000"
	]
	},
	"Sal": {
	"value": [],
	"isMultiValued": "true",
	"selectableValues": [
	"10k",
	"20k",
	"30k",
	"40k"
	]
	}
	},
	"Form-Dictionary": {
	"Select_Person": {
	"value": ""
	},
	"Login_ID": {
	"value": ""
	}
	}
	},

Table 3-20 Requisitions API Table

Area	Examples

	<pre>"0Service": {
	"serviceID": "1",
	"quantity": 1,
	"maxQuantity": 2,
	"Form-Dictionary": {
	"Select_Person": {
	"value": ""
	},
	"Login_ID": {
	"value": ""
	}
	}
	}
	}</pre> Sample output when the form contains a grid dictionary: <pre>{
	"Grid_Service": {
	"serviceID": "114",
	"quantity": 1,
	"grid_form-Grid_dict": {
	"isGrid": true,
	"row-1": {
	"Name": {
	"value": ""
	},
	"Description": {
	"value": ""
	}
	}
	}
	}
	}</pre> **Note** If a field is multi valued the value of the field is seen as a list and is Multivalued flag is seen for the field.
	
	Note If a field is mandatory then the is Mandatory flag is seen for the field.

Table 3-20 Requisitions API Table

Area	Examples

	Sample Input when the Form contains grid dictionaries:
	`{`
	`"Grid_Service": {`
	`"serviceID": "114",`
	`"quantity": 1,`
	`"grid_form-Grid_dict": {`
	`"row-1": {`
	`"Name": {`
	`"value": "Name1"`
	`},`
	`"Description": {`
	`"value": "desc1"`
	`}`
	`},`
	`"row-2": {`
	`"Name": {`
	`"value": "Name2"`
	`},`
	`"Description": {`
	`"value": "desc2”`
	`}`
	`}`
	`}`
	`}`
	`}`
	Sample Output:
	`{`
	`"Requisition ID":6,`
	`"InitiatorLogin":"admin admin",`
	`"customerLogin":"user1 user1",`
	`"startedDate":"Wed Oct 26 12:31:13 IST 2016",`
	`"status":"Ordered"`
	`}`
	Note The payload for the submit requisition APIs can be obtained from the service form details API.
	Note If the grid dictionary is person based then the login ID must be sent in the place of first name.
	Note Only form rules with server side events (PostSubmit) are executed.

	Submits requisition of a transaction Method: POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/v1/requisition

Requisitions Entries

Table 3-21 Requisitions Entries API Table

Area	Examples
CoreAPI	Get Requisition Entry By ID RBAC checking is applied against the current user. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/id/<reqEntryId> Java Example: `RequisitionEntry requisitonEntry = NSApiClient.getTransaction().getRequisitionEntryById(<reqEntryId>);`

	Get Requisition Entries By Requisition ID
	RBAC checking is applied against the current user.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/RequisitionNumber=<reqId>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("startRow", "1"); //optional`
	`paramsMap.put("recordSize","10"); //optional`
	`String filterString = "RequisitionNumber=" + <reqId>;`
	`RequisitonEntryList requisitonEntries = NSApiClient.getTransaction().getRequisitionEntries(paramsMap, filterString);`

Filters	**ServiceName** Displays all ordered services that contains the filter values. Service Name filter value is test. Lists all ordered services where the service name contains "test". REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/ViewName=<viewName>|AND|ServiceName=<wildcard>|AND|Status=<status> Java Example: **isNgcRequest** Displays service request process complete percentage if the isNgcRequest filter value is false or not mentioned. If the isNgcRequest filter value is true, the service process request percentage is not displayed on the Order form page. REST URL http://localhost:8088/RequestCenter/nsapi/transaction/requisitionentries/ViewName=Ordered%20for%20Myself|AND|ServiceName=test|AND|Status=AllOpenExceptPreparation?responseType=json&isNgcRequest=true&recCount=20&sortBy=submitDate&sortDir=desc
Sort Column(s)	Due On, Requisition Entry ID

Response XML	`<requisitionEntries totalCount="x" recordSize="y" startRow="z">`
	`  <requisitionEntry>`
	`  . . .`
	`  </requisitionEntry>`
	`</requisitionEntries>`

Table 3-22 DELETE Requisitions Entries API Table

Area	Examples
CoreAPI	Cancel Services from a Requisition or Cart • Removes a service if status of requisition or cart is un-submitted. • Cancels a service if status of requisition or cart is submitted. DELETE REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/<reqEntryId> **Note** You replace the <reqEntryId> with the requisition ID. Consider http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/175, the response is as provided below.
Filters	Not applicable
Sort Column(s)	Not applicable

Response XML	{
	"requisition id": 175
	}

Authorizations

Table 3-23 Authorizations API Table

Area	Examples
CoreAPI	**Get authorizations for the current user** Gets authorizations with the default filter, that is: ViewName = Authorizations for Self Status = Ongoing REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations Java Example: `AuthorizationList authorizations = NSApiClient.getTransaction().getAuthorizations(null);`
	Get authorization by Id RBAC checking is applied against the current user. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/id/<taskId> Java Example: `Authorization authorizations = NSApiClient.getTransaction().getAuthorizationsById(<taskId>);`

	Get related authorization tasks (approval chain) by ID
	RBAC checking is applied against the current user.
	If the ID passed is for an authorization task at the requisition level (departmental authorization, departmental review, financial authorization), all requisition-level authorization tasks of that particular requisition are returned.
	If the ID passed is for an authorization task at the requisition entry level (service group authorization, service group review), all requisition entry-level authorization tasks of that particular requisition entry are returned.
	The default sorting is in descending order of Due On.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations?taskId=<taskId>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("startRow", "" + 1);  //optional`
	`paramsMap.put("recordSize", "" +10);  //optional`
	`paramsMap.put("sortBy", "dueOn");  //optional`
	`paramsMap.put("sortDir", "asc"); //optional`
	`paramsMap.put("taskId", "<taskId>");`
	`Authorization authorizations = NSApiClient.getTransaction().getAuthorizations (paramsMap);`

Table 3-23 Authorizations API Table

Area	Examples

Filters	**View and Status Filters**
	The views and statuses available for filtering correspond to those in the My Services Authorizations tabs.
	If Status is not specified, it is set to "Ongoing".
	If ViewName is not specified, it is set to "Authorizations for Self".
	Possible values for ViewName:
	• Authorizations for Self – Authorizations assigned to the current user.
	• Assigned and Unassigned Authorizations for Self – Authorizations assigned to the current user or the queues which the user has access to.
	• Authorizations for Others – Authorizations assigned to people in the OUs the current user belongs to (the view returns data only if the user has the "See Authorizations for My Business Units" capability).
	Possible values for Status: Ongoing, Approved, Rejected, Canceled, Reviewed, All.
	If an incorrect value is given the default values ("Ongoing" and "Authorizations for Self") are used.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/ViewName=<viewName>[|AND|Status=<status>]
	Java Example:
	`String filterString = "ViewName=<viewName>|AND|Status=<status>";`
	`AuthorizationList authorizations = NSApiClient.getTransaction().getAuthorizationsWithFilters(filterString);`

Sort Column(s)	Customer Name, Due On, Performer Name, Priority, Requisition ID

Response XML	`<authorizations totalCount="x" recordSize="y" startRow="z" />`
	`<authorization>`
	`. . .`
	`</authorization>`
	`</authorizations>`

Tasks

Table 3-24 Tasks API Table

Area	Examples
CoreAPI	**Get tasks for in the current user** Gets tasks with the default filter, that is: ViewName = AvailableWork REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks Java Example: `TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(null);`
	Get task by ID RBAC checking is applied against the current user. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/id/<taskId> Java Example: `TaskFull tasks = NSApiClient.getTransaction().getDeliveryTaskById(<taskId>);`

Table 3-24 Tasks API Table

Area	Examples

	Get tasks by Requisition Entry ID
	RBAC checking is applied against the current user.
	The default sorting is in ascending order of Activity ID.
	Parameters and possible values:
	• showSkippedTasks: false (default), true
	• taskType: all (default), delivery, authorization
	• showNestedTasks: false, true (default)
	When the parameter is set to true, delivery tasks are returned with the nested parent-child hierarchy maintained in the XML structure.
	• showChildDeliveryPlan: false (default), true
	This applies to bundle services only. When the parameter is set to true, the delivery tasks for all included services are also returned.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=<reqEntryId>?taskType=<taskType>&showSkippedTasks=<false|true>&showNestedTasks=<false|true>&showChildDeliveryPlan=<false|true>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("taskType", "delivery"); //optional`
	`paramsMap.put("showSkippedTasks", "true"); //optional`
	`paramsMap.put("showNestedTasks", "true"); //optional`
	`paramsMap.put("sortBy", "dueOn"); //optional`
	`paramsMap.put("sortDir", "desc"); //optional`
	`String filterString = "RequisitionEntryNumber=" + <reqEntryId>;`
	`TaskList tasks = NSApiClient.getTransaction().getAuthAndDeliveryTasks (paramsMap, filterString);`

	Get milestones by Requisition ID
	RBAC checking is applied against the current user.
	Delivery process milestones (reviews, authorizations, delivery projects) are returned in the chronological order. No sorting and paging is supported.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionNumber=<reqId>
	Java Example:
	`String filterString = "RequisitionNumber=" + <reqId>;`
	`MilestoneList milestones = NSApiClient.getTransaction().getDeliveryProcessForMilestone (filterString);`

Table 3-24 Tasks API Table

Area	Examples
Special Conditions	**Approve Tasks** Perform an HTTP POST with the action and task ID in the REST URL. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks /<taskId>/approve Java Example: `TaskAction approve = NSApiClient.getTransaction().approveTask(<taskId>);`
	Reject Tasks Perform an HTTP POST with the action and task ID in the REST URL. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/reject Java Example: `TaskAction Reject = NSApiClient.getTransaction().rejectTask(<taskId>);`
	Complete Tasks Perform an HTTP POST with the action and task ID in the REST URL. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/done Java Example: `TaskAction Complete = NSApiClient.getTransaction().completeTask(<taskId>);`
	Review Tasks Perform an HTTP POST with the action and task ID in the REST URL. POST REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/review Java Example: `TaskAction Review = NSApiClient.getTransaction().reviewTask(<taskId>);`

Table 3-24 Tasks API Table

Area	Examples

Filters	**View Filters**
	The views available for filtering correspond to those in the Service Manager module. User-defined views may not be used.
	If the ViewName filter is not specified, it is set to AvailableWork.
	Possible values for ViewName: AvailableWork, MyWork, MyLateWork, WorkForeCast
	If an incorrect value is given the default value is used.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks?viewName=<viewName>
	Java Example:
	`MultiValueMap paramsMap = new MultiValueMap();`
	`paramsMap.put("ViewName", "<viewName>");`
	`TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(paramsMap);`
	• You can view all Approval Tasks at ***Requisition Entry level*** (History, including future Approval Tasks).
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=111?taskType=authorization&Status=All
	• You can view all Approval Tasks at ***Requisition level*** (History, including future Approval Tasks).
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionNumber=111?taskType=authorization&Status=All

Sort Column(s)	Activity ID, Completed On, Customer Name, Customer OU Name, Due On, Effort, Initiator Name, Performer Name, Priority, Requisition ID, Scheduled Start Date, Service Name, Task Name, Task Type

Response XML	`<tasks totalCount="x" recordSize="y" startRow="z">`
	`  <task>`
	`  . . .`
	`  </task>`
	`</tasks>`

Policy Alert

Table 3-25 Policy Alert API Table

Area	Examples
Core API	Get Policy Alert by CreatedDate Returns policy alert by Created Date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName}{Operator}{Value}
	Get Policy Alert by Severity Type and CreatedDate Returns policy alert by Severity Type and Created Date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	Get Policy Alert by Service Item Type Name and CreatedDate Returns policy alert by Service Item Type Name and Created Date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	Get Policy Alert by Account Name and CreatedDate Returns policy alert by Account Name and Created Date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	Delete Policy Alert Deletes policy alert is by UniqueID of Policy alert entry. Unique ID is the system generated ID unique for each policy alert entry. REST URL (HTTP DELETE): http://<ServerURL>/RequestCenter/nsapi/transaction/policyalert/delete/uniqueid/{uniqueIDs}
Filters	Supports up to four filters. Comparison Operators: String columns: equals, starts-with operators (contains and like operators are not allowed). Relational Operators: AND (OR is not allowed) Separator = | Allowed Columns: ServiceItemTypeName, AccountName, SeverityType, and CreatedDate Allowed parameters in the URL: startRow, recordSize, sortBy, sortDir and responseType (xml or json)

Table 3-25 Policy Alert API Table

Area	Examples
Sort Column(s)	Sorting is not allowed.
Response XML	<policyAlert recordSize="2" startRow="1" totalCount="1"> <policyAlertRecord id="123" severityLevel="Info" alertContext="Policy Alert Action" requisitionID="12" requisitionEntryID="32"> <uniqueID>e96c062f-fe50-4b77-981a-dda1ab078808</uniqueID> <policy id="8" name="Quota policy 8" /> <serviceItemType id="82" name= "Laptop with Quota"/> <serviceItem id="12" name="Service Item one" /> <account id="1" name="Custom Account 1" /> <agreement id="6" name="Agreement Name"/> <message>Message Message Message Message</message> <createdDate local="04/09/2013 12:44 PM" raw="2013-04-09T19:44:28.720-07:00" /> </policyAlertRecord> </policyAlert>

Billing History

Table 3-26 Billing History API Table

Area	Examples
Core API	**Get billing history by Transaction Date** Returns the billing history by Transaction Date. REST URL: http://<ServerURL>:8088/RequestCenter/nsapi/transaction/billinghistory/{columnName}{Operator}{Value}
	Get billing history by Service Item Name, Service Item Type Name, and Transaction Date Returns the billing history by service item name, service item type name, and transaction date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	Get billing history by Service Item Type Name and Transaction Date Returns the billing history by service item type and transaction date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}

Table 3-26 Billing History API Table

Area	Examples
	Get billing history by Account ID and Transaction Date Returns the billing history by account ID and transaction date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}
	Get billing history by Account Name and Transaction Date Returns the billing history by account name and transaction date. REST URL: http://<ServerURL>/RequestCenter/nsapi/transaction/billinghistory/{columnName1}{Operator}{Value1}|{Join}|{columnName2}{Operator}{Value2}

Filters	Supports up to four filters.
	REST URL:
	Comparison Operators:
	Number/Date columns: =, >, <, >=, <=
	String columns: = equals, starts-with operators (contains and like operators are not allowed)
	Relational Operators: AND (OR is not allowed)
	Separator = |
	Allowed Columns: ServiceItemName, ServiceItemTypeName, AccountID, AccountName, and TransactionDate
	Filter Combinations:
	Allowed filter combinations are as follows:
	• ServiceItemTypeName filter, ServiceItemName filter, and TransactionDate filter
	• TransactionDate filter (one or more times)
	• TransactionDate filter and ServiceItemTypeName filter
	• TransactionDate filter and AccountID Filter
	• TransactionDate filter and AccountName filter
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName1><operator1><value1>[|<AND>|<columnName2><operator2><value2>][|<AND>|<columnName3><operator3><value3>]
	Java Example:
	`String filter = "<columnName1><operator1><value1>|<and>|<columnName2><operator2><value2>|<and>|<columnName3><operator3><value3>";`
	`ServiceItemDTO serviceItems =`

Table 3-26 Billing History API Table

Area	Examples
Sort Column(s)	Sorting is not allowed.
Response XML	<billingHistory recordSize="2" startRow="1" totalCount="1"> <billingRecord id="123"> <requisitionID>45</requisitionID> <requisitionEntryID>56</requisitionEntryID> <transactionDate local="01/03/2013 6:31 PM" raw="2013-01-04T02:31:43.400-08:00"/> <serviceItemType id="5" name="Laptop"/> <serviceItem id="4" name="DELL"/> <organizationalUnit id="5" name="My OU"/> <operation>Assemble Laptop</operation> <account id="5" name="Account Name"/> <agreement id="6" name="Agreement Name"/> <customer id="2" name="John John"/> <rateRecord rate="123.45" rateCode="RateCode1" unitOfMeasure="Seconds"/> <billingAttributes> <attribute name="Attr1">Value1</attribute> <attribute name="Attr2">Value2</attribute> <attribute name="Attr3">Value3</attribute> </billingAttributes> </billingRecord> </billingHistory>

Service Item Data

Service Item Details

Table 3-27 Service Item Details API Table

Area	Examples
CoreAPI	**Get by Name** Returns the details and subscription data of service item instances assigned to the current user for the service item name specified. If the user also has the capability "View Service Items for My Business Units", the services items for people in all the OUs to which the user belongs are returned. The service item name parameter accepts the internal table name of the service item as shown in the Name field on the Design Service Item page (for example, SiVirtualMachine). REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName> Java Example: `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemData("<serviceItemName>", null);`

Special Conditions	Support for special characters while filtering
	If you need to use special characters, such as '=', 'I', or '/ ', then use POST method with the filter in the payload (body) of the request.
	Method: POST
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/{serviceItemName}
	Sample Payload:
	{
	"filterString":"<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"
	}

Table 3-27 Service Item Details API Table

Area	Examples
	View Filters The views available for filtering correspond to what the user sees in My Services and Service Item Manager modules. Possible values for ViewName: • My ServiceItems – All instances of the service item the current user owns. Users who have the "View Service Items for My Business Units" capability also may view the items owned by other people in the OUs they belong to. • Manage ServiceItems – All instances of the service item (the view returns the data only if the user has the "Manage Service Item Instances" capability). The ViewName argument should be placed at the end if other arguments such as sort order or page size are also specified. REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>?ViewName=<viewName> Java Example: `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>", "<viewName>");`

Filters	**Service item attribute filters**
	Supports up to three filters.
	Supports all service item and subscription columns.
	Does not support filter by Service Item Classification Name (group) or Service Item Type (Cisco reserved or user-defined).
	REST URL:
	Comparison Operators:
	Number/Date columns: =, >, <, >=, <=
	String columns: = (case-sensitive; like, contains and starts-with operators are explained below)
	Relational Operators: AND, OR (case-insensitive, order precedence is not supported)
	Separator = |
	Allowed Columns: All columns in service item and subscription
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>]
	Java Example:
	`String filter = "<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>";`
	`ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName>", filter);`

Table 3-27 Service Item Details API Table

Area	Examples
	String column filters Filter for % (starts with) operator. Filter for * (Contains) operator. Does not support (ends with) operator. Examples: Name=service* Name=*g* Name=*g – not allowed REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName> /<columnName>=<wildcardValue> Java Example: `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName> ",`
	Date column attributes Date field values should be in mm-dd-yyyy format. Comparison Operators: =, >, <, <=, >= Example: SubmittedDate=12-10-2010 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceItemName> /<columnName><operator><mm-dd-yyyy> Java Example: `ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceItemName> ", "<columnName><operator><mm-dd-yyyy>");`
Sort Column(s)	Service item attributes: All table columns. Subscription: Assigned Date, Display Name, ID (internal ID), Requisition ID, Submit Date.

Table 3-27 Service Item Details API Table

Area	Examples
Response XML	<serviceitem totalCount="x" recordSize="3" startRow="1" id="62"> <logicName></logicName> <name></name> <subscription> . . . <assignedDate> </assignedDate> <assignedDateRaw></assignedDateRaw> <customerID></customerID> <customerName> </customerName> <displayName> </displayName> <id></id> <organizationalUnitID></organizationalUnitID> <organizationalUnitName> </organizationalUnitName> <requisitionEntryID></requisitionEntryID> <requisitionID></requisitionID> <serviceItemClassificationID></serviceItemClassificationID> <serviceItemTypeID></serviceItemTypeID> <serviceItemTypeName> </serviceItemTypeName> <submittedDate> </submittedDate> <submittedDateRaw></submittedDateRaw> . . . </subscription> </serviceitem>

Table 3-27 Service Item Details API Table

Area	Examples

NSAPI for Service Items v2 CRUD	The version2 (v2) nsAPI for Service items are implemented to support hierarchical service items. All v2 APIs support JSON format only (for input and responses).
	The URLs supported for v2 Service item NSAPI for Read, Create, Update, and Delete are:
	GET API-1:
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	To get all service item records for given service item type logic name, no filters.
	Example URL: http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	Sample Output:
	HTTP Status code: 200 OK
	{
	"serviceitem": {
	"id": 109,
	"name": "Desktop",
	"logicName": "SiDesktop",
	"startRow": 1,
	"recordSize": 2,
	"totalCount": 2,
	"serviceItemData": [
	{
	"items": [
	{
	"Name": "1 desktop",
	"SingleStr": "Single 1",
	"MultiStr": [
	"1-First",
	"1-2nd",
	"1-3rd"
	]
	}
	],

Table 3-27 Service Item Details API Table

Area	Examples

	"subscription": {
	"accountID": 0,
	"agreementID": 0,
	"customerID": 1,
	"id": 1,
	"organizationalUnitID": 1,
	"requisitionEntryID": 0,
	"requisitionID": 0,
	"submittedDateRaw": 1412223018210
	}
	
	]
	}
	}
	}

Table 3-27 Service Item Details API Table

Area	Examples

	GET API-2:
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}/{filters}
	To get all service item records for given service item type logic name that match the filter criteria.
	ExampleURL:
	http:<ServerURL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop/Name=1desktop
	Sample Output:
	HTTP Status code: 200 OK
	{
	"serviceitem": {
	"id": 109,
	"name": "Desktop",
	"logicName": "SiDesktop",
	"startRow": 1,
	"recordSize": 2,
	"totalCount": 2,
	"serviceItemData": [
	{
	"items": [
	{
	"Name": "1 desktop",
	"SingleStr": "Single 1",
	"MultiStr": [
	"1-First",
	"1-2nd",
	"1-3rd"
	]
	}
	],

Table 3-27 Service Item Details API Table

Area	Examples

	"subscription": {
	"accountID": 0,
	"agreementID": 0,
	"customerID": 1,
	"id": 1,
	"organizationalUnitID": 1,
	"requisitionEntryID": 0,
	"requisitionID": 0,
	"submittedDateRaw": 1412223018210
	}
	
	]
	}
	}
	}

	POST API
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	Example URL: http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	Sample Input: (The object structure for 'serviceItemData' is same in GET call response and POST/PUT calls input as well.)
	{
	“serviceitem": {
	"serviceItemData": {
	"Name": "2 desktop",
	"SingleStr": "Single 2",
	"MultiStr": ["2-1st",
	"2-2nd",
	"2-3rd"],
	"Harddisk": {
	"Name": "2 Disk",
	"Size": "200"
	},

Table 3-27 Service Item Details API Table

Area	Examples

	"Harddisk": {
	"Name": "2 Disk",
	"Size": "200"
	},
	"MultiHardDisk": [{
	"Name": "2-1 MDisk",
	"MSize": "21"
	},
	{
	"Name": "2-2 MDisk",
	"MSize": "22"
	}],
	"CPU": {
	"Name": "HP2",
	"Speed": "20"
	},
	"MultiCPU": [{
	"Name": "MHP1",
	"MSpeed": "100"
	},
	{
	"Name": "MHP2",
	"MSpeed": "200"
	}]
	},
	"subscription": {
	"loginID": "admin",
	"ouname": "Site Administration"}
	}
	}

Table 3-27 Service Item Details API Table

Area	Examples

	Sample Output:
	Response HTTP Status Code: 200 OK
	Output: {
	"nsapi-response": {
	"status-messages": [
	{
	"code": "SI_SUCCESS_001",
	"value": "Service item successfully updated."
	}
	]
	}
	}

	DELETE API
	http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/{serviceItemLogicName}
	Deletes a Service item with given input.
	Example URL: http://<ServerURL>/RequestCenter/nsapi/v2/serviceitem/SiDesktop
	Sample Input:
	{
	"serviceitem":{
	"serviceItemData": {
	"Name": "14 desktop"
	}
	}
	}
	Sample Output:
	Response HTTP Status Code: 204

Notes and Validations

1. The Service item definition must be existing to use all of these APIs.
2. The "serviceItemLogicName" in the URL must be a valid one.
3. For Create/Update/Delete inputs, below are the validations:
   a. Service Item Name is mandatory.
   b. For Create, the same service item should not be present earlier, unlike the Update and Delete.
   c. Attributes from the input JSON must exist in the SI definition.
   d. The reference records must be existing in order to refer them as reference in parent record - for both Create and Update APIs.

e. The contained records need not be existing in the Update scenario of parent record.

f. On update of a parent record, the contained child record data can be updated, except the 'Name'. The owner information should match for this.

g. Using update of a parent record, user should be able to add an existing SI record as a contained record to another existing SI (parent), if the child record never had any owner SI before.
For Example: Desktop (Parent SI) and CPU (Child SI), and an instance created for both of them separately (Desktop1 and CPU1). Now you can update the Desktop1 and attach the CPU1 to it, if CPU1 never had an owner SI earlier. With this call, CPU1 will become a contained record for Desktop1.

h. On Update, all SI records (parent or contained child records) in the input should be merged and saved. The existing attribute data should not be modified if those attributes are not specified in the input for Update API.

i. The reference records will never be updated with parent record update. They will just be referenced.

j. If you have one contained object for a parent record, and you do not want to have that contained record for the parent, you should pass empty object {} as input if it is single-value contained, or you should pass empty array [] as input if it is multi-value contained.
For example: Harddisk - Single-value contained type attribute. And MultiHardDisk - Multi-value contained type attribute.

```
{
"serviceitem": {
"serviceItemData": {
"Name": "1 desktop",
"Harddisk": {
},
"MultiHardDisk": [
]
}
}
```

k. To update the parent to remove existing reference records, the input structure should be same as above mentioned.

l. "Name" is mandatory for contained/reference child records.

m. If an attribute is not mentioned in the input for Update API, that attribute data will not be modified by update execution.

4. With Delete API, the parent record and also all the child records related to it will be deleted, except the reference records, which are independent of parent record.

5. bypass the RBAC check for service item based nsAPIs by setting the **serviceitem.nsapi.rbac.check** property in the **newscale.properties** file to *false*.

All Service Items

Table 3-28 All Service Items API Table

Area	Examples
CoreAPI	**Get all items** Shows all Subscription and Service Item data. By default, filters only the Service Items assigned to the current user. If the user also has the "View Service Items for My Business Units" capability, the service items for other people in all the OUs the user belongs to are also returned. REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription Java Example: `ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionData(null) ;`

Special Conditions	Support for special characters while filtering
	If you need to use special characters, such as '=', 'I', or '/ ', then use POST method with the filter in the payload (body) of the request.
	Method: POST
	REST URL:
	/RequestCenter/nsapi/serviceitems/serviceitemsubscription
	Sample Payload:
	{
	"filterString":"<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"
	}

Table 3-28 All Service Items API Table

Area	Examples
Filters	**View Filters** The views available for filtering correspond to what user sees in My Services and Service Item Manager modules respectively. Possible values for ViewName: • My ServiceItems – All service items the current user owns. Users who have the "View Service Items for My Business Units" get also the items owned by other people in the OUs they belong to. • Manage ServiceItems – All service items (the view returns the data only if the user has the "Manage Service Item Instances" capability). The ViewName argument should be placed at the end if other arguments such as sort order or page size are also specified. REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription?ViewName=<viewName> Java Example: `ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData ("<viewName>");`
	Subscription filters Filters for all Service Item Subscription table columns. Supports up to 3 filters. REST URL: Comparison Operators: • Number/Date columns: =, >, <, >=, <= • String columns: = (case-sensitive; like, contains and starts-with operators are explained below) Relational Operators: AND, OR (case-insensitive, order precedence is not supported) Filter Separator: | Supported Columns: All columns http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] Java Example: `String filter = "<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"; ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

Table 3-28 All Service Items API Table

Area	Examples

	String column filters
	Support % (starts with) operator.
	Support * (Contains) operator.
	Does not support (ends with) operator.
	Examples:
	`Name=service*`
	`Name=*g*`
	`Name=*g -- not allowed`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription /<columnName>=<wildcardValue>
	Java Example:
	`String filter = "<columnName>=<wildcardValue>";`
	`ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

	Date column filters
	Date field values should be in mm-dd-yyyy format.
	Comparison Operators: =, >, <, <=, >=
	Example:
	`SubmittedDate=12-10-2010`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription /<columnName><operator><mm-dd-yyyy>
	Java Example:
	`String filter = "<columnName><operator><mm-dd-yyyy>";`
	`ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);`

Sort Column(s)	Assigned Date, Customer ID, Display Name, Organizational Unit ID, Requisition ID, Requisition Entry ID, Service Item Classification ID, Service Item ID, Service Item Type ID, Service Item Type Name, Submitted Date

Table 3-28 All Service Items API Table

Area	Examples

Response XML	<AllServiceItems totalCount="x" recordSize="y" startRow="z">
	  <serviceitemsubscription displayName=" " id="">
	  . . .
	  <serviceItemTypeName> </serviceItemTypeName>
	  <organizationalUnitID></organizationalUnitID>
	  <assignedDate> </assignedDate>
	  <requisitionID></requisitionID>
	  <submittedDate> </submittedDate>
	  <submittedDateRaw></submittedDateRaw>
	  <assignedDateRaw></assignedDateRaw>
	  <customerID></customerID>
	  <requisitionEntryID></requisitionEntryID>
	  <serviceItemID></serviceItemID>
	  <serviceItemTypeID></serviceItemTypeID>
	  <organizationalUnitName> </organizationalUnitName>
	  <customerName> </customerName>
	  <serviceitem id="">
	  <logicName> </logicName>
	  <name> </name>
	  <serviceItemData rowId="">
	  <serviceItemAttribute name=" "> </serviceItemAttribute>
	  . . .
	  </serviceitemsubscription>
	</AllServiceItems>

Table 3-28 All Service Items API Table

Area	Examples

	GET all service items API
	This API is used to get all service items
	The available options are:
	• SearchFilter—name:equals:laptop
	• startRow and recordsPerPage—Pagination
	• SortBy—Column name
	• sortDir—Sort direction
	Note You must use sortBy and sortDir simultaneously.
	REST URL:
	http://<ServerURL>/ RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitemlist?
	Sample output:
	{
	"Result":{
	"recordsReturned":10,
	"totalRecords":1,
	"startIndex":1,
	"records":[
	{
	"serviceItemId":"1",
	"assignedDate":"25/05/2016 7:49 AM",
	"owner":"admin admin",
	"serviceItemTypeId":"107",
	"accountName":"",
	"classification":"UCS Director",
	"submittedDate":"25/05/2016 7:49 AM",
	"organizationUnitName":"Site Administration",
	"requisitionId":"",
	"serviceItemTypeName":"APIC Container",
	"encServiceItemId":"6M+JgN1m4ZpqG79GquDakQ==",
	"agreementName":"",
	"encServiceItemTypeId":"LAZcCouqBeYcZYsLd0hpRQ==",
	"name":"Customer",
	"id":"1"
	}
	]
	}
	}

	Gets service items for All or specific connection REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/SiSubmitSI Method: GET

Table 3-28 All Service Items API Table

Area	Examples

	Download CSV of all Service Items
	To download the CSV of all the service items
	The available options are:
	• SearchFilter—name:equals:laptop
	• startRow and recordsPerPage—Pagination
	• SortBy—Column name
	• sortDir—Sort direction
	Note You must use sortBy and sortDir simultaneously.
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/v1/mdrdatatable/serviceitemlist/CSV?
	Sample output:
	`Service Service Service Requisi Submitted Assigned Organi Acco Agree Custo`
	`Item Item Item tion ID Date Date zational unt ment mer`
	`Group Name Type Unit`
	`"UCS SuFen5 Container 8 04/19/ 04/19/ UCSD::fen::Default`
	`Director" 2016 2016 Group vu1`
	`7:02 AM 7:02 AM vu1`

Table 3-28 All Service Items API Table

Area	Examples
	Retrieve History Information To retrieve history of a service item Payload Content type: application/x-www-form-urlencoded (Form data) The form data is sent as: serviceItemTypeId=149&serviceItemId=1 • serviceItemTypeId:149 • serviceItemId:1 REST URL: http://<ServerURL>/RequestCenter/nsapi/serviceitem/v1/myservices/assetlist/history? Sample output `[ { "name": "", "operation": "Update", "requisitionId": "", "serviceId": "", "submitDate": "06/06/2016" }, { "name": "", "operation": "Create", "requisitionId": "", "serviceId": "", "submitDate": "06/06/2016" } ]`

Table 3-28 All Service Items API Table

Area	Examples
	Retrieve Classification Tree To retrieve the classification tree of the service items REST URL: http://<ServerURL> /RequestCenter/nsapi/serviceitem/v1/sim/serviceitemtypestree/classificationdatatypetree ? Sample Output `{ "data":[{ "data":[{ "hasChildren":false, "iconCls":"sitinstance", "id":"107", "leaf":true, "module":"itemType", "moduleId":"107S", "text":"APIC Container", "vsocTenantNodeType":"" }], "hasChildren":true, "iconCls":"sitfolder", "id":"12", "leaf":false, "module":"classification", "moduleId":"12C", "text":"Cloud Infrastructure", "vsocTenantNodeType":"" }] }`

Create/Update/Delete APIs on Service Items

Following table describes the nsAPI URLs for doing create/update/delete operations on service item. Request Content-Type can be either of application/xml or application/json. Default response content-type will be the same as Request content-type i.e if the input data is in xml format, response data

will also be in xml format. These can be modified by providing additional query parameter 'responseType', whose possible values are 'xml' or 'json'. For example, for an xml input, the response can be got in json format if responseType=json is provided in the URL.

Table 3-29 Create/Update/Delete APIs on Service Items

Area	Examples

Core API	Create Service Item
	POST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	`<serviceitem>`
	`  <name>custom_sitype</name>`
	`  <serviceItemData>`
	`  <serviceItemAttribute name="Name">sit1</serviceItemAttribute>`
	`  <serviceItemAttribute name="Field1">a</serviceItemAttribute>`
	`  <serviceItemAttribute name="Field2">b</serviceItemAttribute>`
	`  </serviceItemData>`
	`  </serviceitem>`
	json:
	`{`
	`  "serviceitem" : {`
	`  "name" : "custom_sitype",`
	`  "serviceItemData" : {`
	`  "serviceItemAttribute" : [ {`
	`  "name" : "Name",`
	`  "value" : "sit1"`
	`  }, {`
	`  "name" : "Field1",`
	`  "value" : "a"`
	`  }, {`
	`  "name" : "Field2",`
	`  "value" : "b"`
	`  } ]`
	`  }`
	`  }`
	`}`

Table 3-29 Create/Update/Delete APIs on Service Items

Area	Examples

	Update Service Item
	PUT URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	`<serviceitem>`
	`<name>custom_sitype</name>`
	`<serviceItemData>`
	`<serviceItemAttribute name="Name">sit1</serviceItemAttribute>`
	`<serviceItemAttribute name="Field1">a</serviceItemAttribute>`
	`<serviceItemAttribute name="Field2">b</serviceItemAttribute>`
	`</serviceItemData>`
	`</serviceitem>`

	Delete Service Item
	DELETE URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	`<serviceitem>`
	`<name>custom_sitype</name>`
	`<serviceItemData>`
	`<serviceItemAttribute name="Name">sit1</serviceItemAttribute>`
	`</serviceItemData>`
	`</serviceitem>`

	Custom Operation on Service Item
	Performs custom operation on a service item.
	PUT URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process?operation=test
	xml:
	`<serviceitem>`
	`<name>custom_sitype</name>`
	`<serviceItemData>`
	`<serviceItemAttribute name="Name">sit1</serviceItemAttribute>`
	`<serviceItemAttribute name="Field1">a</serviceItemAttribute>`
	`<serviceItemAttribute name="Field2">b</serviceItemAttribute>`
	`</serviceItemData>`
	`</serviceitem>`

Table 3-29 Create/Update/Delete APIs on Service Items

Area	Examples

	Create Service Item and associate with particular OU
	Creates a service item and associates it with particular OU, person etc by providing service item subscription details in the request data.
	POST URL:
	http://<ServerURL>/RequestCenter/nsapi/serviceitem/process
	xml:
	`<serviceitem>`
	`<name>custom_sitype</name>`
	`<serviceItemData>`
	`<serviceItemAttribute name="Name">sit1</serviceItemAttribute>`
	`<serviceItemAttribute name="Field1">a</serviceItemAttribute>`
	`<serviceItemAttribute name="Field2">b</serviceItemAttribute>`
	`<subscription>`
	`<loginID>admin</loginID>`
	`<ouname>testOU1</ouname>`
	`<accountName>account1</accountName>`
	`<requisitionEntryID>6</requisitionEntryID>`
	`</subscription>`
	`</serviceItemData>`
	`</serviceitem>`
	json:
	`{`
	`"serviceitem" : {`
	`"name" : "custom_sitype",`
	`"serviceItemData" : {`
	`"serviceItemAttribute" : [ {`
	`"name" : "Name",`
	`"value" : "sit1"`
	`}, {`
	`"name" : "Field1",`
	`"value" : "a"`
	`}, {`
	`"name" : "Field2",`
	`"value" : "b"`
	`} ],`
	`"subscription" : {`
	`"loginID" : "admin",`
	`"ouname" : "testOU1",`
	`"accountName" : "account1",`
	`"requisitionEntryID" : ""`
	`}`
	`}`
	`}`
	`}`

Filters	Supports no filters.

Table 3-29 Create/Update/Delete APIs on Service Items

Area	Examples
Sort Column(s)	Sorting is not allowed.

Response XML	xml:
	<nsapi-response>
	<status-message>
	<value>Service item [sit1] successfully created.</value>
	</status-message>
	</nsapi-response>
	
	<nsapi-response>
	<status-message>
	<value>Operation 'Test' of service item [sit1] successfully performed.</value>
	</status-message>
	</nsapi-response>
	
	json:
	{
	"nsapi-response" : {
	"status-messages" : [{
	"value" : "Service item [sit140211] successfully created."
	}]
	}
	}

Grant and Revoke Permissions

nsAPI is only supported for assigning/revoking permissions for Service Items and Standards. Record level Permissions on SI Instance Data can only be defined from SIM module or through nsAPI (and NOT from Org Designer).

Table 3-30 Grant and Revoke Permissions API Table

Area	Examples
CoreAPI	Grant Permissions Grants permissions on Service Items and Standards. REST URL: http://<ServerURL>/RequestCenter/nsapi/rbac/extensibleschema/grantpermission All permissions related to SIM RBAC can be exercised via this nsAPI: • Grant Specific/All SI Group/Standard Group Definition permission to a Role. • Grant Specific/All Service Item/Standard Definition permission to a Role. • Grant Specific/All ServiceItem/Standard Instance Data permission to a Role. • Grant SI Instance/Record level permission for a specific SI Instance/record to Person/OU/Group/Role/Account. • Grant “Maintain Billing Rates” permission for All Objects.
	Revoke Permissions Revokes permissions on Service Items and Standards. REST URL: http://<ServerURL>/RequestCenter/nsapi/rbac/extensibleschema/revokepermission All permissions related to SIM RBAC can be exercised via this nsAPI: • Revoke Specific/All SI Group/Standard Group Definition permission for a Role. • Revoke Specific/All Service Item/Standard Definition permission to a Role. • Revoke Specific/All ServiceItem/Standard Instance Data permission to a Role. • Revoke SI Instance/Record level permission for a specific SI Instance/record to Person/OU/Group/Role/Account. • Revoke “Maintain Billing Rates” permission for All Objects.
Filters	Supports no filters.

Table 3-30 Grant and Revoke Permissions API Table

Area	Examples
Sort Column(s)	Sorting is not allowed.

Request XML	Grant Permissions:
	Headers : <Username> <password> and <Content-Type=application/xml> Method=POST
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<grantpermission scope="SpecificObject/AllObjects"`
	`object="Instance/Definition/Group"`
	`recipient="Role/OrgUnit/Person/Group/ProjectAccount/TenantAccount"`
	`recipientName=<Name of the Recipient> recipientType="ServiceTeam/BuisnessUnit - Applicable only if the recipient is set to OrgUnit"`
	`domain="ServiceItem/Standard" permission=<Logical Names> instanceName=<Name of the Instance if object="Instance"> objectName=<Name of the attribute object> />`
	Example:
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<grantpermission scope="SpecificObject" object="Instance" recipient="OrgUnit" recipientName="test5466" recipientType="ServiceTeam" domain="ServiceItem" permission="mdr_record_read" instanceName="Cap" objectName="SravItem"/>`
	Revoke Permissions:
	Headers : <Username> <password> and <Content-Type=application/xml> Method=POST
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<revokepermission scope="SpecificObject/AllObjects"`
	`object="Instance/Definition/Group"`
	`recipient="Role/OrgUnit/Person/Group/ProjectAccount/TenantAccount"`
	`recipientName=<Name of the Recipient> recipientType="ServiceTeam/BuisnessUnit - Applicable only of the recipient is set to OrgUnit"`
	`domain="ServiceItem/Standard" permission=<Logical Names> instanceName=<Name of the Instance if object="Instance"> objectName=<Name of the attribute object> />`
	Example:
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<revokepermission scope="SpecificObject" object="Instance"`
	`recipient="OrgUnit" recipientName="test5466" recipientType="ServiceTeam"`
	`domain="ServiceItem" permission="mdr_record_read" instanceName="Cap"`
	`objectName="SravItem"/>`
	For more information about the permissions that you can grant/revoke to specific objects, see Permissions that you can grant/revoke to specific objects table.

Table 3-31 Permissions that you can grant/revoke to specific objects

Scope	UI Label	Permission	object= "Definition", domain= "ServiceItem"	object= "Group", domain= "ServiceItem"	object= "Instance", domain= "ServiceItem"	object= "Definition", domain= "Standard"	object= "Group", domain= "Standard"	object= "Instance", domain= "Standard"
AllObjects /SpecificObject	Read	mdr_record_read	Not Applicable	Not Applicable	Not Applicable for AllObjects, Applicable for Specific Objects if instance Name is not null	Not Applicable	Not Applicable	Not Applicable
	Write	mdr_record_write	Not Applicable	Not Applicable	Not Applicable for AllObjects, Applicable for Specific Objects if instance Name is not null	Not Applicable	Not Applicable	Not Applicable
	Read Definition	mdr_datatype_read	Applicable	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable
	ReadWrite Definition	mdr_datatype_write	Applicable	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable
	Read all Instance Data	content_def_read_allcontentdata	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Applicable
	ReadWrite all Instance Data	content_def_write_allcontentdata	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Applicable
	Read all Instance Data in my BU	allinstancedata_my_bu_read	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	ReadWrite all Instance Data in my BU	allinstancedata_my_bu_write	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable

Table 3-31 Permissions that you can grant/revoke to specific objects

Scope	UI Label	Permission	object= "Definition", domain= "ServiceItem"	object= "Group", domain= "ServiceItem"	object= "Instance", domain= "ServiceItem"	object= "Definition", domain= "Standard"	object= "Group", domain= "Standard"	object= "Instance", domain= "Standard"
	Read all Instance Data in my BU and their Sub-Units	allinstance data_my_bu_hierarachy_read	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	ReadWrite all Instance Data in my BU and their Sub-Units	allinstance data_my_bu_hierarachy_write	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	Read all Instance Data in my Tenant Account	allinstance data_my_tenantaccount_read	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	ReadWrite all Instance Data in my Tenant Account	allinstance data_my_tenantaccount_write	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	Read all Instance Data in my Project Account	allinstance data_my_projectaccount_read	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	ReadWrite all Instance Data in my Project Account	allinstance data_my_projectaccount_write	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	Read all Instance Data Subscribed by Self	allinstance data_subscribed_self_read	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable

Table 3-31 Permissions that you can grant/revoke to specific objects

Scope	UI Label	Permission	object= "Definition", domain= "ServiceItem"	object= "Group", domain= "ServiceItem"	object= "Instance", domain= "ServiceItem"	object= "Definition", domain= "Standard"	object= "Group", domain= "Standard"	object= "Instance", domain= "Standard"
	ReadWrite all Instance Data Subscribed by Self	allinstance data_subscribed_self_write	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
	Create New Instance Data	create_new_service_item	Not Applicable	Not Applicable	Applicable	Not Applicable	Not Applicable	Not Applicable
	Create New Standard Instance Data	create_new_standard_instance_data	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Applicable
	Read all Definitionin this Group	mdr_class_read	Not Applicable	Applicable	Not Applicable	Not Applicable	Applicable	Not Applicable
	ReadWrite all Definitionin thiGroup	mdr_class_write	Not Applicable	Applicable	Not Applicable	Not Applicable	Applicable	Not Applicable
AllObjects	MaintainBilling Rates	billingrates_maintain	object="Definition"	Not Applicable	domain="Billing"	object="Definition"	Not Applicable	Not Applicable

Standards

Table 3-32 Standards API Table

Area	Examples
CoreAPI	**Get by Name** By default returns the first 50 entries for the specified Standard. REST URL: http://<ServerURL>/RequestCenter/nsapi/standard/<standardName> Java Example: `StandardDTO standards = NSApiClient.getStandard().getStandardData("<standardName>", null);`
Filters	Supports up to three filters. REST URL: Comparison Operators: • Number/Date columns: =, >, <, >=, <= (case-sensitive) • String columns: = (like, contains and starts-with operators are explained below) Relational Operators: AND, OR (case-insensitive, order precedence is not supported) Filter Separator: | Supported Columns: All columns http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>] Java Example: `String filter: "<columnName1><operator1><value1>|<and|or>|<columnName2><operator2><value2>|<and|or>|<columnName3><operator3><value3>"; StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", filter);`

Table 3-32 Standards API Table

Area	Examples

	String column filters
	Support % (starts with) operator.
	Support * (Contains) operator.
	Does not support (ends with) operator.
	Examples:
	`Name=service*`
	`Name=*g*`
	`Name=*g -- not allowed`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName>=<wildcardName>
	Java Example:
	`StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<wildcardName>");`

	Date column filters
	Date field values should be in mm-dd-yyyy format.
	Comparison Operators: =, >, <, <=, >=
	Example:
	`SubmittedDate=12-10-2010`
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName><operator><mm-dd-yyyy>
	Java Example:
	`StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<columnName><operator><mm-dd-yyyy>");`

Sort Column(s)	All table columns.

Response XML	`<standard totalCount="x" startRow="y" recordSize="z" id="a">`
	`<loginName></loginName>`
	`<name></name>`
	`<standardData rowId="">`
	`. . .`
	`<standardAttribute name="id" />`
	`. . .`
	`<standardURL><a ></a></standardURL>`
	`<standardURLOnly> </standardURLOnly>`
	`. . .`
	`</standardData>`
	`</standard>`

Service Catalog Data

Custom Content

Custom content comprises user-defined tables that serve as a source of content for the service portal. Such tables can be referenced as the data source for portlets just like Standards.

Table 3-33 Custom Content API Table

Area	Examples
CoreAPI	**Get by Name** By default returns the first 50 entries in the specified table. REST URL: http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName> Java Example: `CustomContentDTO customs = NSApiClient.getCustomContent().getcontentData("<customContentName>", null);`

Filters	Support all custom content columns.
	Support % (starts with) operator.
	Do not support (ends with) operator.
	Examples:
	`Name=custom*`
	`Name=*g -- not allowed`
	REST URL:
	Comparison Operators: =, >, <, >=, <= (case-sensitive)
	Relational Operators: AND, OR (case-insensitive)
	Filter Separator: |
	Supported Columns: All columns
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName1><operator1><value1>[|<AND|OR>|<columnName2><operator2><value2>][|<AND|OR>|<columnName3><operator3><value3>]
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName>=<wildcardName>
	http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName><operator><mm-dd-yyyy>
	Java Example:
	`String filter = "<columnName><operator><value>";`
	`CustomContentDTO customs = NSApiClient.getCustomContent().getcustomContentDataWithFilters("CoPortalContent", filter);`

Table 3-33 Custom Content API Table

Area	Examples
Sort Column(s)	All table columns
Response XML	<customContent totalCount="x" startRow="y" recordSize="z" id="a"> <logicName></logicName> <name></name> . . . <customContentData rowId=""> <customContentAttribute name=" "></customContentAttribute> . . . </customContentData> </customContent>

The complete list of column names and descriptions for the entities can be found in the “Reference Data” section in the “Designing Portals”, chapter of the *Cisco Prime Service Catalog Designer Guide*.

Tenant Management

Table 3-34 Tenant Management API Table

Area	Examples
Get API	**Fetch User Roles API** Method: GET REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/userroles Sample Response: `{ "roles": { "startRow": 0, "totalCount": 0, "recordSize": 0, "role": [{ "roleId": 73, "roleName": "Service Administrator", "statusId": 1, "roleTypeId": 2, "logicName": "service-administrator", "inherited": false, "parentId": 0 }, { "roleId": 1336, "roleName": "Team Administrator", "statusId": 1, "roleTypeId": 2, "logicName": "team-administrator", "inherited": false, "parentId": 0 }] } }`
Get API	**Fetch Tenant Management Configuration API** Method: GET REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/tenantconfiguration

Table 3-34 Tenant Management API Table

Area	Examples

	Sample Response:
	{
	"ArrayList": [
	{
	"name": "TenantType",
	"value": "Multi"
	},
	{
	"name": "TenantID",
	"value": "System generated UUID"
	},
	{
	"name": "TenantName",
	"value": "User ID"
	},
	{
	"name": "Organization",
	"value": "CSV list imported"
	},
	{
	"name": "Billing",
	"value": "Active"
	},
	{
	"name": "BillingType",
	"value": "Credit Card"
	},
	{
	"name": "BillingDept",
	"value": "Dynamic join to people data in LDAP"
	},
	{
	"name": "FinApp",
	"value": "Yes"
	},
	{
	"name": "FinAppType",
	"value": "Add approvers using CSV"
	},
	{
	"name": "EntApp",
	"value": "Yes"
	},
	{
	"name": "TenantApp",
	"value": "Yes"
	},
	{
	"name": "TenantAppType",
	"value": "Parent Tenant Admin (Multi Only)"
	}
	]
	}
	Note If you don't have configuration, the Response = **ArrayList: [ ]**.

Table 3-34 Tenant Management API Table

Area	Examples

Get API	**Fetch Hierarchical teams and team details API for Card and list view with admin member information**
	Method: GET
	REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}
	Sample Request:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/1040
	Sample Response:
	{
	"TeamList": {
	"startRow": 0,
	"totalCount": 2,
	"recordSize": 0,
	"teams": [
	{
	"teamName": "Root Team",
	"teamId": 39,
	"statusId": 0,
	"status": "Active",
	"parentId": 0,
	"parentName": "",
	"accountId": 22,
	"hasChild": false,
	"memberCount": 2,
	"createdOn": 1477035094013,
	"createdOnString": "10/21/2016 12:31 AM",
	"adminMembers": [
	{
	"memberName": "root cisco",
	"memberId": 25,
	"status": "Active",
	"email": "root@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T122",
	"parentTenant": "",
	"rootTenantDisplayName": "Root Team",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": true
	},
	{
	"teamName": "Tenant A",
	"teamId": 1040,
	"statusId": 0,
	"status": "Active",

Table 3-34 Tenant Management API Table

Area	Examples


```
                "parentId": 0,
                "parentName": "",
                "accountId": 23,
                "hasChild": true,
                "memberCount": 2,
                "createdOn": 1478049861410,
                "createdOnString": "11/01/2016 6:24 PM",
                "childTeams": [
                    {
                        "teamName": "Tenant B",
                        "teamId": 1041,
                        "statusId": 0,
                        "status": "Active",
                        "parentId": 1040,
                        "parentName": "Tenant A",
                        "accountId": 23,
                        "hasChild": true,
                        "memberCount": 2,
                        "createdOn": 1478050009523,
                        "createdOnString": "11/01/2016 6:26 PM",
                        "adminMembers": [
                            {
                                "memberName": "adminB cisco",
                                "memberId": 35,
                                "status": "Active",
                                "email": "adminB_tl@cisco.com",
                                "type": "Team Admin"
                            }
                        ],
                        "rootTenant": "T123",
                        "parentTenant": "T123",
                        "rootTenantDisplayName": "Tenant A",
                        "currentTenantAdmin": false,
                        "currentParentTenantAdmin": false
                    }
                ],
                "adminMembers": [
                    {
                        "memberName": "UserA UserA",
                        "memberId": 14,
                        "status": "Active",
                        "email": "usera_tl@cisco.com",
                        "type": "Team Admin"
                    }
                ],
                "rootTenant": "T123",
                "parentTenant": "",
                "rootTenantDisplayName": "Tenant A",
                "currentTenantAdmin": false,
                "currentParentTenantAdmin": false
            }
        ]
    }
}
For the first time loading data, you have to pass teamid=0.
For example: http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/0
```


Note If you don't have tenant account associated, you will get {} blank response back.

Table 3-34 Tenant Management API Table

Area	Examples

Get API	**Fetch team list API for list view with admin member information**
	SearchPattern would be in this format: *name*
	SortBy: Name, Status, CreatedOn
	SortDir: Asc, Desc
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}&sortDir={sortDir}&name={searchName}
	Sample Request:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/teamlist?startRow=1&recordSize=20&sortBy=Nams&sortDir=Asc&name=*
	Sample Response:
	<pre>{
	"TeamList": {
	"startRow": 1,
	"totalCount": 4,
	"recordSize": 20,
	"teams": [
	{
	"teamName": "Root Team",
	"teamId": 39,
	"statusId": 0,
	"status": "Active",
	"parentId": 0,
	"parentName": "",
	"accountId": 22,
	"hasChild": false,
	"memberCount": 2,
	"createdOn": 1477035094013,
	"createdOnString": "10/21/2016 12:31 AM",
	"adminMembers": [
	{
	"memberName": "root cisco",
	"memberId": 25,
	"status": "Active",
	"email": "root@cisco.com",
	"type": "Team Admin"
	}
	],
	"rootTenant": "T122",
	"parentTenant": "",
	"rootTenantDisplayName": "Root Team",
	"currentTenantAdmin": false,
	"currentParentTenantAdmin": false
	},</pre>

Table 3-34 Tenant Management API Table

Area	Examples


```
                {
                 "teamName": "Tenant A",
                 "teamId": 1040,
                 "statusId": 0,
                 "status": "Active",
                 "parentId": 0,
                 "parentName": "",
                 "accountId": 23,
                 "hasChild": true,
                 "memberCount": 2,
                 "createdOn": 1478049861410,
                 "createdOnString": "11/01/2016 6:24 PM",
                 "adminMembers": [
                     {
                         "memberName": "UserA UserA",
                         "memberId": 14,
                         "status": "Active",
                         "email": "usera_tl@cisco.com",
                         "type": "Team Admin"
                     }
                 ],
                 "rootTenant": "T123",
                 "parentTenant": "",
                 "rootTenantDisplayName": "Tenant A",
                 "currentTenantAdmin": false,
                 "currentParentTenantAdmin": false
             },
             {
                 "teamName": "Tenant B",
                 "teamId": 1041,
                 "statusId": 0,
                 "status": "Active",
                 "parentId": 1040,
                 "parentName": "Tenant A",
                 "accountId": 23,
                 "hasChild": true,
                 "memberCount": 2,
                 "createdOn": 1478050009523,
                 "createdOnString": "11/01/2016 6:26 PM",
                 "adminMembers": [
                     {
                         "memberName": "adminB cisco",
                         "memberId": 35,
                         "status": "Active",
                         "email": "adminB_tl@cisco.com",
                         "type": "Team Admin"
                     }
                 ],
                 "rootTenant": "T123",
                 "parentTenant": "T123",
                 "rootTenantDisplayName": "Tenant A",
                 "currentTenantAdmin": false,
                 "currentParentTenantAdmin": false
             },
```

Table 3-34 Tenant Management API Table

Area	Examples


```
                    {
                "teamName": "Tenant C",
                "teamId": 1042,
                "statusId": 0,
                "status": "Active",
                "parentId": 1041,
                "parentName": "Tenant B",
                "accountId": 23,
                "hasChild": false,
                "memberCount": 1,
                "createdOn": 1478050137437,
                "createdOnString": "11/01/2016 6:28 PM",
                "adminMembers": [
                    {
                        "memberName": "adminC cisco",
                        "memberId": 36,
                        "status": "Active",
                        "email": "adminC_tl@cisco.com",
                        "type": "Team Admin"
                    }
                ],
                "rootTenant": "T123",
                "parentTenant": "T124",
                "rootTenantDisplayName": "Tenant A",
                "currentTenantAdmin": false,
                "currentParentTenantAdmin": false
            }
        ]
    }
}
```

Table 3-34 Tenant Management API Table

Area	Examples

Get API	**Fetch SubTeams by parent team Id API**
	searchPattern would be in this format: *name*
	SortBy : Name, Status, CreatedOn, or MemberCount
	SortDir: Asc, Desc
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamid}/teamlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}& sortDir={sortDir}&name={searchName}
	Sample Request:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/3/teamlist?startRow=1&recordSize=10&sortBy=Name& sortDir=Asc&name=*
	Sample Response:
	{
	"TeamList": {
	"startRow": 1,
	"totalCount": 2,
	"recordSize": 10,
	"teams": [
	{
	"teamName": "Program Management",
	"teamId": 4,
	"statusId": 1,
	"status": "Active",
	"parentId": 3,
	"parentName": "Cisco Team",
	"accountId": 0,
	"hasChild": false,
	"memberCount": 3,
	"createdOn": 1475721937253,
	"createdOnString": "10/05/2016 7:45 PM"
	},
	{
	"teamName": "UX Team",
	"teamId": 5,
	"statusId": 2,
	"status": "Inactive",
	"parentId": 3,
	"parentName": "Cisco Team",
	"accountId": 0,
	"hasChild": false,
	"memberCount": 0,
	"createdOn": 1475721947930,
	"createdOnString": "10/05/2016 7:45 PM"
	}
	]
	}
	}

Table 3-34 Tenant Management API Table

Area	Examples

Get API	**Fetch team member list API**
	SortBy: Name, Email, or Status
	SortDir: Asc, Desc
	SearchName: *name*
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/{teamId}/memberlist?startRow={startRow}&recordSize={recordSize}&sortBy={sortBy}& sortDir={sortDir}&name={searchName}
	Sample Request:
	http://<ServerURL>/RequestCenter/nsapi/directory/tenant/v2/id/3/memberlist?startRow=1&recordSize=10&sortBy=Name&sortDir=ASc&name=*
	Sample Response:
	{
	"TeamMemberList": {
	"startRow": 1,
	"totalCount": 3,
	"recordSize": 10,
	"teamMembers": [
	{
	"memberName": "admin admin",
	"memberId": 1,
	"status": "Active",
	"email": "thanes_demo@cisco.com",
	"type": "Member"
	},
	{
	"memberName": "Ryan Marfone",
	"memberId": 5,
	"status": "Active",
	"email": "Ryan@cisco.com",
	"type": "Team Admin"
	},
	{
	"memberName": "test test",
	"memberId": 3,
	"status": "Active",
	"email": "test@test.com",
	"type": "Team Admin"
	}
	]
	}
	}

Table 3-34 Tenant Management API Table

Area	Examples

PUT API	**Promote member as team admin**
	This API will:
	• assign team-admin role to user
	• unassign team-admin role to existing team admin user
	• assign it-user role to existing team admin user
	Method: PUT
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantId}/admins
	QueryParam: action = add/remove
	Sample Request:
	http://localhost:8080/RequestCenter/nsapi/directory/v2/tenant/6/admins
	Sample Payload:
	{
	"List": [20,21,12]
	}

Post API	**Update Cloud Center user API key**
	Method: POST
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/{personId}/updateapikey
	Sample Request:
	http://<ServerURL>/RequestCenter/nsapi/directory/people/id/10/updateapikey
	Sample Response:
	{
	"status-message": {
	"code": "TA_054",
	"value": "API Key updated successfully"
	}
	}
	Error Response:
	{
	"status-message": {
	"code": "TA_054",
	"value": "Cloud Center User ID does not exist"
	}
	}
	Error Response:
	{
	"status-message": {
	"code": "TA_054",
	"value": "Cloud Center Profile for the User does not exist"
	}
	}

Table 3-34 Tenant Management API Table

Area	Examples
	Fetches all the information of the project tenants Method: GET REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v1/projecttenant

	Gets assigned services for person with the filters for name, description, connection service group
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantId}/services/assigned?
	Sample response:
	`[`
	`{`
	`"description": "Java based blogger application",`
	`"id": 162,`
	`"inherited": false,`
	`"name": "RollerWeBlog_CICD (v5.6) (CLI)"`
	`},`
	`{`
	`"description": "SugarCRM enables businesses to create extraordinary customer relationships with the most innovative and affordable CRM solution in the market.\nUser:admin, Pass:password",`
	`"id": 164,`
	`"inherited": false,`
	`"name": "SugarCRM Community Ed (v6.5.18r) (CLI)"`
	`}`
	`]`
	QueryParams:
	`name = {name}`
	`description = {descirption}`
	`connection = <connectionshortName>`
	`>> service from this connection will be returned`
	`serviceGroup = <service group id>`
	`>> services from this service group will be returned`

Table 3-34 Tenant Management API Table

Area	Examples
	Gets all the tenants matching the filter Filter: startRow=1&recordSize=20&sortBy20=Name&sortDir=ASC&type=2,3,4&tenantName=*&type=3,2&status=active&adminUserId=1&parent=4&myTenants=true Method: GET REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant? • havingServicesFromConnection=<connectionShortName>, returns tenants which have services from given connection assigned to them. • havingServicesFromServiceGroup=<serviceGroupID>, returns tenants which have services from given service group assigned to them.
	Gets all the tenant admins of the specified tenant ID Method: GET REST URL: http://<ServerURL>//RequestCenter/nsapi/directory/v2/tenant/{TenantID}/admins? Sample response: `[ { "id": 7, "name": "testuser1" },{ "id": 8, "name": "testuser2" } ]`
	Creates sub teams by the tenant admin Method: POST REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant Sample Payload: `{ "TenantDTO": { "name":"Project team 11", "parentTenantId": 22, "type": 4 } }`

Table 3-34 Tenant Management API Table

Area	Examples
	Adds users to the team Method: PUT REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}/users?action=add Sample Payload: {"List":[15]}
	Removes users from the team Method: PUT REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}/users?action=remove Sample Payload: {"List":[15]}
	Activates or deactivates team Method: PUT REST URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantID}?activate={true or false}

Table 3-34 Tenant Management API Table

Area	Examples

	Gets all the requisitions ordered by the User for all teams the user belongs to
	Method: GET
	REST URL:
	http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/RequisitionViewName=[MyTeams]
	Sample Response:
	{
	"requisitions": {
	"startRow": 1,
	"totalCount": 1,
	"recordSize": 1,
	"requisition": [
	{
	"tenantId": 1,
	"userId": 7,
	"ownerId": 6,
	"serviceId": 6,
	"customerId": 6,
	"expectedDuration": 0.0,
	"actualDuration": 0.0,
	"startedDate": "05/31/2017 5:15 AM",

Table 3-34 Tenant Management API Table

Area	Examples
	"dueDate": "05/31/2017 4:00 PM", "expectedCost": 0.0, "status": "Ongoing", "requisitionId": 5, "flagImage": "/RequestCenter/images/flaglate.gif", "lateFlag": true, "customerName": "user1 user1", "organizationalUnitName": "PT2User1", "submitDate": "05/31/2017 5:15 AM", "statusId": 1, "serviceName": "service1", "ownerName": "user1 user1", "organizationalUnitId": 17, "startedDateRaw": 1496187954343, "dueDateRaw": 1496226600000, "submitDateRaw": 1496187956843, "requisitionURL": "5", "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=5", "milestoneLink": " ", "percentageCompleted": 0.0, "customerEmail": "suprajas@cisco.com", "isCancelable": true, "rating": "0", "reviewsCount": 0, "selfRating": 0, "isServiceOrderable": 1, "isServiceActive": 1, "tenantPrimaryId": 9, "tenantDisplayName": "PT2User1" }] } }

Table 3-34 Tenant Management API Table

Area	Examples
	Gets all the requisitions for the tenant the logged in user belongs to or has admin privileges on the tenant URL: http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/RequisitionViewName=[MyTenants]?teamId={teamId} Method: GET

	Gets the Global Configurations for team Management
	Note: This API can be called only by the Site Administrator.
	Method: GET
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/globalConfigurations
	Sample Response:
	{
	"approvalRequired": false,
	"tMActive": true,
	"tenantType": "ProviderBusinessTenant"
	}

	Sets the Global Configurations for team Management
	Note: This API can be called only by the Site Administrator.
	Method: PUT
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/globalConfigurations
	Sample Payload:
	{
	"TenantGlobalConfigurationDTO": {
	"approvalRequired" : false
	}
	}

	Gets all the active or inactive teams list URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant?status={active or inactive} Method: GET

Table 3-34 Tenant Management API Table

Area	Examples

	Get all teams of the logged in user
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/users/{userId}
	Method: GET
	Sample Response:
	{
	"tenant": {
	"tenantId": 88,
	"name": " Test Team",
	"displayName": " Test Team",
	"description": " Test Team",
	"typeId": 3,
	"parentTenantName": "ProviderBusinessTenant",
	"adminDetails": [
	{
	"id": 57553,
	"firstName": "testadmin",
	"lastName": "testadmin",
	"loginName": "testadmin",
	"email": "testadmin@cisco.com"
	}
	],
	"parentTenantId": 2,
	"childCount": 0,
	"requisitionsCount": 3,
	"serviceItemsCount": 1,
	"userRequisitionsCount": 2,
	"userServiceItemsCount": 1,
	"membersCount": 1,
	"assignedServicesCount": 0,
	"accountId": 3,
	"orgId": 35484,
	"status": "Active",
	"serviceGroupId": 0
	}
	}

Table 3-34 Tenant Management API Table

Area	Examples

	Get pending activities for team
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/users/{userid}/pendingactivities
	Method: GET
	Sample Response:
	{
	"List": [
	{
	"requisitionId": 101362,
	"serviceName": "Create Team",
	"stateId": 11,
	"tenantName": "PSC BLR team",
	"parentTenantName": "ProviderBusinessTenant",
	"submittedOn": 1503871669000,
	"submittedOnDate": "08/27/2017 08:37 PM"
	}
	]
	}

	Get roles for team
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/v2/tenant/{tenantid}/roles
	Method: GET
	{
	"roles": {
	"startRow": 1,
	"totalCount": 1,
	"recordSize": 1,
	"role": [
	{
	"roleId": 13,
	"roleName": "My Services Con-sumer",
	"statusId": 0,
	"roleTypeId": 2,
	"logicName": "myservices-consumer",
	"inherited": false,
	"parentId": 0
	}
	]
	}
	}

Table 3-34 Tenant Management API Table

Area	Examples

	Gets paginated and filterable list of users which belong to tenants accessible to the logged in user as tenant admin.
	Only members will be shown. Does not cover the tenant admin users if they are not members.
	URL: http://<ServerURL>/RequestCenter/nsapi/directory/users
	Method: GET
	Sample Response:
	{
	"PersonList": {
	"startRow": 1,
	"totalCount": 3,
	"recordSize": 3,
	"people": [
	{
	"name": "admin admin",
	"firstName": "admin",
	"lastName": "admin",
	"personId": 1,
	"recordStateId": 0,
	"timeZoneId": 0,
	"homeOrganizationalUnitId": 0,
	"login": "admin",
	"localeId": 0,
	"statusId": 0,
	"status": "Inactive",
	"supervisorId": 0,
	"teamCount": 0,
	"personExtension": {
	"id": 1,
	"personId": 1,
	"intManagementLevel": 0
	}
	},

Catalog Deployer API

Table 3-35 Catalog Deployer API

Area	Example
Rex API	**Get Service data** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/service/{name} Request Header Parameters: username & password Response: rex xml
	Get Service data **Note** REST URL: RequestCenter/rexapi/entity/service?name=servicename Request Header Parameters: username & password Response: rex xml
	Get Category data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/category/{name} Request Header Parameters: username & password Response: rex xml

Table 3-35 Catalog Deployer API

Area	Example
Rex API	**Get Organizationalunit data** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/organizationalunit/{name} Request Header Parameters: username & password Response: rex xml. For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Group data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/group/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Person data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/person/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Queue data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/queue/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Functional position data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/functionalposition/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.

Table 3-35 Catalog Deployer API

Area	Example
Rex API	**Get Role data** Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/role/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Email Template data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/emailtemplate/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Agent data Method: GET REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/agent/{name} Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.
	Get Rex data Method: POST REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/getdata Request Header Parameters: username & password Response: rex xml For more information on rex xml, *see* Example of the Rex XML Structure. **Note** The return xml will not include Header parameters.
	To import data Method: POST REST URL: http://<ServerURL>/RequestCenter/rexapi/entity/putdata Request Header Parameters: username & password Request Body: rex xml For more information on rex xml, *see* Example of the Rex XML Structure.

Example of the Rex XML Structure

```
Service:
<rex>
    <Header requestAction="put"/>
```

```
    <ServiceDefinitions>
        <ServiceDefinition name="Service A" serviceGroupName="0 Service Icons"
catalogName="" type="notspecified" isActive="true" isReportable="false"
isEntitlement="false" isOrderable="true" computePrice="false" authorizations="site"
canStartLater="false" dateQuality="tbd" guid="EBC80F54-01E3-45C1-8DF2-1C35C1165838"
orderingMode="Add review enabled" paginationViewMode="Classic View"
hideInServiceCatalog="false" isTemplate="false" showOrderSummary="false" maxQuantity="0"
isBaseService="false">
            <Transaction actionRequested="getXml" actionResult="Succeeded"
actionResultCode="0" actionTaken="fetched" detail=""/>
            <Duration value="0.0" units="hours" isDefinedDuration="false"/>
            <Billing rate="0.0" type="none"/>
            <Expensing expenseCode=""/>
            <Authorization>
                <ServiceGroupReview>
                    <AuthRoles/>
                    <Escalations/>
                </ServiceGroupReview>
                <ServiceGroupAuthorization>
                    <AuthRoles/>
                    <Escalations/>
                </ServiceGroupAuthorization>
            </Authorization>
            <Presentation>
                <Description>
                    <Icon documentId="0"/>
                </Description>
               <SectionOne isEnabled="false"/>
                <SectionTwo isEnabled="false"/>
                <SectionThree isEnabled="false"/>
            </Presentation>
            <PositionNames>
                <PositionName name="Author" type="Service" serviceDefinitionName="Service
A" serviceDefinitionGuid="EBC80F54-01E3-45C1-8DF2-1C35C1165838" objectType="Person"
instanceName="admin"/>
            </PositionNames>
            <Categories>
                <ConsumerServicesCategories/>
                <ServicePortfoliosCategories/>
            </Categories>
            <AssociatedKeywords/>
            <AssociatedObjectives/>
            <Pricing price="0.0" pricingSchema="fixedprice" pricingSchemaString="Fixed
Price" priceDisplaySchema="" priceDisplaySchemaString="">
                <Description>
                    <Text>
                        <![CDATA[]]>
                    </Text>
                </Description>
            </Pricing>
            <Costing>
                <Costs/>
           </Costing>
            <ServiceLevel/>
            <Bundling canBeBundled="true" isABundle="false">
                <ServiceContainsServices/>
                <ServiceIsContainedByServices/>
            </Bundling>
            <ServicePreRequisites/>
            <ServiceAccessories/>
            <Manufacturer name="" partNumber=""/>
            <Supplier name="" type="" partNumber=""/>
            <Plans>
                <Plan>
```

```
                    <PlanAttributes type="delivery" name="PrimaryDeliveryPlan"
subtaskFlow="sequential" maxDepth="1" hoursPerDay="8.0" isAutomaticStart="true">
                        <PlanSubject>
                            <![CDATA[<s ID="11"><p>#Name#</p></s>]]>
                        </PlanSubject>
                        <PlanMonitorTask>
                            <PlanMonitorWorkflowRoles>
                                <PlanMonitorWorkflowRole assignmentMethod="none"
expression=""/>
                            </PlanMonitorWorkflowRoles>
                            <Notifications>
                                <Notification eventType="plantaskcanceled"
emailTemplate=""/>
                            </Notifications>
                        </PlanMonitorTask>
                    </PlanAttributes>
                    <Tasks/>
                    <Escalations/>
                </Plan>
            </Plans>
            <DictionaryFormAccessPolicies/>
            <ServiceForms/>
            <RelatedEntities>
                <Forms/>
                <ServiceGroups>
                    <ServiceGroupName name="0 Service Icons"
guid="956A6E7B-1FB6-4B49-B258-C58F5AC727D3"/>
                </ServiceGroups>
                <Categories/>
                <EmailTemplates/>
                <Objectives/>
                <ServiceDefinitions/>
                <Keywords/>
                <Persons/>
                <OrganizationalUnits/>
                <Queues/>
                <Agents/>
            </RelatedEntities>
            <PermissionsAssignment>
                <OperationsWeArePermitted/>
                <OperationsTheyArePermitted>
                    <Operation name="service_change_rights"
displayName="service_change_rights">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_delivery"
displayName="service_change_delivery">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_forms"
displayName="service_change_forms">
                        <Objects/>
                    </Operation>
                    <Operation name="service_change_presentation"
displayName="service_change_presentation">
                        <Objects/>
                    </Operation>
                    <Operation name="service_order_service"
displayName="service_order_service">
                        <Objects/>
                    </Operation>
                </OperationsTheyArePermitted>
            </PermissionsAssignment>
            <Images/>
```

```
            <ServiceExtensions/>
            <BaseTemplateID>
                <![CDATA[0]]>
            </BaseTemplateID>
        </ServiceDefinition>
    </ServiceDefinitions>
</rex>
```

Configuring Rate Limits for REST API Requests

The Prime Service Catalog administrator globally configures the number of requests allowed and the time-slot (in seconds) for the requests to be allowed through the rate limiter.

The requests (from an external application or within the Prime Service Catalog UI) are accepted as per the configuration and are rejected if they exceed the configuration limit.

When the requests are rejected, the HTTP response code for rate limiting - *429 Too Many Requests (RFC 6585)*, is returned to the requesting user.

The following global default values are set by the administrator for the rate limiting feature:

- **enabled**: This flag is set to **true** or **false** to enable or disable rate limiting for REST calls (global setting that is applied for API or UI-based nsAPI REST calls). The flag is disabled by default, however, you can enable ratelimiting by configuring the nsApiRateLimit.json file. The setting for the flag is applied equally to global and all of the overrides. In addition, separate rate limits can be defined for nsAPI REST calls from UI and from external applications.
- **nsApiRateLimit**: specifies the maximum number of calls allowed.
- **nsApiRateIntervalSecs**: specifies the timeslot (in seconds) within which the requests set by **nsApiRateLimit** can be allowed.

In addition, administrator can override the global (default) values using the following qualifiers (Specific URLs, one or more methods will be overridden):

Note You cannot override the time slot. Each override uses the global setting.

- **url**: The nsAPI Rest URIs are either complete URLs or wild card based (*). If there is no wild card, an absolute match is assumed between the input uri and the configured. Any query parameters in the configuration will be omitted by the rate limiter.
- **method**: The HTTP methods is a string literal array which contains one or many of ["GET", "PUT", "POST", "DELETE"].
- **nsApiRateLimit**: specifies maximum number of calls allowed. This overrides the similar global setting for this URI only.

Configure rate limit as per the following JSON structure:

Note JSON validation is a manual process. When you save the configuration file nsAPIRateLimit.json,the application does not validate JSON, and would not work in case of an invalid JSON.

```
{
    "api": {
        "config": {
            "enabled": "false",
            "nsApiRateLimit": 2,
            "nsApiRateIntervalSecs": 10,
```

```
                "overrides": [
                    {
                        "url": "/nsapi/serviceitem/SiServItem*",
                        "method": [
                            "GET"
                        ],
                        "nsApiRateLimit": 5
                    },
                    {
                        "url": "/nsapi/serviceitem/*",
                        "method": [
                            "GET"
                        ],
                        "nsApiRateLimit": 3
                    },
                    {
                        "url": "/nsapi/serviceitem/*",
                        "method": [
                            "POST",
                            "PUT"
                        ],
                        "nsApiRateLimit": 3
                    },
                    {
                        "url": "/nsapi/transaction/authorizations/*",
                        "method": [
                            "GET",
                            "PUT",
                            "POST",
                            "DELETE"
                        ],
                        "nsApiRateLimit": 8
                    },
                    {
                        "url":
"/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self| Status=Approved",
                        "method": [
                            "GET",
                            "PUT",
                            "POST",
                            "DELETE"
                        ],
                        "nsApiRateLimit": 3
                    },
                    {
                        "url": "/nsapi/definition/categories",
                        "method": [
                            "GET",
                            "PUT",
                            "POST",
                            "DELETE"
                        ],
                        "nsApiRateLimit": 5
                    }
                ]
            }
        },
        "ui": {
            "config": {
                "enabled": "false",
                "nsApiRateLimit": 4,
                "nsApiRateIntervalSecs": 10,
                "overrides": [
                    {
```

```
                    "url": "/nsapi/transaction/requisitions/*",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/transaction/authorizations/*",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 3
                },
                {
                    "url": "/nsapi/definition/categories",
                    "method": [
                        "GET",
                        "PUT",
                        "POST",
                        "DELETE"
                    ],
                    "nsApiRateLimit": 5
                }
            ]
        }
    }
}
```

Error Messages

Different HTTP response codes are returned by nsAPI depending on the nature of the exceptions:

- HTTP Status code **400** Mandatory Field Missing Error / Redirect URL or EntityID from metadata already exists / Duplicate IDP configuration name – Field missing error or the metadata information already exists or duplicate of IDP configuration name.
- HTTP Status code **401** (Unauthorized) and XML error response message "**User does not have proper authentication**." or messages could pertain to password policy violation – Invalid or no authentication parameters.
- HTTP Status code **404** (Not Found) and XML error response message "**Requested resource could not be found**." – Data could not be fetched for the specified parameter/URL values.

  EXAMPLES:

  ```
  nsapi/directory/people/id/-1
  nsapi/directory/people/id/foo
  nsapi/directory/people/id/1000 (there is no person with id = 1000)
  nsapi/directory/people/name/<non existent person>
  nsapi/directory/people/idxyz/1
  ```

- HTTP Status code **403** (Forbidden) and XML error response message "**The user does not have sufficient permissions to perform the operation this object**." – Data could not be fetched because the user does not have sufficient permissions to perform the specified operation on the object.

- HTTP Status **500** (Internal Error) and XML error response message "**Internal Error: Invalid parameter values specified or unexpected error**." – Incorrect parameters, any other exceptions that occur within nsAPI or any other general server error.

  EXAMPLES:

  ```
  nsapi/directory/people?startRow=5000 (non-existent 5000 row)
  nsapi/directory/people?sortBy=wrongColumn&sortDir=Asc (unsupported column)
  nsapi/directory/people?recordSize=-1 (negative or zero value for recordSize)
  ```

- Http Status **422** (Unprocessable Entity) and XML validation error response message for post/update operations – The request data does not have values for mandatory fields or contains invalid values for the fields.

nsAPI returns an error message if no result is found for the filters provided or if invalid filter criteria is provided. For example:

```
<nsapi-response>
<status-code>failed</status-code>
<status-message>Service item custom_sit201 does not exist, none of the service items were
updated. </status-message>
</nsapi-response>
```

nsAPI throws an NSAPIException from Java for all exceptions encountered when executing methods in nsAPI.

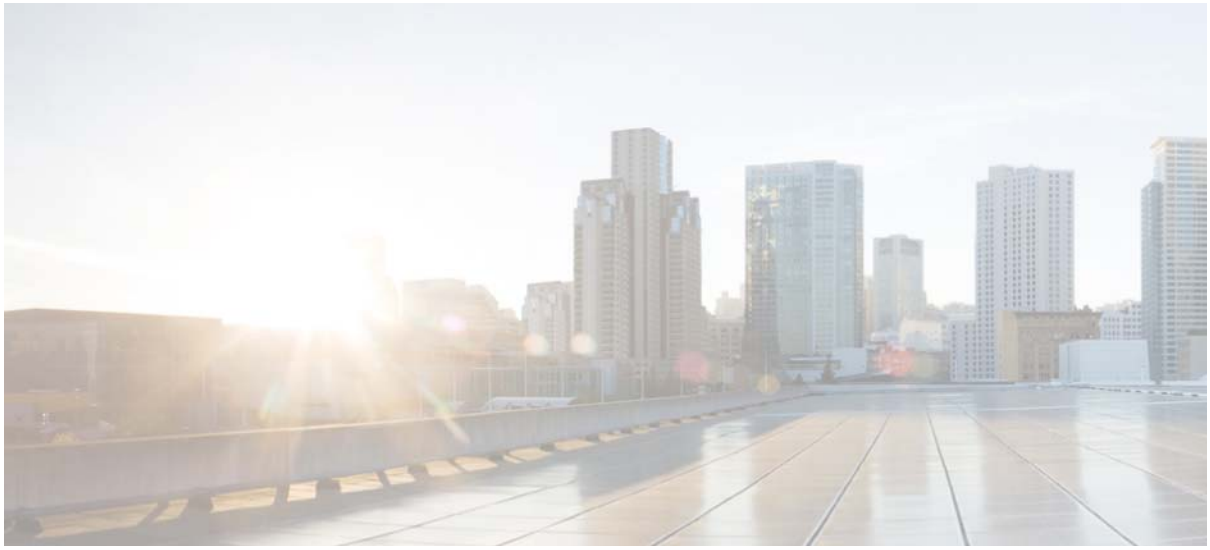
Summary of Supported Operations

The chart below provides a summary of operations supported for the different entity types

Reference Table

Entity Name	Get by Id	Get by Name	Get all	Wildcard Name Search	Sorting	Paging	Convert REST Filters to Java Client methods	Update	Filters	Nested Entities
Authentication										
Authentication - Login/Logout					X		X			
Definitional Data										
Categories	X	X	X	X	X	X	X		Catalog Type	Subcategories, Included Services, Included Offerings
Services	X	X	X	X	X	X	X		Categoryid, Category Name, Keyword	Categories, Keywords, Included Services
Agreement	X	X	X	X	X	X	X		Name	
Agents	X	X	X	X	X	X	X		Name	
Transactional Data										
Requisition	X		X		X	X	X		View Name, Status	
Requistion Entry	X		X		X	X				
Authorization	X		X		X	X	X		View Name, Status	
Task	X		X		X	X	X		View Name	
Task – Tasks for a specific requistion entry			X		X	X	X		Task Type, Status (Skipped)	Included Service Tasks
Task – Milestones for a specific requisition			X							
Task – Actions								X		
Directory Data										
Organizational Units	X	X	X	X	X	X	X		OU Type	
Person	X	X	X	X	X	X	X	X	OU Name, Group Name, Role Name	OU, Group, Roles, Address, Contact, Preferences, Delegates
Groups	X	X	X	X	X	X	X		Name	

Entity Name	Get by Id	Get by Name	Get all	Wildcard Name Search	Sorting	Paging	Convert REST Filters to Java Client methods	Update	Filters	Nested Entities
Accounts	X	X	X	X	X	X	X		Name	
Service Item Data										
All Service Items			X		X	X			Any column	
Service Items			X		X	X	X		Any column	
Standards			X		X	X	X		Any column	
Service Catalog Data										
Custom Content		X	X	X	X	X	X		Any Column	



PART 2

Southbound Integration

Integrating with AMQP

Overview

Advanced Message Queuing Protocol (AMQP) is an open standard for passing business messages between applications or organizations. You can use Service Designer module to define an AMQP task. An AMQP task publishes the service request to an external system (Process Orchestrator) via a message broker.

The AMQP sends the service request to an exchange, which accepts the message from Service Catalog and routes them to message queues. RabbitMQ is an open source message broker software that implements the AMQP standard. An external system (Process Orchestrator) will access RabbitMQ with the required APIs to retrieve the messages and process them.

A new task type called Queue Service Request is available in the Service Designer module under the Plan tab to publish the service request data to a message broker. For more information on how to publish data using AMQP task, see Cisco Prime Service Catalog Designer Guide.

Message Queue

An exchange accepts messages from a producer application and routes them to message queues. The exchange that will be created for the pre, post, and the main AMQP tasks will be all of 'fanout' type with a default queue created for each of them and bound to each of them. The names of the default queues would be <topic-name>_queue. The topic name is the name that the user enters in the **Service Designer > Plan** tab for the 'Queue Service Request' task.

Note To prevent poodle attack, Prime Service Catalog integration with RabbitMQ server supports SSL protocol with TLSv1.2 version only. If TLSv 1.2 is not specified, then clients cannot connect to RabbitMQ server for consumption of messages and connections fail with this exception: javax.net.ssl.SSLException: Received fatal alert: protocol_version

Following is a sample code to create an exchange and queue and how to consume a message.

```
package amqpProject;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.QueueingConsumer;
```

```
public class SampleProcess {

  public static void main(String[] argv) throws Exception {
    String exchangeName = "testExchange";
    String queueName = exchangeName+"_queue";
    String brokerIpAddress = "10.142.10.77";
    String userName = "admin";
    String password = "cisco123";
    ConnectionFactory factory = new ConnectionFactory();
    factory.useSslProtocol("TLSv1.2"); !--Included to prevent Poodle attack
    factory.setHost(brokerIpAddress);
    factory.setUsername(userName);
    factory.setPassword(password);

    Connection connection = factory.newConnection();

    Channel channel = connection.createChannel();
    channel.exchangeDeclare(exchangeName, "fanout", true, false, null);

    channel.queueDeclare(queueName, true, false, false, null);
    channel.queueBind(queueName, exchangeName, "");

    QueueingConsumer consumer = new QueueingConsumer(channel);
    channel.basicConsume(queueName, true, consumer);
    while (true) {

      QueueingConsumer.Delivery delivery = consumer.nextDelivery();
      String message = new String(delivery.getBody());
      System.out.println(" [x] Received from "+queueName+":-");
      System.out.println(message);

    }
  }
}
```

REST-based nsAPIs

There is a set of REST API for returning the message broker details for a service and its tasks. The signature and the response returned by it are given below. The input required is the service name that will be accepted as a path parameter. All the AMQP tasks for that service with the exchange details will be returned to the caller.

Given below is the sample response for a Service specific API:

http://localhost:8088/RequestCenter/nsapi/messagebroker/service/<service name>

{"tasks":[{"name":"PRE:amqpTask1","connectionIdentifier":"am1","messageFormat":"JSON","exc
hangeName":"AQAB_Pre","payloadType":"All Message Details
(large)","isAutoComplete":"true","transformationType":"JOLT","outboundTransformationName":
"","inboundTransformationName":""},{"name":"amqpTask1","connectionIdentifier":"am1","messa
geFormat":"JSON","exchangeName":"post_exchange_0307","payloadType":"All Message Details
(large)","isAutoComplete":"true","transformationType":"JOLT","outboundTransformationName":
"","inboundTransformationName":""},{"name":"POST:amqpTask1","connectionIdentifier":"am1","
messageFormat":"JSON","exchangeName":"AQAB_Post","payloadType":"All Message Details
(large)","isAutoComplete":"false","transformationType":"JOLT","outboundTransformationName"
:"","inboundTransformationName":""}]}

Overview API

The Overview API gathers all the exchanges that are configured in the Service Designer module and makes a call to find out what are the exchanges and queues that are already created in the RabbitMQ server. The Overview API performs a lookup on AMQP server to get information on queues and the output is sent across in JSON format.

Using Overview API and noCache parameter:

- When you specify RabbitMQ information through the UI and use the Overview API without noCache parameter, the information is retrieved from both cache and database.
- However, when you directly access database to insert RabbitMQ information, you must use Overview API with noCache =1 to ensure that the latest information is fetched.

The sample response from the RabbitMQ server is as follows:

http://localhost:8088/RequestCenter/nsapi/messagebroker/overview

{"connections":[{"connectionIdentifier":"am1","host":"10.78.0.247","port":5671},{"conne
ctionIdentifier":"am2","host":"10.78.0.247","port":5671},{"connectionIdentifier":"am3",
"host":"10.78.0.247","port":5671},{"connectionIdentifier":"AM4","host":"10.78.0.247","p
ort":5671},{"connectionIdentifier":"am5","host":"10.78.0.247","port":5671}]}

**http://localhost:8080/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=a
m1**

{"rabbitmq_version":"3.6.1","username":"admin","password":"UhP0zDGRoC2R7isv9qCQ6A==","i
pAddress":"10.78.0.247","recoveryInterval":300000,"vhost":"/","inboundQueue":"psc_inbou
nd_queue","authorization_key":"owL7ViRfE4Sce0aG1jSzZInkIVw5CsM7acQ5rlswpXzF/kForoUjlfrV
VUOuA+CqSFSTuJlLQ5GRUMmzzxbV2xtrMvhGed6x1WU08MbDLKydSSY5UQvoAS7aZ0dROdXvF+G4uD4nAlQ6HSI
c6dBct3M+dDJcm02z9OshvMaCmkvZa380B8/MbbBhu5Q3FntzWkAVY/FobU9gvlidoDt1Ty0CmAgfvPbP6joLXY
aTPHjaqjYaBaX2Y4m+1V7Wm3Rb+oLpHZkCVm7Pr1zlLByPs6d+qaQShIfr0yeXXbZrQNh3s8qHl+YbYFRtNin/J
EeLa6pY5J19m6zUV78n2BstRQ==","message-time-to-live":"300000","useSSL":true,"port":"5671
","exchanges":[{"name":"exchange2","vhost":"/","type":"fanout","created":"false"},{"nam
e":"HeatStack","vhost":"/","type":"fanout","created":"true"},{"name":"exchange3","vhost
":"/","type":"fanout","created":"false"}],"queues":[{"name":"exchange2_queue","vhost":"
/","created":"false"},{"name":"HeatStack_queue","vhost":"/","created":"true"},{"name":"
exchange3_queue","vhost":"/","created":"false"}],"certificate":"-----BEGIN
CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVRlc3RDQ
TAeFw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG9zdG5hbWUpMQ8wDQYD\n
VQQKDAZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq8v2yJT7tv+nOwFSo\nEE1c0oVg3
skd86JN7jVJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw+a+0ext2WRHgQLTMPvVO\nA86jwuPd/bhUxXg8je
EfE4V/1Seci9Xz+5VxqCybCNOzJQ12/vLXvIJK43U/+1GdXnpWxFaF0yd0\nht3iUy6mfUAHfNMI2SOfJwbbdUa
MyD0/Krsiu+X+vFQBUDmM7Y0priItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvk
TET3tIVsKDi1Fn9IdKQuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCM
AAwCwYDVR0PBAQDAgUgMBMGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4kuu/d
pLeFWNUhb5Uyk6AqbTRAyD\nlq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94gq8luSem1Z8lQz

UCOCE6rsaznrw\nm9jJO7gXA5SSmy7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEPREqnshs6f6bQSm8tzSNETy3
OyHL\npUo7YVTBCfJMQ/e2nZxCJSDuEL6QaIL4kmEYeu8j/1RplBAofMkDfDe+yMx2MJYl+MopVggGexpa\nMqy
bCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNVenCpGCtgGIcupRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----E
ND CERTIFICATE--"}

Encrypt Credentials using Public Key GUID

Prime Service Catalog supports a secure way to return AMQP credentials, using the GUID of the Public Key passed to the nsAPI.

If the external system Public Key GUID is passed as a Query parameter, then credentials are encrypted (using PO Secure String Format) in the response with the Public Key associated to the GUID

If the external system Public Key GUID is not passed, encrypted string in DB is returned as is (note: external system cannot decrypt this)

Encrypted sensitive data is returned in the following AMQP overview APIs:

http://localhost:8088/RequestCenter/nsapi/messagebroker/overview?publicKeyGUID=...

Encryption is done using the public key configured at the connection level.

Generating AuthorizationKey API

For Service Item operations like create, update, and delete, Authorization Key is of more of importance than channel-id. But channel-id is also required in the message for uniquely identifying the external task. Authorization key generated using the below nsAPI is unique for a user. Authorization key is same for all the AMQP connections for a particular user even though its value seems to change every time the above nsAPI is called. This key is used for authorizing the user and to access various modules based on the permissions the user is granted.

Authorization key can be obtained from the following nsAPI:

http://<ServerURL>
/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=<particular amqp connection identifier>

{"rabbitmq_version":"3.6.1","username":"admin","password":"UhP0zDGRoC2R7isv9qCQ6A==","ipAd
dress":"10.78.0.247","recoveryInterval":300000,"vhost":"/","inboundQueue":"psc_inbound_que
ue","authorization_key":"owL7ViRfE4Sce0aG1jSzZInkIVw5CsM7acQ5rlswpXzF/kForoUjlfrVVUOuA+CqS
FSTuJlLQ5GRUMmzzxbV2xtrMvhGed6x1WU08MbDLKydSSY5UQvoAS7aZ0dROdXvF+G4uD4nAlQ6HSIc6dBct3M+dDJ
cm02z9OshvMaCmkvZa380B8/MbbBhu5Q3FntzWkAVY/FobU9gvlidoDt1Ty0CmAgfvPbP6joLXYaTPHjaqjYaBaX2Y
4m+1V7Wm3Rb+oLpHZkCVm7Pr1zlLByPs6d+qaQShIfr0yeXXbZrQNh3s8qHl+YbYFRtNin/JEeLa6pY5J19m6zUV78
n2BstRQ==","message-time-to-live":"300000","useSSL":true,"port":"5671","exchanges":[{"name
":"exchange2","vhost":"/","type":"fanout","created":"false"},{"name":"HeatStack","vhost":"
/","type":"fanout","created":"true"},{"name":"exchange3","vhost":"/","type":"fanout","crea
ted":"false"}],"queues":[{"name":"exchange2_queue","vhost":"/","created":"false"},{"name":
"HeatStack_queue","vhost":"/","created":"true"},{"name":"exchange3_queue","vhost":"/","cre
ated":"false"}],"certificate":"-----BEGIN
CERTIFICATE-----\nMIIC5DCCAcygAwIBAgIBATANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVRlc3RDQTAe
Fw0x\nNjA2MTExMTUwMTJaFw0xNzA2MTExMTUwMTJaMCcxFDASBgNVBAMMCyQoaG9zdG5hbWUpMQ8wDQYD\nVQQKDA
ZzZXJ2ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq8v2yJT7tv+nOwFSo\nEE1c0oVg3skd86JN7j
VJaz/mMOyJjDmf1147iUwZPMTTCB34ovYUXFkw+a+0ext2WRHgQLTMPvVO\nA86jwuPd/bhUxXg8jeEfE4V/1Seci9
Xz+5VxqCybCNOzJQ12/vLXvIJK43U/+1GdXnpWxFaF0yd0\nht3iUy6mfUAHfNMI2SOfJwbbdUaMyD0/Krsiu+X+vF
QBUDmM7Y0priItiDVDq7rxug2UOPACzzMG\n5yJ5aJjNLSlJRwKKst/jjvesqHIgWNo0qKvkTET3tIVsKDi1Fn9IdK
QuuoI1n225+58cWSANmZ5M\n4BdSIf4z6QRuKliBRi55AgMBAAGjLzAtMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgUgMB
MGA1UdJQQM\nMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQC4L9fk4kuu/dpLeFWNUhb5Uyk6AqbTRAyD\n
lq1m11E09EhmTH7cnoFsz0ELBDppASIUADSb9jmUeNKJtjW94gq8luSem1Z8lQzUCOCE6rsaznrw\nm9jJO7gXA5SS

my7PgdokdhbeTzlSYA3kkkR5ZE8M403Qv5cEPREqnshs6f6bQSm8tzSNETy3OyHL\npUo7YVTBCfJMQ/e2nZxCJSDu
EL6QaIL4kmEYeu8j/1RplBAofMkDfDe+yMx2MJYl+MopVggGexpa\nMqybCchrDTJ/8sI/R18Ld80TQ2Km70sQqvNV
enCpGCtgGIcupRWaAOpsQH0PNaUK+lcwYRIvf0VS\n09QE\n-----END CERTIFICATE--"}

Transforming JSON Using JOLT Work flow

From Prime Service catalog 12.0 onwards, inbound transformations are also supported. However, transformations are not supported for Service Item operations. For inbound messages transformation types supported are XSL and JOLT, and for outbound XSL, JOLT, and FTL are supported.

The below is the high-level work flow for creating a transformations are as follows:

Step 1 Add an AMQP connection from **Administration > Manage Connections > AMQP**. Select the For more information, see section Managing AMQP Connections in the *Cisco Prime Service Catalog Administration and Operation Guide*.

Step 2 Create outbound and inbound transformations in JOLT format. For details see Managing Transformations, page 5-18.

Step 3 In the service designer configure AMQP task for the service request. For details see section Configuring AMQP Tasks for Publishing Service Request to an External System of *Cisco Prime Service Catalog Designer Guide*.

Points to remember:

- You can define default values for the following which can be overridden in AMQP task parameter popup page.
  - "Public Key
  - "Message Type i.e the message format (XML/JSON) in which outbound and inbound message processing would happen by default for the particular connection.
- Message type can be set at the AMQP connection level or at the task level, but the task level setting takes precedence.
- Different transformations are fetched based on the transformation type selected i.e., if XML is selected only XSL transformations will be available for outbound transformation, inbound transformation selection and for JSON it can be either JOLT or FTL transformations.
- FTL transformation is supported only for outbound messages.

Sample JSON Transformation Using JOLT

JSON inbound AMQP input:

```
{
    "message": {
        "updateData": {
            "dataValue": [
```

```
                {
                    "namePO": "TextField.Text",
                    "valuePO": [
                        "QoE"
                    ],
                    "multiValuedPO": false
                }
            ]
        },
        "addCommentsPO": {
            "comment": [
                "test comment 1",
                "test comment 2",
                "test comment 3"
            ]
        },
        "takeAction": {
            "actionPO": "DONE"
        },
        "channelId": "51515F28-E36A-478C-8773-E35B215CF36C"
    }
}
```

JOLT Transformation spec:

```
[
  {
    "operation": "shift",
    "spec": {
      "message": {
        "updateData": {
          "dataValue": {
            "*": {
              "namePO": "message.updateData.dataValue.[&1].name",
              "valuePO": "message.updateData.dataValue.[&1].value",
              "multiValuedPO": "message.updateData.dataValue.[&1].multiValued"
            }
          }
        },
        "*": "&1.&0",
        "channelId": "message.channelId"
      }
    }
  },
  {
    "operation": "shift",
    "spec": {
      "message": {
        "takeAction": {
          "actionPO": "message.&1.action"
        },
        "*": "&1.&0",
        "channelId": "message.channelId"
      }
    }
  },
  {
    "operation": "shift",
    "spec": {
      "message": {
        "addCommentsPO": "message.addComments",
        "*": "&1.&0",
        "channelId": "message.channelId"
```

```
      }
    }
  }
]
```

JSON Output:

Below is the output generated by Prime Service Catalog inbound AMQP code based on the above transformation and input, which is finally processed.

```
{
  "message" : {
    "addComments" : {
      "comment" : [ "test comment 1", "test comment 2", "test comment 3" ]
    },
    "channelId" : "51515F28-E36A-478C-8773-E35B215CF36C",
    "takeAction" : {
      "action" : "DONE"
    },
    "updateData" : {
      "dataValue" : [ {
        "multiValued" : false,
        "name" : "TextField.Text",
        "value" : [ "QoE" ]
      } ]
    }
  }
}
```

Sample AMQP Inbound XML

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<update-data>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a_newer</value>
</data-value>
</update-data>
</message>

<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<add-comments>
<comment>testing new comments</comment>
</add-comments>
</message>


<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<take-action action="done">
</take-action>
</message>
```

Sample AMQP Inbound JSON

```
{
  "message" : {
    "addComments" : {
      "comment" : [ "test comment 1", "test comment 2", "test comment 3" ]
    },
```

```
    "channelId" : "51515F28-E36A-478C-8773-E35B215CF36C",
    "takeAction" : {
      "action" : "DONE"
    },
    "updateData" : {
      "dataValue" : [ {
        "multiValued" : false,
        "name" : "TextField.Text",
        "value" : [ "QoE" ]
      } ]
    }
  }
}
```

Inbound Message

Two types of operations are supported for inbound messages from the third-party system—requisition operations and service item operations.

The most important element within the nsXML is the channel-id, an ID that uniquely identifies the external task. This ID is provided to the third-party system and needs to appear in their response if the corresponding data update is to be successfully applied by the business engine.

Requisition Operations for AMQP

Requisition operations supported for AMQP are similar to Service Link requisition operations. However, in AMQP inbound messages supports XML and JSON formats. For details on each of the operations, see section Inbound nsXML Message, page 5-25.

take-action

The take-action operation marks the delivery task as completed.

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<take-action action="done">
</take-action>
</message>
```

update-data

The dictionary fields data in the requisition are updated with the new values.

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<update-data>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a_newer</value>
</data-value>
</update-d n ata>
</message>
```

add-comment

An add-comments message is used to add comments to the System Comments section of the requisition.

```
<message channel-id="30FCC75D-83A0-4DD9-E050-007F01019778">
<add-comments>
<comment>testing new comments</comment>
```

```
</add-comments>
</message>
```

Service Item Operations for AMQP

Note Transformations are not supported for Service Item operations.

For the service item operations, authorization key must be generated and included in the messages. Use the below nsAPI to generate the authorization key:

```
http://<ServerURL>
/RequestCenter/nsapi/messagebroker/overview?connectionIdentifier=<particular amqp
connection identifier>
```

For more details on this API, see section Generating AuthorizationKey API, page 4-4. Below are the sample Service Item operations for AMQP.

create

In create messages, the attribute values are added to the service items.

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
<create>
<serviceitem>
<name>AnandServiceItem</name>
<serviceItemData>
<serviceItemAttribute name="Name">Hannah</serviceItemAttribute>
<serviceItemAttribute name="DOB">1995-01-19</serviceItemAttribute>
<serviceItemAttribute name="Age">25</serviceItemAttribute>
</serviceItemData>
</serviceitem>
</create>
</message>
```

update

In update messages, omitting a service item attribute results in no change to the attribute value. When an attribute is explicitly specified in the message but contains no value, the value of the attribute for the service item is set to blank for text fields and zero for numeric fields.

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
 <update>
   <serviceitem>
   <name>AnandServiceItem</name>
   <serviceItemData>
   <serviceItemAttribute name="Name">Anand2update</serviceItemAttribute>
   <serviceItemAttribute name="RAM">Primarymemory3</serviceItemAttribute>
   <serviceItemAttribute name="MemoryInt">759001</serviceItemAttribute>
   <serviceItemAttribute name="Model512">Lenovo T440 updated</serviceItemAttribute>
   <serviceItemAttribute name="Money">80000</serviceItemAttribute>
   <serviceItemAttribute name="ManufTime">2016-05-01 16:45</serviceItemAttribute>
   </serviceItemData>
```

```
   </serviceitem>
  </update>
```

delete

Delete service item requests require only the names for the service item type and instance. Additional service item attribute and subscription information is ignored.

```
<message channel-id="A984F860-DBE7-48EB-B3A2-F6A0159B093F"
authorization-key="liDBm1NeRVbVmwbBmAd+cca0/Z8jf863aYOKl3QXSWcYwn5aGuPxn09CjDzv5EKR7/CcJx5
+2gu1MhUopAuiX3WwjX/DP1Xw3nmFLMz6dmaiiq0+5v4XOKmNqf1J3GvBJYjTxAkH1VbCQ2y1fNxE8/cP5wyd3mU6M
LlD9tjc/Iro950gLlTq+9C2/QMis6ya52O2D8F652jnHnWbZrDf6zPdOSlFpCKkg05YSVpqifGgqjEh3RTyPs9w0Qc
NcxWcEqD1vvBMNW0VAL/YaW+MNmoUnpglR0cUeOJ3WFr8/1l/uyNyhfvAYfDUznVCNfVDtkMzCWhA4eH25XQUtmKwG
w==">
   <delete>
      <serviceitem>
      <name>ServiceItem</name>
      <serviceItemData>
      <serviceItemAttribute name="Name">Anand3</serviceItemAttribute>
      </serviceItemData>
   </serviceitem>
   </delete>
</message>
```

hierarchy

In hierarchy messages, the attribute values are nested in the service items.

Outbound Message

The message format published is in the nsXML, JSON, or FTL formats similar to the format that is currently used by Service Link.

```
{
        "message": {
                "create": {
                        "hierarchy": true,
                        "serviceitem": [ {
                                "name": "Computer",
                                "serviceItemData": [{
                                        "Name": "Dell-HP Computer",
                                        "OS": "Linux 7",
                                        "RAM": "7878",
                                        "HD": "520",
                                        "Mouse": {
                                                "Name": "HPE"
                                        },
                                        "Network": {
                                                "Name": "My Mac Network",
                                                "IPAddress": "3.3.3.3",
```

```
                                                    "MACAddress": "12.12.12.12",
                                                    "AdapterType": "MAC"
                                                }
                                            }]
                                    }]
                            },
                            "channelId": "6A55AF3D-4EEC-402C-9055-CB79081C138F"
                    }
            }
```

Sample XML Outbound Message

The sample nsXml Data Structure is given below.

- Sample message structure for Data; No Service Details(default; small)

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="F203ACC7-E6CA-A2EA-E040-007F0101140E"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<task-started task-type="task">
<task>
<actual-duration>0.0</actual-duration>
<completed-date/>
<context-id>69</context-id>
<context-type>Requisition Entry</context-type>
<due-date>2014-04-25 20:00:00</due-date>
<effort>10.0</effort>
<estimated-date/>
<expected-duration>10.0</expected-duration>
<flag-id>0</flag-id>
<is-sharable>true</is-sharable>
<is-shared>true</is-shared>
<next-action-id>2</next-action-id>
<performer-actual-duration>0.0</performer-actual-duration>
<priority>2</priority>
<scheduled-start-date>2014-04-24 18:00:00</scheduled-start-date>
<start-date>2014-04-23 14:11:13</start-date>
<state-id>2</state-id>
<subject>queueServiceRequest1</subject>
<task-id>266</task-id>
</task>
<requisition>
<services>0</services>
<actual-cost>0.0</actual-cost>
<actual-duration>0.0</actual-duration>
<closed-on/>
<customer>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
```

```
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</customer>
<due-on>2014-04-25 20:00:00</due-on>
<expected-cost>0.0</expected-cost>
<expected-duration>0.0</expected-duration>
<external>false</external>
<initiator>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</initiator>
<organizational-unit>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</organizational-unit>
<requisition-entry>
<closed-date/>
<data-values>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a</value>
</data-value>
<data-value multi-valued="false">
<name>amqpDict1.field2</name>
<value>b</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field3</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgAC/EsDiHp1ERsOpdKSKdrtgMjFWcVoO96aCFSkCgwa2tkBo
vKoOHzaillNiJ47+aY8zCWKwd17tCjmMLEkeQlTvrDvIHR/DT1iWSmN08DI9+Ns7hY3A/g6ijUoM
gcuMczH+5F/pGtgupLZF/L7FwOwu4VcKVWM/2N5tuXGz+1aHTRAAAAByQXr/yD/75Ysy57LnQLxc
</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field4</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgABReynDp5AdBgRi5ivhS05Unv98BgWgxc5YNcZrCihhhH/1
rzZqhjiIjAoRelIjADDb3IQP72armXsLRTvh0/fusd4jLdVIm4q1s+GaSTt6F3oqQeZ4RLhVUopo
p2zNAxmjaGj629C8gWREes3Z8EjDvXg5K1YVi90fXJV1jDoADhAAAABP1hct5TNV2d8R1cN/qDtK
</value>
</data-value>
</data-values>
<due-date>2014-04-25 20:00:00</due-date>
<item-number>1</item-number>
<price-per-unit>0.0</price-per-unit>
```

```
<priced>true</priced>
<quantity>1</quantity>
<rejected>false</rejected>
<rejected-date/>
<requisition-entry-id>69</requisition-entry-id>
<revision-number>105</revision-number>
<service>
<estimated-cost>0.0</estimated-cost>
<name>queueService1</name>
<parameters>
<default-duration>0.0</default-duration>
<priority>2</priority>
<start-date/>
<start-mode>0</start-mode>
</parameters>
<pricing-schema>0</pricing-schema>
<quantity>1</quantity>
<service-id>2</service-id>
<version>105</version>
<standard-duration>0.0</standard-duration>
</service>
<start-after/>
<start-date>2014-04-23 14:10:48</start-date>
<start-mode>0</start-mode>
<status>1</status>
</requisition-entry>
<requisition-id>64</requisition-id>
<started-on>2014-04-23 14:10:47</started-on>
<status>1</status>
</requisition>
<context>
<requisitionentryref itemnumber="1"/>
</context>
</task-started>
</message>
```

- Sample message structure with secure strings in it.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="F203ACC7-E6CA-A2EA-E040-007F0101140E"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<task-started task-type="task">
<task>
<actual-duration>0.0</actual-duration>
<completed-date/>
<context-id>69</context-id>
<context-type>Requisition Entry</context-type>
<due-date>2014-04-25 20:00:00</due-date>
<effort>10.0</effort>
<estimated-date/>
<expected-duration>10.0</expected-duration>
<flag-id>0</flag-id>
<is-sharable>true</is-sharable>
<is-shared>true</is-shared>
<next-action-id>2</next-action-id>
<performer-actual-duration>0.0</performer-actual-duration>
<priority>2</priority>
<scheduled-start-date>2014-04-24 18:00:00</scheduled-start-date>
<start-date>2014-04-23 14:11:13</start-date>
<state-id>2</state-id>
<subject>queueServiceRequest1</subject>
<task-id>266</task-id>
</task>
<requisition>
```

```
<services>0</services>
<actual-cost>0.0</actual-cost>
<actual-duration>0.0</actual-duration>
<closed-on/>
<customer>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</customer>
<due-on>2014-04-25 20:00:00</due-on>
<expected-cost>0.0</expected-cost>
<expected-duration>0.0</expected-duration>
<external>false</external>
<initiator>
<company-address/>
<email>internal@newscale.com</email>
<fax/>
<first-name>admin</first-name>
<home-ou>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</home-ou>
<home-phone/>
<last-name>admin</last-name>
<login-name>admin</login-name>
<person-id>1</person-id>
<personal-address/>
<supervisor-name/>
<timezone>Pacific Standard Time</timezone>
<work-phone/>
</initiator>
<organizational-unit>
<name>&lt;s ID=&quot;847&quot;/></name>
<organizational-unit-id>1</organizational-unit-id>
</organizational-unit>
<requisition-entry>
<closed-date/>
<data-values>
<data-value multi-valued="false">
<name>amqpDict1.field1</name>
<value>a</value>
</data-value>
<data-value multi-valued="false">
<name>amqpDict1.field2</name>
<value>b</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field3</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgAC/EsDiHp1ERsOpdKSKdrtgMjFWcVoO96aCFSkCgwa2tkBo
vKoOHzaillNiJ47+aY8zCWKwd17tCjmMLEkeQlTvrDvIHR/DT1iWSmN08DI9+Ns7hY3A/g6ijUoM
gcuMczH+5F/pGtgupLZF/L7FwOwu4VcKVWM/2N5tuXGz+1aHTRAAAAByQXr/yD/75Ysy57LnQLxc
```

```
</value>
</data-value>
<data-value is-secure="true" multi-valued="false">
<name>amqpDict1.field4</name>
<value>ut3u4RC699wz7Jd2O+4cix23m9XXgABReynDp5AdBgRi5ivhS05Unv98BgWgxc5YNcZrCihhhH/1
rzZqhjiIjAoRelIjADDb3IQP72armXsLRTvh0/fusd4jLdVIm4q1s+GaSTt6F3oqQeZ4RLhVUopo
p2zNAxmjaGj629C8gWREes3Z8EjDvXg5K1YVi90fXJV1jDoADhAAAABP1hct5TNV2d8R1cN/qDtK
</value>
</data-value>
</data-values>
<due-date>2014-04-25 20:00:00</due-date>
<item-number>1</item-number>
<price-per-unit>0.0</price-per-unit>
<priced>true</priced>
<quantity>1</quantity>
<rejected>false</rejected>
<rejected-date/>
<requisition-entry-id>69</requisition-entry-id>
<revision-number>105</revision-number>
<service>
<estimated-cost>0.0</estimated-cost>
<name>queueService1</name>
<parameters>
<default-duration>0.0</default-duration>
<priority>2</priority>
<start-date/>
<start-mode>0</start-mode>
</parameters>
<pricing-schema>0</pricing-schema>
<quantity>1</quantity>
<service-id>2</service-id>
<version>105</version>
<standard-duration>0.0</standard-duration>
</service>
<start-after/>
<start-date>2014-04-23 14:10:48</start-date>
<start-mode>0</start-mode>
<status>1</status>
</requisition-entry>
<requisition-id>64</requisition-id>
<started-on>2014-04-23 14:10:47</started-on>
<status>1</status>
</requisition>
<context>
<requisitionentryref itemnumber="1"/>
</context>
</task-started>
</message>
```

Sample JSON Outbound Message

```
{
  "message": {
    "taskStarted": {
      "task": {
        "actualDuration": 0.0,
        "calendarEntries": {},
        "checkLists": {},
        "completedDate": "",
        "contextId": 17,
        "contextType": "Requisition Entry",
```

```
"dueDate": "2016-11-29 19:00:00",
"effort": 10.0,
"estimatedDate": "",
"expectedDuration": 10.0,
"flagId": 0,
"isSharable": true,
"isShared": true,
"nextActionId": 2,
"performer": {
  "companyAddress": "",
  "email": "",
  "fax": "",
  "firstName": "Default Service Delivery",
  "homeOu": {
    "name": "<s ID=\"847\"/>",
    "organizationalUnitId": 1
  },
  "homePhone": "",
  "lastName": "Queue",
  "loginName": "",
  "personId": 2,
  "personalAddress": "",
  "supervisorName": "",
  "timezone": "Pacific Standard Time",
  "workPhone": ""
},
"performerRole": {
  "name": ""
},
"performerActualDuration": 0.0,
"priority": 2,
"queue": {
  "companyAddress": "",
  "email": "",
  "fax": "",
  "firstName": "Default Service Delivery",
  "homeOu": {
    "name": "<s ID=\"847\"/>",
    "organizationalUnitId": 1
  },
  "homePhone": "",
  "lastName": "Queue",
  "loginName": "",
  "personId": 2,
  "personalAddress": "",
  "supervisorName": "",
  "timezone": "Pacific Standard Time",
  "workPhone": ""
},
"scheduledStartDate": "2016-11-28 17:00:00",
"startDate": "2016-11-26 12:32:36",
"stateId": 2,
"subject": "amqpTask1",
"supervisor": {
  "companyAddress": "",
  "email": "",
  "fax": "",
  "firstName": "Default Service Delivery",
  "homeOu": {
    "name": "<s ID=\"847\"/>",
    "organizationalUnitId": 1
  },
  "homePhone": "",
  "lastName": "Queue",
```

```
    "loginName": "",
    "personId": 2,
    "personalAddress": "",
    "supervisorName": "",
    "timezone": "Pacific Standard Time",
    "workPhone": ""
  },
  "supervisorRole": {
    "name": ""
  },
  "taskId": 67,
  "waiting": false
},
"requisition": {
  "services": 0,
  "actualCost": 0.0,
  "actualDuration": 0.0,
  "closedOn": "",
  "comments": {},
  "costCenterCode": "",
  "customer": {
    "companyAddress": "",
    "email": "internal@newscale.com",
    "fax": "",
    "firstName": "admin",
    "homeOu": {
      "name": "<s ID=\"847\"/>",
      "organizationalUnitId": 1
    },
    "homePhone": "",
    "lastName": "admin",
    "loginName": "admin",
    "personId": 1,
    "personalAddress": "",
    "supervisorName": "",
    "timezone": "Pacific Standard Time",
    "workPhone": ""
  },
  "dueOn": "2016-11-26 12:32:35",
  "expectedCost": 0.0,
  "expectedDuration": 0.0,
  "external": false,
  "initiator": {
    "companyAddress": "",
    "email": "internal@newscale.com",
    "fax": "",
    "firstName": "admin",
    "homeOu": {
      "name": "<s ID=\"847\"/>",
      "organizationalUnitId": 1
    },
    "homePhone": "",
    "lastName": "admin",
    "loginName": "admin",
    "personId": 1,
    "personalAddress": "",
    "supervisorName": "",
    "timezone": "Pacific Standard Time",
    "workPhone": ""
  },
  "invocations": {},
  "organizationalUnit": {
    "name": "<s ID=\"847\"/>",
    "organizationalUnitId": 1
```

```
},
"requisitionEntry": [
  {
    "closedDate": "",
    "dataValues": {
      "dataValue": [
        {
          "name": "amqpDict1.field1",
          "value": [
            "a"
          ],
          "multiValued": false
        },
        {
          "name": "amqpDict1.field2",
          "value": [
            "b"
          ],
          "multiValued": false
        }
      ]
    },
    "dueDate": "2016-11-29 19:00:00",
    "itemNumber": 1,
    "pricePerUnit": 0.0,
    "priced": true,
    "quantity": 1,
    "rejected": false,
    "rejectedDate": "",
    "rejector": {
      "companyAddress": "",
      "email": "",
      "fax": "",
      "firstName": "",
      "homeOu": {},
      "homePhone": "",
      "lastName": "",
      "loginName": "",
      "personId": 0,
      "personalAddress": "",
      "supervisorName": "",
      "timezone": "",
      "workPhone": ""
    },
    "requisitionEntryId": 17,
    "revisionNumber": 3,
    "service": {
      "estimatedCost": 0.0,
      "form": {
        "fields": {}
      },
      "name": "amqpService1",
      "parameters": {
        "defaultDuration": 0.0,
        "priority": 2,
        "startDate": "",
        "startMode": 0
      },
      "pricingSchema": 0,
      "quantity": 1,
      "serviceId": 142,
      "version": 3,
      "standardDuration": 0.0
    },
```

```
            "startAfter": "",
            "startDate": "2016-11-26 12:32:36",
            "startMode": "0",
            "status": 1,
            "bundle": false
          }
        ],
        "requisitionId": 17,
        "startedOn": "2016-11-26 12:32:35",
        "status": 1
      },
      "context": {
        "requisitionentryref": {
          "itemnumber": 1
        }
      },
      "taskType": "task"
    },
    "channelId": "42346EF5-D2A7-EC03-E050-11AC02003EE1"
  }
}
```

CHAPTER 5

Designing Integrations with Service Link Standard Adapters

Overview

Service Link is the Service Catalog module that provides integration with external systems. It supplies a framework for configuring interfaces that allow delivery tasks, authorizations or reviews defined within a Service Catalog workflow to be performed by other systems and a user interface for monitoring the operation of these interfaces.

The most common scenario for the use of Service Link is where data associated with a delivery plan task needs to be passed outside of Service Catalog in order to ensure that the service is delivered satisfactorily. For example, Service Link might invoke Cisco Process Orchestrator to fulfill a service request, a message might be passed to a hardware vendor for a procurement action or to an inventory or asset management system for a data record update. The external application may then send one or more messages back to Service Catalog. Each message, in turn, could update Service Catalog with the current status of the task within the external system, eventually indicating that the task has been completed and that the Service Catalog workflow (delivery plan) can continue with subsequent tasks.

Service Link provides a number of built-in adapters to facilitate communication with external applications using different transport mechanisms including the interchange of files; database updates; web communication via http post requests or web services; and queue-based messaging. In addition to these default adapters, developers may use the Service Link Adapter Development Kit (ADK) to develop and deploy custom adapters.

Prerequisites to Develop Service Link Integrations

Developing Service Link integrations requires a range of technical skills. These include:

- Understanding of service design, including how to configure dictionary usage in Active Form Components (AFCs) and how to design tasks in a delivery plan.
- Thorough knowledge of the target third-party system, including the servers hosting the application.
- For all adapters, a basic understanding of XML tag structure since Service Link operates by sending XML messages between Service Catalog and the external system.
- For all adapters, an intermediate grasp of configuring XML Stylesheet Language (XSL) transformations, to supplement the XML transformations which can be applied by use of the Service Link wizards.
- If database adapters are to be used, SQL knowledge is also needed.

- If an http/web services adapter is to be used to pass messages between Service Catalog and a web service, knowledge of web services components like SOAP, WSDL, RESTful API, and web service security is helpful.

Service Link Design Components

Service Catalog offers two approaches to designing integrations.

- The Integration wizard, available in Service Designer, provides a wizard-driven approach for creating web services integration.
- The Service Link module provides capabilities for creating and maintaining all integrations, regardless of the messaging protocols used to communicate with the external system.

Once an integration has been added, it may be viewed and maintained through the advanced configuration capabilities available through Service Link. Advanced users may create even web services integrations using this functionality, bypassing the wizards if desired.

Administrators also use Service Link to administer and troubleshoot integrations in a production environment.

An integration consists of the following components:

- Adapters

  An adapter is a logical representation of a transport component by which Service Catalog sends XML documents or other messages to third-party systems. Prepackaged adapters support different message transport protocols; including file, http/web service, JMS, IBM MQ, and database.

  Adapters are composed of two components:

  - An inbound adapter

    Inbound adapters manage messages coming from an external system. The external system message may be altered into a “standard” nsXML (formerly known as newScale XML) format through the use of transformations so that the data can be interpreted by Service Catalog.

    There are two types of inbound adapters: pollers and listeners. A poller is a thread that periodically wakes up and looks for incoming messages, while a listener waits and is awakened by an incoming external message. An example of a poller is the inbound file adapter, which needs to periodically check for messages. An example of a listener adapter is the Web Services Listener Adapter which waits until an HTTP response is received.

  - An outbound adapter

    Outbound adapters manage the XML messages coming out of Service Catalog and send them to the configured external system. A “standard” nsXML outbound message comes to Service Link which may then alter the message through the use of transformations, so that it meets the expected format of messages directed to the external system. The outbound adapters then apply the correct protocol and logic to send the messages to the external system.

- Agents

  An agent is a logical representation of a transport mechanism by which Service Catalog communicates to/from a third-party system. Agents may be used by service designers to direct tasks to their proper third-party destination. In addition to tasks, authorizations and reviews can be externalized by specifying an agent to direct this action to an external system.

  An agent is composed of an inbound and outbound adapter, optional message transformation (XSLT) components, optional parameters, and other settings to address error conditions.

- Transformations

  XML stylesheet (XSL) transformations transform outgoing messages into a format understood by a third-party system, and transform incoming messages into a format understood by Service Catalog.

  An agent which includes an outbound adapter automatically s an nsXML message, containing information relevant to the current requisition and task. A transformation associated with the agent may then transform that message into an external message, which are delivered to the external system via the outbound adapter configured for the agent. Similarly, an inbound agent receives an external message via the associated adapter. A transformation must then transform the message into an incoming message type that is recognized and processed by the Business Engine, the Workflow Manager for Service Catalog.

- Dictionaries and Active Form Components

  A dictionary is the service design component that holds fields of data required to fulfill a specific service request. Agent parameters mapped to dictionary fields (or other data available in the service request) provide a standard outbound message format easily understood by external systems. Agent parameters in an inbound message received from an external system instruct Service Link to update the value of the dictionary fields mapped to those parameters. The changed form data is immediately available in the service form. The active form component in which the dictionary is included must, in turn, be included in the service that implements the Service Link integration.

  You can optionally bypass the validation check to prevent the removal of active form component, which contains 1 or more Service Item-based dictionaries by setting the *serviceform.simtask.validation.skip* property in the newscale.properties file to true.

The Integration Wizard automatically s an agent and transformation, as well as an integration dictionary and active form component to complete the agent configuration. Once these components have been created, they are maintained through Service Link and Service Designer.

Service Link Interaction with Business Engine and nsXML

The key to understanding Service Link is to understand its interaction with the Business Engine. The Business Engine is the component that is responsible for all workflow. Workflow actions include:

- Starting tasks in the correct sequence in a delivery plan.
- Marking a task as complete when all requirements for completion have been met.
- Sending emails as configured when the triggering event occurs.

In a task plan that doesn't use Service Link (that is, where all tasks are internal to Service Catalog), the operation of the Business Engine is largely invisible. The use of the Business Engine becomes apparent in Service Link, because Service Link must handle or generate messages that the Business Engine understands in order for the status of external tasks to be changed.

When the Business Engine starts an external task (that is, a task which is to be handled by Service Link), it generates an outbound nsXML message. The Service Link agent that is handling the outbound task is then responsible for transforming that nsXML message into a format that can be understood by target system and delivering that message to the target system via the outbound transport mechanism (adapter) specified in the agent.

Similarly, if Service Link is configured to receive an inbound message from an external system, it must transform that message into an inbound nsXML message that can be understood by the Business Engine. Inbound messages are available to update the service form data for the current request; to complete the current task; or to add user comments to the current request.

Valid nsXML messages are discussed in more detail in the following section.

Designing Service Link Integrations

Cisco Prime Service Catalog offers two approaches for designing, developing and deploying Service Link integrations with third-party systems:

- Integration Wizard: If the integration is via a web service, you can use the Integration Wizard via Service Designer to all Service Catalog integration components. The Web Service Definition Language file (wsdl) must be available to use this approach.
- Service Link configuration: If the integration uses any other transport mechanism or you wish to review or modify components originally created via the Integration Wizard, use the screens provided by Service Link and Service Designer to configure the integration components.

Service Link and Service Designer configuration uses the following methodology to design, develop and deploy integrations with third-party systems:

- Design the communication protocol to be used with the third-party target system. This includes inbound and outbound adapter selection, message format and content.
- If necessary and deploy a custom Service Link adapter.
- Use Service Designer to design the service that implements the Service Link integration. The design components typically include one or more dictionaries, which contain data to be passed to the external system via agent parameters, as well as the active form components that include those dictionaries and configure the display properties for the fields in those dictionaries.
- Configure agents, selecting appropriate outbound and inbound adapters, defining the properties of each, together with parameters passed in either direction. Service Link includes wizards and drop-down lists to partially automate the definition of agent parameters.
- Add transformations, if needed, for Service Catalog to understand messages from third-party systems, and for third-party systems to understand messages from Service Catalog.
- Use Service Designer to associate the agent with a task in the service's delivery plan. If applicable, ensure that agent parameters are properly mapped to dictionary fields used in that service.
- Test the configuration by requesting the service containing your task and then monitoring messages and external tasks via the corresponding Service Link pages.

This process is discussed in more detail in the following sections.

The Integration Wizard is described in the Using the Integration Wizard in Service Designer.

Accessing Service Link

To access Service Link, choose **Service Link** from the Module Menu. The Service Link home page appears.

The Service Link home page contains the elements shown in Service Link Home Page Elements below.

Table 5-1 Service Link Home Page Elements

Element	Description
Menu Bar	Located at the top of the page, contains the tabs used to administer a Service Link environment and to develop and maintain Service Link integrations.
Common Tasks	Located on the left side of the page, contains quick links to view a complete list of failed messages and to an agent.

Table 5-1 Service Link Home Page Elements

Service Link Status	Located on the left side of the page, shows the current status of Service Link.
Messages (Last 30 days)	Located in the right pane of the page, this graph summarizes message volume for the most recent 30-day period.
Recent Failed Messages	Located on the bottom right of the page, this grid lists the most recent Service Link messages that were not delivered successfully. Hyperlinks are provided to the message, requisition, and agent details.

Designing the Communication Protocol

In addition to the configuration and testing work that are executed in Service Link, equivalent work must be undertaken by the technical resources responsible for the third-party system. A well considered design is essential if the interface is to operate robustly.

The interfacing capabilities of the system to be communicated to will normally dictate the basic design of the integration, that is, which adapters will be used.

File and database adapters are the simplest to configure. If JMS, MQ or http/web service adapters are deployed, some expertise from network management teams may be involved to ensure that connectivity issues do not prevent the data from moving from one system to the other. Data security concerns are likely to be a factor if the target system exists outside your company's network—for example, to use a SOAP message sent via http or https to communicate with an outside vendor.

Normally the data required for the outbound Service Link communication would be assessed first. Consideration needs to be given to what fields are readily available (via the nsXML outbound message) and what additional data needs to be provided (via form data and agent parameters). While outbound communications only occur once per task (when the task starts), multiple separate inbound communications can be supported. On receiving its instruction to perform work, a third-party system may issue a single (inbound) communication on completion. Alternatively, multiple updates could be sent before the work is complete. Examples include where a reference ID may be communicated, textual status updates have to be sent back and finally the completion confirmation is communicated.Managing Service Link Adapters

Managing Integrations

This section describes about managing integrations.

Managing Adapters

Service Link adapters are preconfigured for use. Additional adapters may be developed using the Adapter Development Kit. You may review the available adapters using the Adapters page of the Manage Integrations tab in Service Link.

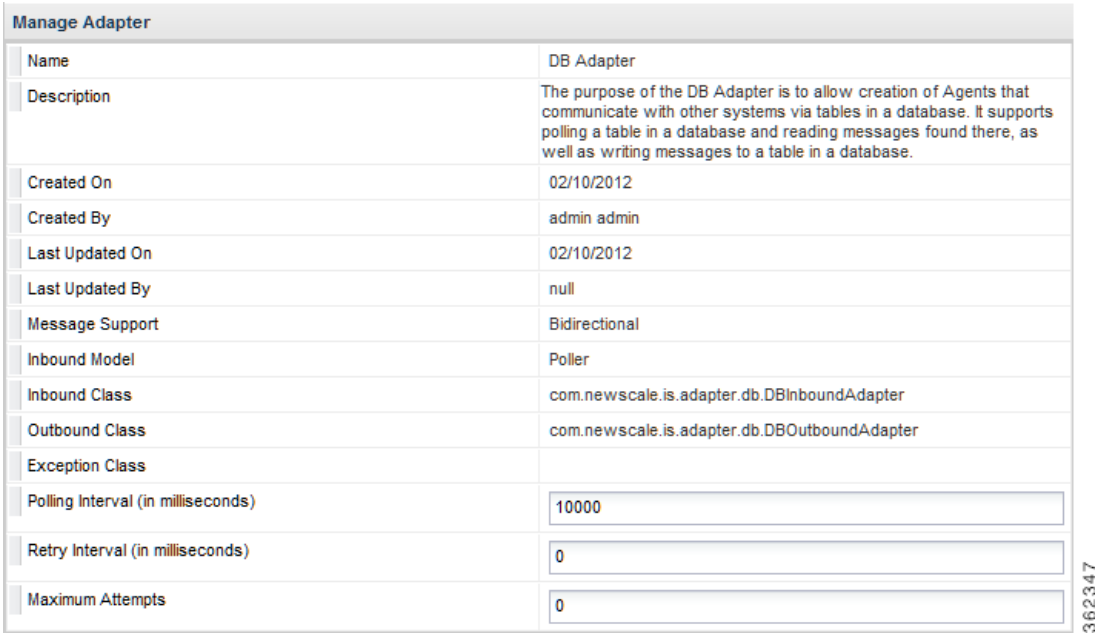
Step 1 From the Service Link home page, click **Manage Integrations**. Then click the **Adapters** sub tab.

The Adapters page appears.

Step 2 In the Name column, click the desired adapter.

The Manage Adapter page appears for the chosen adapter. The details for the Database Adapter are shown below.

Figure 5-1 Manage Adapter Page



Most of these general properties should typically not be changed by Service Link developers. The "Polling Interval", "Retry interval" and "Maximum Attempts" may need to be changed as per requirement. Any changes are inherited by all agents that use this adapter type.

Additional outbound and inbound properties are specified when the adapter is used in an agent. These properties are described in the Managing Service Link Adapters.

Managing Agents

The Agent Wizard walks you through configuring an agent. The wizard consists of eight pages; some pages may be skipped, depending on options chosen on the previous pages.

The pages of the Agent Wizard are summarized in Agent Wizard Pages Table below.

Table 5-2 Agent Wizard Pages Table

Page	Description/Usage
General Information	The name, action, and description for the agent, as well as other general information regarding its configuration and behavior
End Points	Adapters and transformations used by the agent
Outbound Properties	Detailed configuration options for the outbound adapter
Inbound Properties	Detailed configuration options for the inbound adapter
Outbound Request Parameters	Outbound parameters for all adapter types except the VMware adapter

Table 5-2 Agent Wizard Pages Table

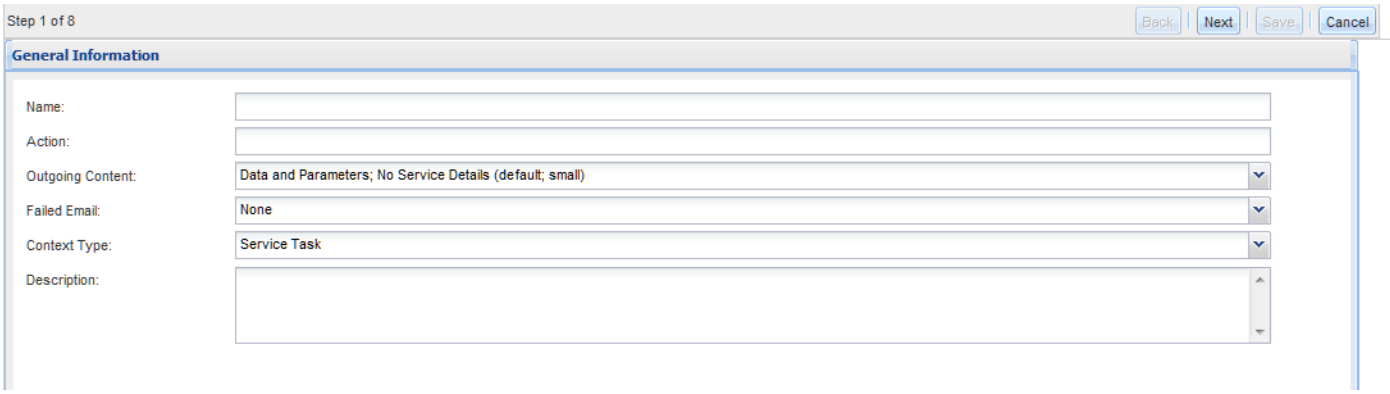
Outbound Response Parameters	Parameters that are received in a synchronous response to an outbound message send to an http/ws adapter
Inbound Parameters	Parameters received as part of the inbound message
Summary	Summary page displaying all information previously entered

To manage an agent:

Step 1 From the Common Tasks area of the Service Link home page, click **Agent**, or choose **Manage Integrations > Agents > Agent**.

The General Information page of the Agent wizard appears. This is the first of eight pages that comprise the wizard. Some pages might not be relevant for a particular agent configuration, and can be skipped.

Figure 5-2 General Information Page



Provide values for the fields described in Creating Agents – General Attributes Table below, then click **Next**.

Table 5-3 Creating Agents – General Attributes Table

Setting	Description
Name	A name for the agent. The name must be unique.
Action	A description of the action performed by the agent, for example, “Service Catalog To Remedy- Ticket”. Although uniqueness is not enforced, actions should be unique, since they are presented in a pick-list when you are asked to assign the agent to a task.
Outgoing Content	An option for specifying which nodes of the outbound nsXML message should be generated for this agent. More details are given below.
Failed Email	An email template that is sent if the outbound message cannot be delivered to its destination. Email templates must be defined via the Notifications option of the Administration module.
Context Type	The type of agent. Service Link agents are “Service Tasks”, to allow the agent to be used as an external task in a delivery plan. “Service Item” agents are used to import the definition or data or service items as configured via Service Item Manager.
Description	A detailed description of the agent. A full description here can assist with supporting the integration.

Failed Email

The Failed Email notification can be generated in case an outbound message cannot be delivered. In addition to the standard sets of namespaces available for all task-related Emails, Service Link Failed Emails can include details about the message being generated at the time of the failure. Including these namespaces in the Email template may help in diagnosing the problem. Details on available namespaces are given in the *Cisco Prime Service Catalog Designer Guide*.

Failed email is not applicable to inbound messages; a notification is not generated when Service Link fails to process an inbound message.

Outbound Message Content

The outbound adapter generates an nsXML message that is stored in the Service Catalog database. This message is then subject to a transformation, and the resultant external message is delivered via the specified adapter to the desired destination.

The format of the message is documented in Designing Integration with Adapter Development Kit and by the corresponding XML schema available on the application server at ISEE.war/WEB-INF/classes/nsxml.xsd. The complete message includes all information about the service request. Such messages can get quite large (easily exceeding 500 K, depending on the number of dictionaries and fields used in the service) and consequently consume large amounts of storage within the database, as well as consuming significant amounts of CPU to produce.

To reduce resource consumption, Service Catalog offers the following options.

- The Administration Setting to “Compress messages” can compress Service Link messages stored in the database. This greatly reduces storage requirements, but potentially complicates troubleshooting, since messages are no longer human readable by a DBA or support personnel.

- Service Link Message Purge scripts are available. These scripts can purge messages for completed tasks from the transactional database. This reduces storage requirements for the messages in the database. Details on using the Message Purge scripts are given in the Service Link Troubleshooting and Administration.
- The outgoing content can be configured to include only selected nodes of the nsXML message by manipulating the "Outgoing message content" property. This reduces memory and CPU requirements for processing outbound messages.

The default message content is "Data and parameters; no Service Details (small)," which generates a nsXML message that does not include content nodes describing the service requested. Agent parameters and transformation must be designed with the outgoing content type in mind, to ensure that all required content is included in the nsXML message. Specifically, if eliminating the dictionary data from the outbound message, agent parameters must be mapped to appropriate form fields (or constants). In cases where a service has many form fields that are not needed for an external task, the XML size reduction and associated CPU utilization reduction are substantial.

Outgoing content options are summarized in Outgoing Message Content – Options Table below.

Table 5-4 Outgoing Message Content – Options Table

Option	Description
All Message Details (large)	The complete Service Link message is generated.
Data, Form and Parameters (medium-large)	Information about the service and its tasks is omitted. The message is restricted to data (field values on the service form), form (complete metadata about the dictionaries and fields on the service form) and parameters (values supplied to agent parameters).
Data and Parameters (medium)	The message is restricted to information about the requisition, all data values entered on the service form and parameter values.
Data and Parameters; No Service Details (small)	The message is restricted to information about the requisition, all data values entered on the service form and parameter values (the default). The "small" option must be specified for the VMware adapter.
Only Parameters (minimal)	The message is restricted to information about the requisition and the agent parameters.

Adapter Selection

An agent may be configured to manage both outbound and inbound communications; just outbound communications; or just inbound data. It is possible to use different adapter types for each direction, for example, a database adapter could be used to write data outbound but that system would then respond by writing files into a directory that would be read by an inbound file adapter.

Once an adapter type is chosen, subsequent pages of the wizard are adjusted to display properties relevant to the adapter type and usage (outbound or inbound). Property values must be supplied as part of the agent definition.

The End Points page of the Agent wizard (page 2 of 8) allows you to designate the adapters to be used in the agent as well as any transformations to be applied to the outgoing or incoming message. Transformations must have previously been defined using the Transformations subtab of the Manage Integrations option.

Figure 5-3 End Points page of the Agent wizard

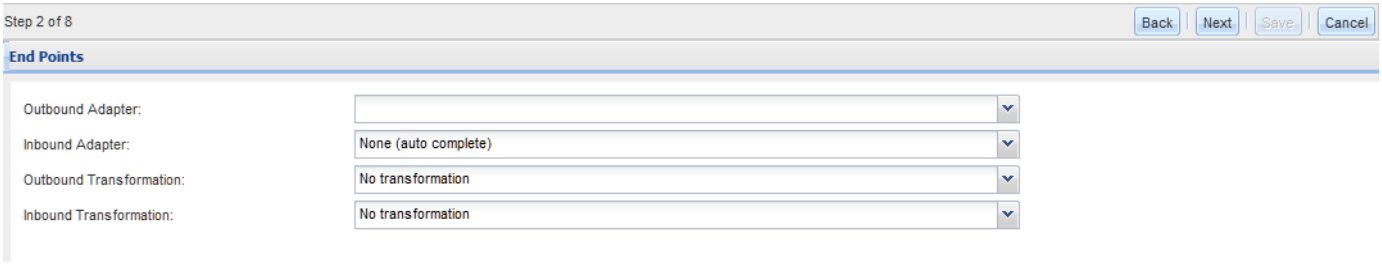


Table 5-5 Creating Agents – End Point Properties Table

Setting	Description
Outbound Adapter	The default or custom adapter used for sending a message from Service Catalog to the external system.
Outbound Transformation	The transformation (or none) to apply to the nsXML message produced by the outbound adapter, in order to generate an external message which can be understood by the external system
Inbound Adapter	The adapter to be used for receiving messages from an external system, or "Auto Complete" if no inbound message is expected.
Inbound Transformation	A transformation (or none) to apply to the incoming message in order to produce a nsXML message understood by the Business Engine; applicable to Service Tasks only.

Properties applicable to each type of adapter are described later in this chapter. In fact, the next two pages of the wizard, dedicated to configuring the outbound and inbound adapters, will vary, depending on which adapter has been chosen. However, it is worth discussing two special cases: Dummy adapters and the auto complete option.

Dummy adapters can be configured within an agent as either the outbound or inbound adapter. If a dummy adapter is chosen, it means that the agent is only operating in a single direction. For example, an agent configured with a dummy inbound adapter means that the agent is only responsible for outbound communications. In turn, there could be a separate (inbound only) agent configured that would be dedicated to updating and closing tasks.

The auto complete option is available only as the choice for the inbound adapter part of the agent. Its effect is similar to choosing "Dummy adapter", that is, the agent will only be managing outbound communications. The key difference is that after the outbound communication associated with the task has been sent, the task will automatically be completed and the rest of the delivery plan will continue to be executed.

Using Agent Parameters

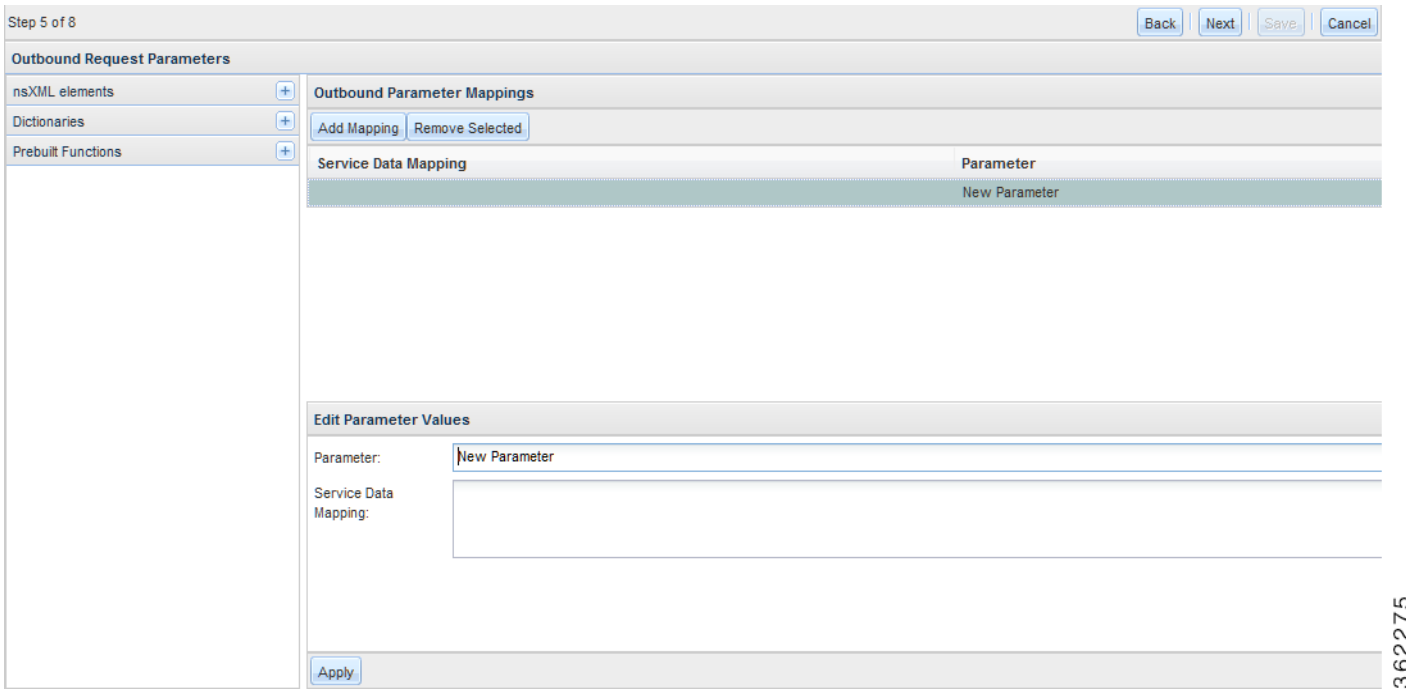
Agent parameters may be used in conjunction with both outbound and inbound adapters.

Parameter mappings specified as part of the agent definition provide default value to be used. These mappings can be overridden on a service-by-service basis by editing the task definition for the service in Service Designer.

Outbound Request Parameters

Agent parameters used in an outbound message provide a way to supplement the content nodes in the standard nsXML outbound message with additional data and to organize content nodes in an easy-to-address format. The parameters are easily accessible via the XSL transformation, to allow their inclusion in the external message.

Figure 5-4 Outbound Request Parameters



A parameter mapping is assigned by typing the source elements in the Service Data Mapping area, or by building an expression by using the elements available in the drop-down lists to the left of the Parameter Mappings pane. A mapping may consist of a combination of:

- A constant value
- A dictionary field on a service form
- A nsXML element
- A prebuilt function applied to one of the above elements

Mapping an agent parameter has the following advantages:

- A transformation that extracts that parameter value does not need to refer to the name of the dictionary field in which the value was supplied, but may refer simply to the agent parameter by name. This encourages agent reuse across different services and dictionaries.
- A smaller outbound message content type may be used, provided all other content required in the message is supported, since parameters are included in all content types.
- nsXML elements and XPATH operations that would not be accessible without using a transformation can be included in the external message.

To an outbound parameter:

Step 1 On the Outbound Request Parameters page, click **Add Mapping**.

The Edit Parameter Values dialog box appears at the bottom of the page.

Step 2 Enter a name for the parameter.

Parameter names can include spaces, but should not include special characters (such as “>” or “&”), which have significance in XML messages.

Step 3 Specify the value/mapping for the parameter, using the guidelines given below.

Constant Values

Sometimes a constant value that is not dependent on the requisition or service details must be passed to the external system. For example, if the system needs a name for the source of the external system, “Service Catalog” can simply be typed as the Service Data Mapping (without the quotation marks).

Mapping to Outbound nsXML

Selected elements of the standard nsXML outbound message are available to be mapped to agent parameters. These are summarized in the table below.

Table 5-6 Outbound nsXML Mappings

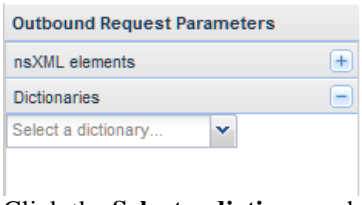
nsXML Element	Contents/Description
Customer	The login name of the customer for the requisition
Initiator	The login name of the requestor (initiator) of the requisition
requisition-entry-id	Number that uniquely identifies a service request within a requisition; typically used to differentiate multiple services within a shopping cart
expected-cost	Transactional price for the service
expected-duration	Service standard duration
requisition-id	Number that uniquely identifies the requisition; referenced in My Services and Service Manager
channel-id	Globally unique identifier (GUID) that identifies an external task

Mapping Agent Parameter to Dictionary Field

To map an agent parameter to a dictionary field:

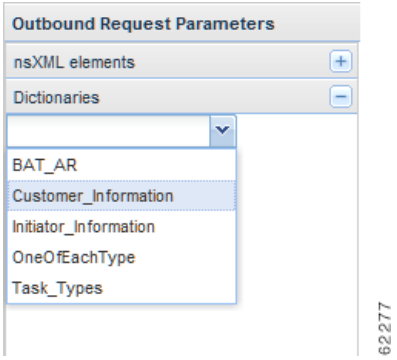
Step 1 Expand the Dictionaries node so that the “Select a dictionary” option appears.

Figure 5-5 Mapping Agent Parameter to Dictionary Field



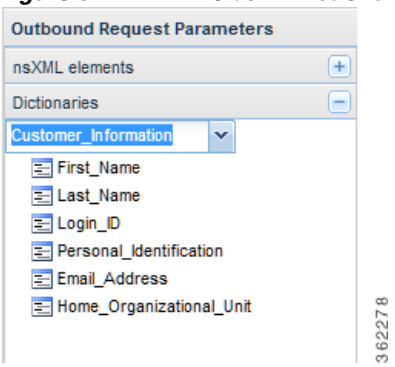
Step 2 Click the **Select a dictionary** drop-down menu to display a list of all Service Catalog dictionaries.

Figure 5-6 Select a dictionary drop-down



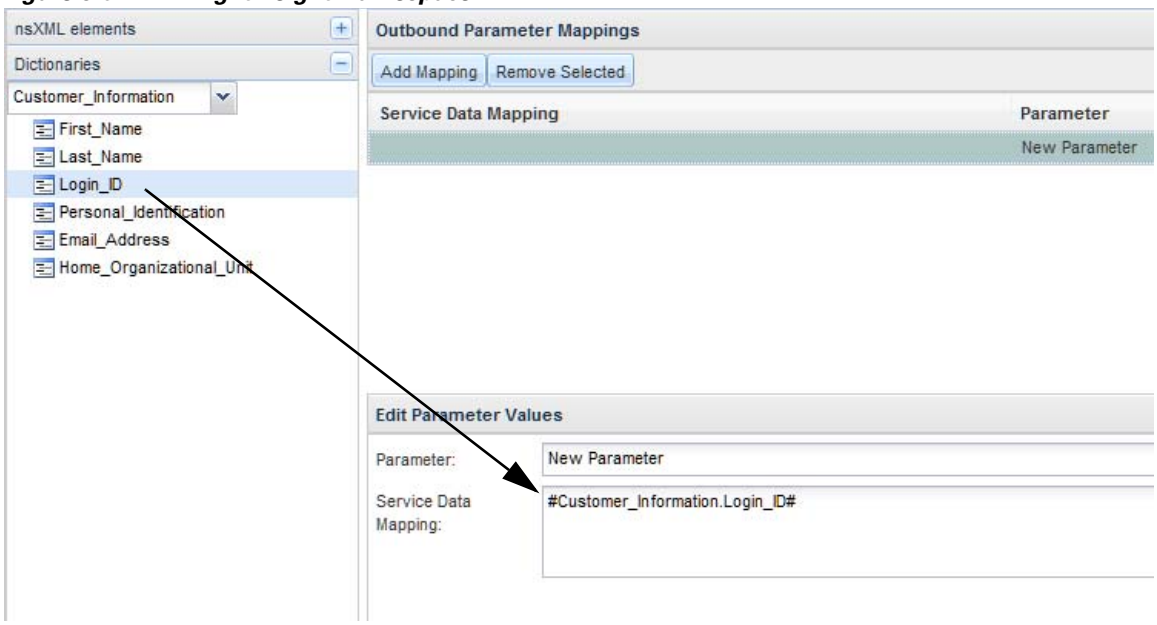
Step 3 Choose the dictionary containing the field to be mapped to the agent parameter. A list of all fields in the dictionary appears.

Figure 5-7 Fields in Dictionary



Step 4 Click the field to be mapped to the agent parameter and drag it to the Service Data Mapping text area. When the drag icon changes to a green check mark, release the mouse. A lightweight namespace for the selected field appears.

Figure 5-8 Lightweight Namespace



Since the agent is defined independent of a service, with the exception of grid dictionaries, any dictionary field can be chosen. It is the responsibility of the service designer to ensure that the referenced dictionary is, in fact, included in the service in which this agent is used.

For any integration that passes the contents of one or more grid dictionaries, you will need a transformation to handle the rows in each grid dictionary, which are stored as multiple dictionary instances. They follow the naming convention of "DictionaryName-n", where n is the grid row number, in the <data-values> section of the outbound nsXML. For example, if a grid dictionary named VMOperation has two rows of data in the service request, the values are represented as below:

```
<data-values>
   <data-value multi-valued="true">
      <name>VMOperation-1.Name</name>
      <value>vmgw01</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.GuestOS_Name</name>
      <value>winNetStandardGuest</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.CPUCount</name>
      <value>1</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-1.Memory</name>
      <value>2048 MB</value>
   </data-value>
   <data-value multi-valued="true">
      <name>VMOperation-2.Name</name>
      <value>vmgw01</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-2.GuestOS_Name</name>
      <value>winNetStandardGuest</value>
   </data-value>
   <data-value multi-valued="false">
```

```
      <name>VMOperation-2.CPUCount</name>
      <value>1</value>
   </data-value>
   <data-value multi-valued="false">
      <name>VMOperation-2.Memory</name>
      <value>2048 MB</value>
   </data-value>
</data-values>
```

There is no support for inbound agent parameter mapping and update to grid dictionary fields.

Applying a Prebuilt Function

Prebuilt functions can be applied to the mapped elements, so that the parameter value fits the semantics or formatting requirements expected in the target system. For example, a field may be shortened, by applying a substring function, if the data definition for the field in the target system accommodates fewer characters than are maintained in Service Catalog. Prebuilt functions are summarized below and explained in more detail in the Prebuilt Functions.

Table 5-7 Prebuilt Functions

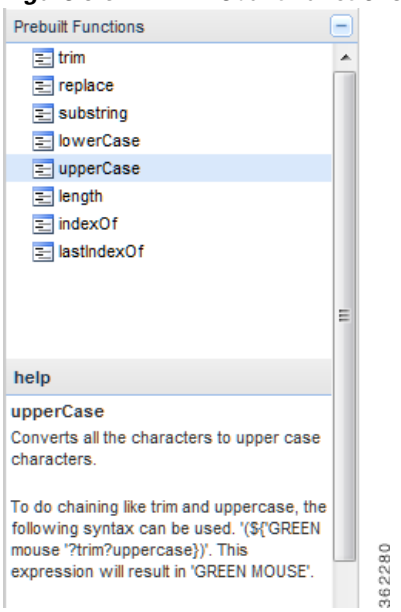
Function	Usage/Description
trim	Trims leading or trailing spaces from the value; especially useful for form data and for incoming messages from the database adapter, which encloses values in CDATA tags.
replace	Replaces all occurrences of one character or pattern with another.
substring	Selects a portion of the string, specified by a starting point and optional length.
lowerCase	Converts the value to all lower case.
upperCase	Converts the value to all capital letters.
length	Returns the number of characters in the string.
indexOf	Returns the index within this string of the first occurrence of the specified substring.
lastIndexOf	Returns the index within this string of the last occurrence of the specified substring.

To apply a prebuilt function to an agent parameter:

Step 1 Expand the Prebuilt Functions node so that the function names appear.

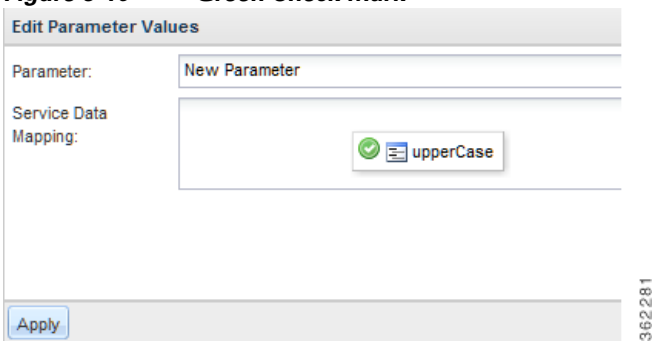
Step 2 Highlight the function you want to use—notice that Help appears, explaining function usage, at the bottom of the pane.

Figure 5-9 Prebuilt Functions



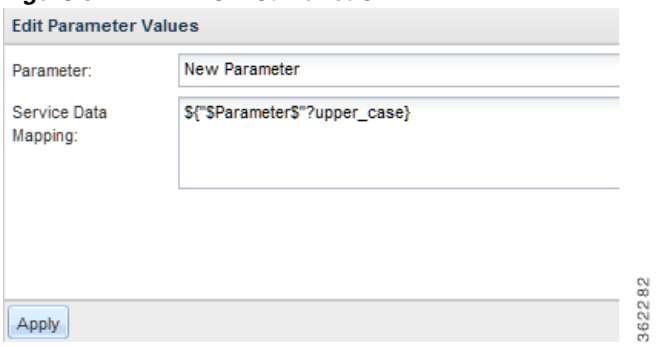
Step 3 Drag the function into the Service Data Mapping box for the parameter. When the drag icon changes to a green check mark, release the mouse.

Figure 5-10 Green Check Mark



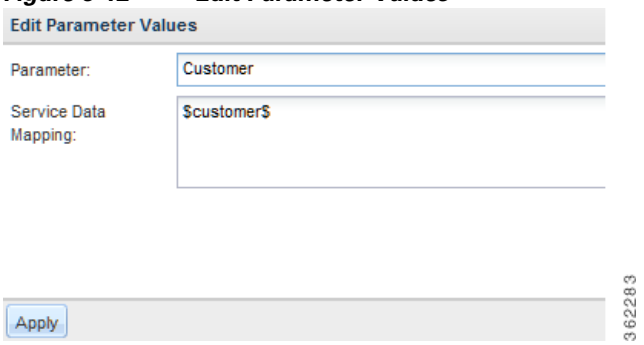
Step 4 The function is defined, with $Parameter$ as a placeholder for the actual value.

Figure 5-11 Defined Function



Step 5 Replace the placeholder with the dictionary field or element of the nsXML message to be used. You need to drag the field or nsXML element to the Service Data Mapping text box, then manually edit the parameter definition.

Figure 5-12 Edit Parameter Values



Adding Agent Parameters to Outbound nsXML

Agent parameters are added to the end of the outbound nsXML message. For example, agent parameters shown below generate the nsXML snippet immediately following.

Figure 5-13 sXML snippet

Service Data Mapping	Parameter
$initiator$	Initiator
$expectedCost$	Transactional Price
$customer$	Customer

362284

```
<agent-parameter multi-valued="false">
    <name>Initiator</name><value>ltierstein</value></agent-parameter>
<agent-parameter multi-valued="false">
    <name>Transactional Price</name><value>0.0</value></agent-parameter>
<agent-parameter multi-valued="false">
    <name>Customer</name><value>Customer</value></agent-parameter>
```

Outbound Response Parameters

Outbound response parameters may be used in conjunction with an outbound http/web service adapter. If the adapter's Process Response setting is true, the response to the original request is processed. That response may include a “Send Parameters” message type. Parameters are defined as for Inbound Agent Parameters.

Inbound Agent Parameters

When used in conjunction with an inbound “Send Parameters” message type, agent parameters allow the external task to update the dictionary field to which the parameter is mapped.

Table 5-8 Inbound Agent Parameter Settings

Setting	Description
Parameter	The parameter name.
Dictionary Field	Select a dictionary field from the drop-down list that displays a list of all dictionary fields. The field name is in the format: DictionaryName.FieldName without enclosing hash marks (#).
Mapping	A prebuilt function applied to derive the value that should be used to update the specified dictionary field.
Mandatory	Check **Mandatory** if the field must be present in the Inbound message. This check box is typically unchecked if a change is made to a service and the parameter is no longer required. Obsolete parameters should not be deleted.

Managing Transformations

To manage a transformation:

Step 1 On the Manage Integrations tab, click the **Transformations** subtab.

Step 2 Click **Transformation**.

The Transformation page appears.

A transformation may be applied to either an outbound and inbound message, by designating the Direction. Two transformations may need to be used for web services outbound messages—one is applied to the outbound request and the second may be applied to a response to that request, if the Process Response setting is turned on.

The Validate button parses the XSLT to ensure that the transformation is well-formed. If it is not (for example, if an XML tag is misspelled or missing), a diagnostic message appears. You need to fix the error before saving the transformation.

However, Service Link does not validate the transformation. For example, no error is detected if a transformation refers to an element that does not exist in the source message; a well-formed XML message would be produced, but it would not be valid for the target system. Therefore, a runtime error would be produced if Service Link produced an external message that was not recognizable by the target application or an inbound message that was not recognizable to the Business Engine.

If you have access to an XML development environment and are familiar with its usage, it may be efficient to use that environment to test the transformation. For an outbound transformation, simply copy the nsXML produced by the agent (before you apply a transformation) into the XML development environment and use this as the source XML. Once you have validated the transformation, copy and paste the XSLT code into a Service Link transformation and associate it with the appropriate agent.

Alternatively, use the "Test" function on this page to quickly preview the output XML from the transformation, especially if you are making a minor change to an existing transformation. Upon clicking the Test button, a popup window is shown where you can paste the nsXML into the source XML panel and exercise the transformation to get the output XML.

The process of manually developing and debugging a transformation is eliminated for outbound web service integrations that are developed using the Integration Wizard. A transformation is automatically created that will transform outbound nsXML into a format compatible with the specified WSDL for the web service. However, if the wsdl or integration requirements change, you will need to follow the steps outlined above to update the transformation.

Step 3 Provide values for the fields described in Transformation Settings Table below.

Table 5-9 Transformation Settings Table

Setting	Description
Name	A name for the transformation. Transformation names should be descriptive of the nature of the interface. They may include the names of the source and target systems, for example “Service Catalog To Remedy”.
Direction	Inbound or Outbound.
Description	A description of the transformation; required.
Format	The supported transformation formats are XST, JOLT, and FTL
Created By	Enter the creator of the transformation.
Created On	Enter when was the transformation created.
Updated By	Enter who updated the transformation.
Last Updated	Enter the date when the transformation was last updated.
Request subtab	The XSLT code which is applied to the nsXML message for outbound adapters and to the external message for inbound adapters.
Response subtab	The XSLT code that is applied to the response received from an http/web service request if the *Process Response* setting for the agent is set to true.

Step 4 Click **Validate** to check that the transformation contains well-formed XSL.

Step 5 Click **Save**.

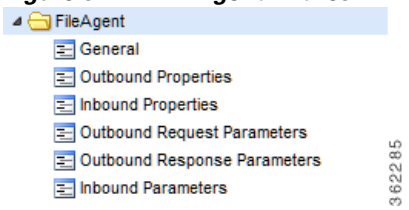
Reviewing Agent Definitions and Property Sheets

You can review or revise any agent definition, whether the agent was created in Service Link or via the Integration Wizard. Once the Agents subtab of the Manage Integrations tab appears, you can either:

- Use the List pane on the left-hand side of the page and click the agent name.
- Scroll through agent information listed on the right-hand side of the page and click the agent name.

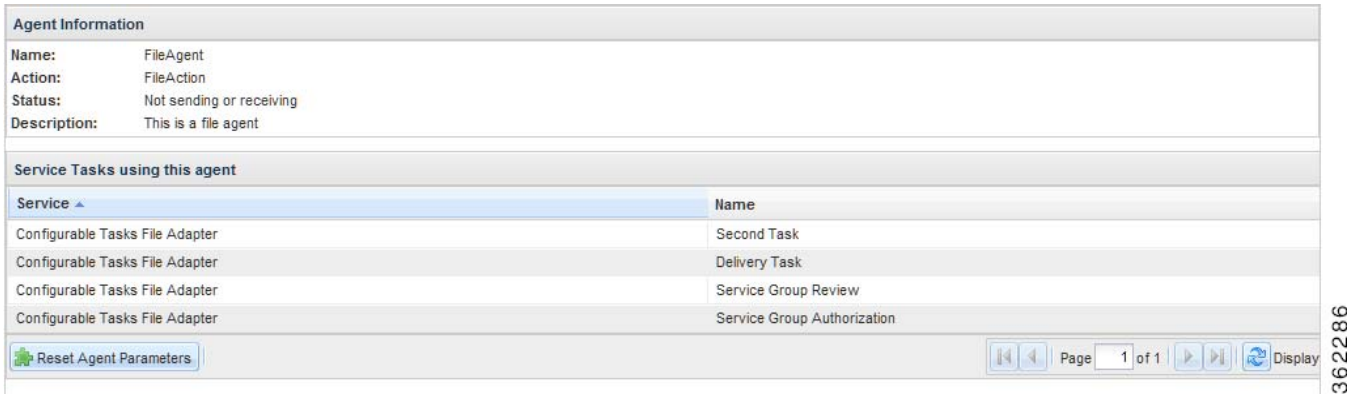
The Agent entry in the list pane is expanded. Click the property sheet for that portion of the agent definition to be edited or reviewed. Once the property sheet appears, enter any changes and click **Save** to save them.

Figure 5-14 Agent Entries



Clicking on the agent name provides an overview of the agent definition:

Figure 5-15 Agent Information



This page is read-only except for the button to Reset Agent Parameters.

Creating and Deploying a Service Link Agent

The procedure below shows the typical sequence of tasks required to deploy a Service Link integration using a file adapter. It can also be used to validate a Service Link installation.

Step 1 From the Common Tasks area of the Service Link home page, select an agent that uses an outbound file adapter by clicking the **Agent** wizard, filling in the location fields and supplying other outbound adapter properties. (Details on these properties are explained in the File Adapter.)

Step 2 Start the agent by navigating to the **Control Agents** tab, locating the agent, choosing it by clicking the mouse anywhere on that line except on the agent name (which is a link to the agent definition) and clicking **Start Selected** at the upper right of the page. Did it start? If not, one of your Service Link configuration settings is wrong or the Integration Server (ISEE) did not start correctly.

Step 3 Verify that the file directories you entered exist on the application server; if not, them. Assure that both Service Catalog and the external application have appropriate access (write or read) to the directories. If these conditions are not met, file transmission will fail at runtime.

Step 4 Go to Service Designer and a service to use this agent.

Step 5 Go to My Services and order the service.

Step 6 If the requisition is created successfully, congratulations! the ISEE outbound queue is working. If you get an “our apologies” page, the JMS queues are not working.

Step 7 Go to the Messages page, accessible from the View Transactions tab. If you see messages from the requisition you just created, congratulations. Your message should have status of “Message sent”.

Step 8 Go to the outbound files directory (for example, C:\cisco\SL\OutboundFiles). If there is an XML file there (verify the date time stamp of the XML file to make sure that it is a new one corresponding to your requisition), your outbound trip for the file agent is completed. Congratulations! The outbound XML file would be a valid nsXML message.

Step 9 For your requisition in the Message Type column, click the **Execute Task** link. The Message Details page appears.

Step 10 Verify that the Requisition ID is correct. Copy the “Channel ID” from the message details screen.

Step 11 an XML file named SampleInbound.xml as follows. Where it says “insert your Channel ID here”, paste the value of the Channel ID that you copied in the last step. (Leave the double-quotes intact).

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="insert your Channel ID here">
    <take-action action="done"/>
</message>
```

For example, after pasting the Channel ID value, the SampleInbound.xml file would look something similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <take-action action="done"/>
</message>
```

Step 12 Put the SampleInbound.xml file in the inbound files directory (for example, C:\cisco\SL\InboundFiles).

Step 13 When the File Agent polls for input, it will automatically pick up the inbound file. (The default setting for the File Adapter's Polling Interval Time is every 10 seconds.) If the SampleInbound.xml file is processed successfully, it will disappear from the directory.

Step 14 Go to the Messages page of the View Transactions tab and look for your requisition. If there is another message for your requisition with the Type=Take Action, and the Status=Inbound Message Completed, then you have achieved a roundtrip.

Step 15 Click the **Requisition ID** link to open the Requisition Status page. Verify that your requisition has the status of “Closed (1 of 1 completed)”.

Configuring a Task to use a Service Link Agent

Once the agent has been defined, it can be used in a service by creating an external task whose workflow invokes the agent. Once the workflow has been configured, you may review or override any agent parameters defined for the included agent.

Creating an External Task

To direct a task to an external (third-party) application using a Service Link agent:

Step 1 Start Service Designer. Select the service that is to include the external task. Click the **Plan** tab.

Step 2 You can use either the Tasks subtab or the Graphical Designer subtab to specify the external task.

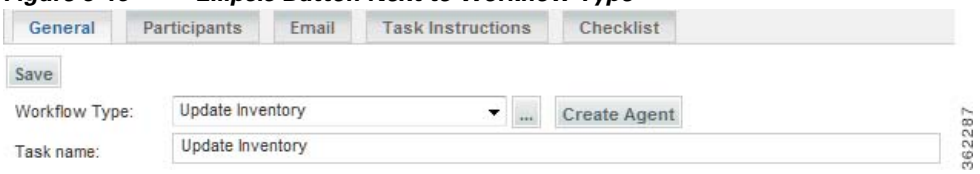
- Using the Tasks subtab for the service, define the task and place it in the correct sequence in the delivery plan.
- Using the Graphical Designer subtab, a task on the diagram, and place it in the correct sequence in the diagram using the Associate tools. Double-click the task to display its property sheet.

Step 3 Using the Workflow Type drop-down list on the General tab, select the desired action from the drop-down list. Note that it is the defined actions in the Service Link module and not the agent names that are listed in this drop-down.

Step 4 Save the workflow/task plan. If you were using the Workflow Designer, return to the Tasks subtab.

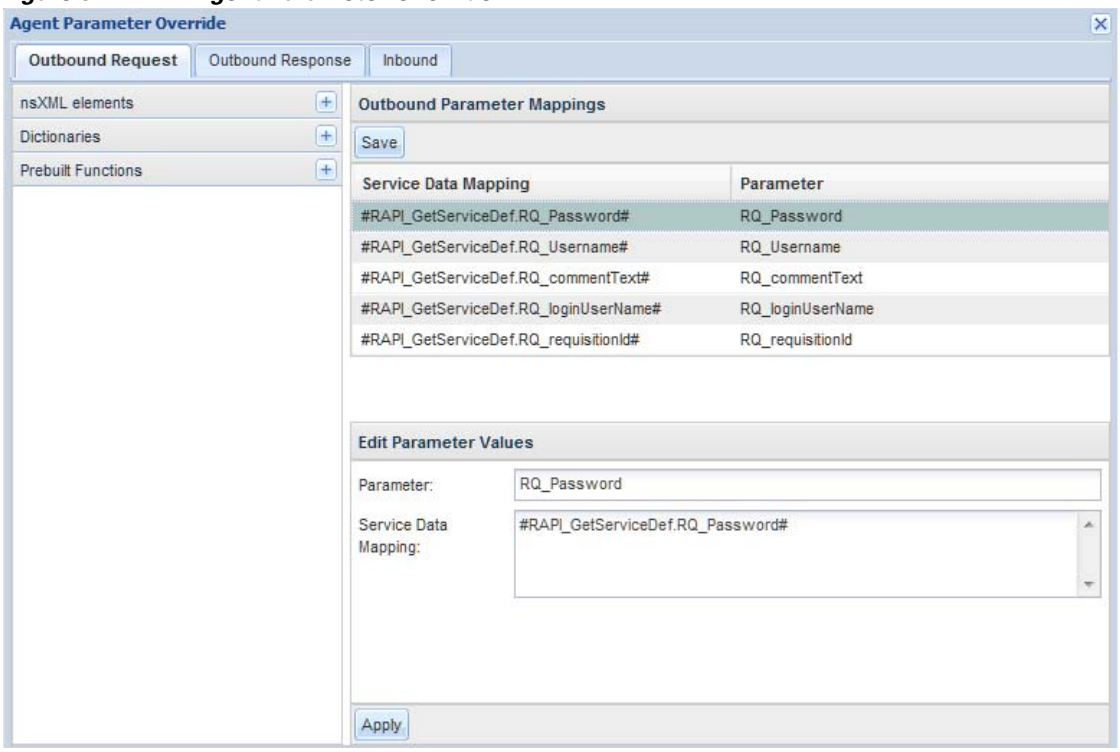
Step 5 Once you have saved the external task, an ellipsis button appears next to the Workflow Type. Clicking the ellipsis allows you to review the parameter mappings currently in effect for the agent (if any) or to change these mappings for this specific service. Click the **ellipsis** button.

Figure 5-16 Ellipsis Button Next to Workflow Type



Step 6 The Agent Parameter Override popup window appears. Review the agent parameters or change the mapping for one or more parameters. Be sure to click **Apply** as you change each parameter and **Save** before closing the window.

Figure 5-17 Agent Parameter Override



Step 7 You may wish to define a performer (person, queue, or functional position) on the Participants tab. The calendar of that performing entity will then be used to calculate the Due Date of the task. If you do not set a value on the Participants tab for external tasks, the calendar of the Default Service Queue is used to calculate the Due Date. In this fashion, due dates are set in the plan and Service Catalog can calculate delivery Operating Level Agreement (OLA) compliance for external tasks and compliance with the Service Level Agreement (SLA) for services containing such tasks.

Synchronizing Agent Mappings and Service Definitions

When you and save a task that uses an agent, the agent parameter mappings specified for the agent are automatically inherited by that individual task. As described above, a service designer may override any of the agent mappings at the task level.

However, if the agent is subsequently modified, to include a different set of agent parameter mappings, such changes are not automatically inherited by tasks that were previously defined to use the agent. Such changes may include:

- Adding an agent parameter
- Deleting an agent parameter
- Changing the mapping of an existing agent parameter

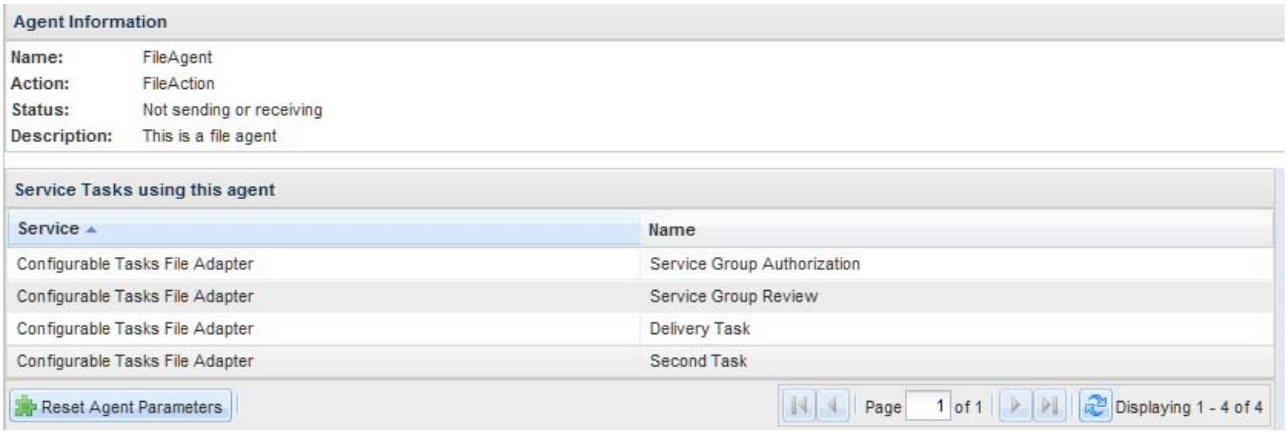
Propagating these changes to services that use the agent can be automated, by following the procedure below:

Step 1 From the Service Link Manage Integrations tab, click the **Agents** subtab.

Step 2 Click the name of the agent whose parameter mappings have been changed.

The Agent Information page appears.

Figure 5-18 Agent Information Page



Step 3 Choose the service or services whose agent parameter mappings need to be resynchronized with the updated agent definition.

Step 4 Click **Reset Selected Tasks**. This button automatically resets all parameter mappings to their agent defaults, so any task-specific mappings would need to be reapplied.

nsXML Messages

The transformation must not only contain well-formed XML, it must produce a well-formed and valid nsXML message. All nsXML messages must conform to the nsXML schema (an XML document that describes the structure of an XML document). The schema is available on the application server at ISEE.war/WEB-INF/classes/xsl/nsxml.xsd.

Outbound nsXML Message

When an external task moves to a status of Ongoing, an outbound nsXML message is generated.

The generated nsXML for each message can be viewed in the Service Link module, by clicking on the nsXML message in the Message Details page. It contains task related data as well as data associated with the parent requisition.

The most important element within the nsXML is the channel-id, an ID that uniquely identifies the external task. This ID is provided to the third-party system and needs to appear in their response if the corresponding data update is to be successfully applied by the business engine.

The channel-id is formatted as a Globally Unique Identifier (GUID). GUIDs are most commonly written in text as a sequence of hexadecimal digits such as:

```
3F2504E0-4F89-11D3-9A0C-0305E82C3301
```

This text notation consists of 5 sets of data, each separated by a hyphen. The GUID consists of 32 characters plus 4 hyphens, for a total length of 36 characters.

There are two outbound message types.

task-started

The task-started message type is generated when an external task is started. A detailed description of the elements of the task-started message is available in Designing Integration with Adapter Development Kit.

```
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <task-started task-type="task">
       <task>
       .   .    .
```

task-canceled

The task-canceled message type is generated when the request which includes an external task is cancelled. If the user is not allowed to cancel a request once the external task has been performed (via the corresponding setting in the service's delivery plan), this message would never be generated. If, however, canceling the request is allowed, the transformation used in the agent responsible for the external task must be “smart enough” to handle both a task-started and task-canceled message. The transformation would need to test for the task-canceled message type and to send an appropriate message to the external system:

```
<xsl:if test="/message/task-started">
    <-- Original XSLT goes here/>
</xsl:if>

<xsl:if test="/message/task-canceled">
    <-- XSLT for the cancel message goes here/>
</xsl:if>
```

Inbound nsXML Message

Two types of operations are supported for inbound messages from the third-party system—requisition operations and service item operations. Requisition operations are used for the update of request data and task status. Service item operations are used for the addition, modification, deletion, and retrieval of service items associated with the request. Certain operations may be combined in one inbound message, known as a “Composite Message”. The details and restrictions for each operation are described in the following sections.

Requisition Operations

The third-party system may send one or multiple inbound messages for an external task by referencing the channel-id of the corresponding outbound message. The external task is completed when one of the take-action operations is sent and this allows the next task in the authorization/delivery plan to proceed.

take-action

A take-action message may be applied to an authorization or delivery task, to change the status of the task. The action attribute of the take-action tag identifies the action to be taken. Valid actions are summarized in Take-action Messages Table below.

Table 5-10 Take-action Messages Table

Action	Task Type	Description
done	Delivery task	Mark the delivery task as completed.
cancel	Delivery task	Cancel the delivery task.

Table 5-10 Take-action Messages Table

ok	Review task	Mark the review as completed.
reject	Authorization task	Reject the authorization.
approve	Authorization task	Approve the authorization.

When the last delivery task in a task plan is marked as done, the requisition is closed (completed). An approval task can be marked as Approved or Rejected, by setting the “action” attribute of the take-action tag to the corresponding value.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <take-action action="done"/>
</message>
```

send-parameters

Parameters are data elements that are bound to dictionary fields within the agent definition. The send-parameters message type allows one or more specified parameters to be updated which, in turn, updates the corresponding dictionary fields in the service. Using this type of inbound message is the preferred way for the external system to update dictionary fields used in a service request.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <send-parameters>
           <agent-parameter>
            <name>Status</name>
              <value>Resolved</value>
           </agent-parameter>
           <agent-parameter>
            <name>ResolvedBy</name>
            <value>Help Desk</value>
           </agent-parameter>
    </send-parameters>
</message>
```

add-comments

An add-comments message is used to add comments to the System Comments section of the requisition.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <add-comments>
        <comment>Test Comment</comment>
    </add-comments>
</message>
```

Service Item Operations

Service items are entities defined in Service Item Manager. Their lifecycles are associated with service requests—from the point the service item instances are provisioned, to the point when they are decommissioned. In the cases when the service item lifecycle events are handled by external systems, service item data can be synchronized with Lifecycle Center via Service Link service item, update, delete, and get messages. These message types are supported only through the web service-based Service Item Listener Adapter (see the Service Item Listener Adapter).

One or more service item types and service item instances can be included in these messages. Multiple service item operations cannot be combined in a message. In other words, , update, or delete operations have to be sent in separate inbound messages. Whenever an error condition is encountered, all the service item operations in the same message are rolled back.

Service item attributes of datetime type must be specified in the format YYYY-MM-DD HH:MI:SS or YYYY-MM-DD. All times are stored as UTC time.

Service item subscriptions can be included optionally at the time a service item instance is created. If no subscription information is provided in the operation, the item is assigned to the customer of the requisition and that person's Home Organizational Unit. If values for either the customer login ID or Organizational Unit name are specified in the subscription section of the message, those values are used to override the default service item assignment. For more details about subscription processing rules, see the Service Designer chapter in the *Cisco Prime Service Catalog Designer Guide*.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-LENVT60-17032</serviceItemAttribute>
            <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
            <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
            <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
            <serviceItemAttribute name="Memory">3</serviceItemAttribute>
            <serviceItemAttribute name="ManufactureDate">2009-04-15
12:00:00</serviceItemAttribute>
            <subscription>
               <loginID>jsmith</loginID>
               <ouname>Finance</ouname>
               <accountName>account1</accountName>
               <agreementName>agreement</agreementName>
            </subscription>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV200-02274</serviceItemAttribute>
            <serviceItemAttribute name="Model">Vostro 200</serviceItemAttribute>
            <serviceItemAttribute name="Brand">DELL</serviceItemAttribute>
            <serviceItemAttribute name="Price">755.99</serviceItemAttribute>
            <serviceItemAttribute name="Memory">4</serviceItemAttribute>
            <serviceItemAttribute name="ManufactureDate">2010-03-01
12:00:00</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
   </>
</message>
```

update

In update messages, omitting a service item attribute results in no change to the attribute value. When an attribute is explicitly specified in the message but contains no value, the value of the attribute for the service item is set to blank for text fields and zero for numeric fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channelId="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<update>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
            <serviceItemAttribute name="Memory">4</serviceItemAttribute>
            <subscription>
               <loginID>dcohen</loginID>
            </subscription>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV200-00394</serviceItemAttribute>
            <subscription>
               <loginID></loginID>
               <ouname></ouname>
               <accountName>account</accountName>
               <agreementName>agreement1</agreementName>
            </subscription>
         </serviceItemData>
      </serviceitem>
   </update>
</message>
```

delete

Delete service item requests require only the names for the service item type and instance. Additional service item attribute and subscription information is ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channelId="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<delete>
      <serviceitem>
         <name>LaptopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">LT-TOSH900-0021</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
      <serviceitem>
         <name>DesktopComputer</name>
         <serviceItemData>
            <serviceItemAttribute name="Name">DT-DELLV100-0394</serviceItemAttribute>
         </serviceItemData>
      </serviceitem>
   </delete>
</message>
```

getRequest

The getRequest operation is used for retrieving service item instances. The channel-id and topic-id attributes are optional, unlike the /update/delete service item requests. Each inbound message may contain only one getRequest operation and within it, only one service item type. There is no logging of the request as seen in the Service Link user interface.

Service item instances are retrieved according to the search filters specified in the request XML, using the service item attributes and subscription data (that is, Customer Login ID, Organizational Unit Name, Account ID, and Agreement). Up to five filters may be used in a getRequest and they are interpreted as AND joins. Search Filter Operators for getRequest Table below shows the operators that are supported in search filters.

Table 5-11 Search Filter Operators for getRequest Table

Datatype	Supported Filter Operators
STRING(32)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
STRING(128)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
STRING(512)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
INTEGER	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
MONEY	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
LONG INTEGER	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
DOUBLE FLOAT	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
DATE TIME	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo

The response data from the getRequest contains service item attribute names and values, as well as its subscription information. The maximum number of records returned in each getRequest operation is 100. The next set of records can be retrieved by specifying the ‘startRow’ and ‘count’ elements in the request. The startRow element indicates the beginning row number of the result set. The count element indicates the number of records to be returned. The ‘startRow’ and ‘count’ values are defaulted to 1 and 100, respectively, if they are absent in the request XML. The value for count is limited to 100 for performance reasons.

Here is an example of the getRequest XML:

```
<getRequest>
   <serviceItemType>LaptopComputer</serviceItemType>
   <startRow>1</startRow>
   <count>1</count>
   <subscription>
      <loginID>jsmith</loginID>
      <ouname>Finance</ouname>
   </subscription>
   <filters>
   <!--1 to 5 repetitions:-->
      <filter attributeName="Name" operator="Equals" value="LT-LENVT60-17032" />
      <filter attributeName="Price" operator="GreaterThan"  value="800"/>
   <filter attributeName="ManufactureDate" operator="GreaterThan" value="2004-04-10"/>
   </filters>
</getRequest>
```

Response for the above 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 xmlns:ns1="http://externaltask.api.newscale.com">
      <response channel-id="CHANNELID not retrieved"
xmlns="http://externaltask.api.newscale.com">
         <status-code>success</status-code>
         <status-message>Service item data read successfully.</status-message>
         <getResponse>
            <serviceitem>
               <name>LapTopComputer</name>
               <serviceItemData>
                  <serviceItemAttribute
name="Name">LT-LENVT60-17032</serviceItemAttribute>
                  <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
                  <serviceItemAttribute name="Memory">3</serviceItemAttribute>
                  <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
                  <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
                  <serviceItemAttribute name="ManufactureDate">Fri Apr 16 00:00:00
GMT+05:30 2004</serviceItemAttribute>
                  <subscription>
                     <loginID>jsmith</loginID>
                     <ouname>Finance</ouname>
                    <accountID>1<<accountID>
                    <accountName>tenantaccount<accountName>
                    <agreementID>1<agreementID>
                    <agreementName>agreement<agreementName>
                     <requisitionID>0</requisitionID>
                     <requisitionEntryID>0</requisitionEntryID>
                     <assignedDate>2012-07-20T05:21:29.187+05:30</assignedDate>
                     <submittedDate>2012-07-20T05:17:21.503+05:30</submittedDate>
                  </subscription>
               </serviceItemData>
            </serviceitem>
         </getResponse>
      </response>
   </soap:Body>
</soap:Envelope>
```

getDefinitionRequest

The getDefinitionRequest operation is used for retrieving the metadata or definition of a service item type. Like the getRequest operation, the channel-id and topic-id attributes are optional. Each inbound message may contain only one getRequestDefinition operation, and within it, only one service item type. There is no logging of the request as seen in the Service Link user interface.

Here is an example of the service item getDefinitionRequest:

```
<getDefinitionRequest>
   <serviceItemType>LaptopComputer<serviceItemType>
<getDefinitionRequest>
```

Response for the above 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 xmlns:ns1="http://externaltask.api.newscale.com">
      <response channel-id="" xmlns="http://externaltask.api.newscale.com">
         <status-code>success</status-code>
         <status-message>Service item definition read successfully.</status-message>
         <getDefinitionResponse>
            <serviceItemDef>
               <name>LaptopComputer</name>
```

```
                <classification>Laptops</classification>
                <displayName>LaptopComputer</displayName>
                <serviceItemProperty name="Name" type="string" />
               <serviceItemProperty name="Model" type="string" />
                <serviceItemProperty name="Brand" type="string" />
                <serviceItemProperty name="Price" type="real64" />
                <serviceItemProperty name="Memory" type="sint32" />
                <serviceItemProperty name="ManufactureDate" type="datetime" />
             </serviceItemDef>
          </getDefinitionResponse>
       </response>
    </soap:Body>
</soap:Envelope>
```

Composite Messages

The above message types can be combined in a single inbound message. Such a combination is known as a “composite” message. The order of execution matters; you must send the parameters or add comments before including the take-action tag, and place the service item operation tags last.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
    <add-comments>
          <comment>Task closed per override …</comment>
    </add-comments>
    <send-parameters>
           <agent-parameter>
            <name>Status</name>
               <value>Resolved</value>
           </agent-parameter>
    </send-parameters>
    <take-action action="done"/>
<update>
       <serviceitem>
          <name>LaptopComputer</name>
          <serviceItemData>
             <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
             <serviceItemAttribute name="Memory">4</serviceItemAttribute>
             <subscription>
                <loginID>dcohen</loginID>
             </subscription>
          </serviceItemData>
       </serviceitem>
    </update>
</message>
```

SIM Import Messages

A Service Item Manager (SIM) Import message type supports importing service item and standards definitions and data from an external system into Service Catalog. Unlike the service item /update/delete operations, SIM import is based on the File Adapter protocol which polls for incoming files located in a specific directory. In addition to service item instance operations, SIM Import also supports the maintenance of service item groups and service item types. For details on Service Item Manager imports, see the *Cisco Prime Service Catalog Designer Guide*.

Inbound JSON Message

Starting from the 12.0.1 release, Prime Service Catalog supports JSON inbound messages using FTL transformation. The JSON inbound message is supported only for File and HTTP/WS adapters. To achieve this write a FTL transformation which maps the custom JSON to nsXML. For more details on writing FTL transformation see *http://freemarker.org/*.

For more details on HTTP/WS Adapter and File Adapter, see HTTP/WS Adapter, page 5-51 and File Adapter, page 5-49.

Following are the operations that can be performed on a requisition using HTTP/WS adapter and File adapter:

- Add Comments
- Adding Parameters
- Updating Values
- Completing Delivery Task
- Review Task
- Approve Task
- Reject Task

Each of the operations are explained in detail below:

add-comments

An add-comments message is used to add comments to the System Comments section of the requisition.

Example JSON Payload

```
{
"key":"CF04ECC6-1868-4D7B-9C83-463054C27453",
"message":"Adding comment to requisition”,
}
```

Corresponding Example Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="${root.key}">
<add-comments>
<comment>${root.message}</comment>
</add-comments>
</message>
```

Where,

- `${root}` represents *org.json.JSONObject* of the incoming JSON string.
- `${root.key}` corresponds to the channel-id/key attribute from the JSON payload.
- `${root.message}` corresponds to the message attribute to be added to the requisition.

add-parameters

Parameters are data elements that are bound to dictionary fields within the agent definition. The add-parameters message type allows one or more specified parameters to be added which, in turn, add the corresponding dictionary fields in the service. Using this type of inbound message is the preferred way for the external system to update dictionary fields used in a service request.

Example JSON Payload

```
{
"key":"150A3FB7-5E2B-486B-8288-D197506A9762",
"param":"Param1",
"value":29
}
```

Corresponding Example Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="${root.key}">
<send-parameters>
<agent-parameter>
<name>${root.param}</name>
<value>${root.value}</value>
</agent-parameter>
</send-parameters>
</message>
```

Where,

- `${root}` represents *org.json.JSONObject* of the incoming string.
- `${root.key}` corresponds to the channel-id/key attribute from the JSON payload.
- `${root.param}` corresponds to the parameter attribute.
- `${root.value}` is the value of the parameter to be added.

update-values

The update-value message type allows one or more specified parameters to be updated with a new value.

Example JSON Payload

```
{
"key":"150A3FB7-5E2B-486B-8288-D197506A9762",
"param":"D.Age",
"value":65
}
```

Corresponding Example Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="${root.key}">
<update-data>
<data-value multi-valued="false">
<name>${root.param}</name>
<value>${root.value}</value>
</data-value>
</update-data>
</message>
```

Where,

- `${root}` represents *org.json.JSONObject* of the incoming string.
- `${root.key}` corresponds to the channel-id/key attribute from the JSON payload.
- `${root.param}` corresponds to the parameter attribute.
- `${root.value}` is the value of the parameter to be added.

update-multiple-values

The update-value message type allows multiple specified parameters to be updated with a new value.

Example JSON Payload

```
{
"key":"EE44AB32-6940-4423-A3A8-46EFDE8ED054",
"keypair":{"param":"D.Country",
                   "values":[{"value":"China"},
               {"value":"US"}]
                }
}
```

Corresponding Example Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="${root.key}">
<update-data>
<data-value multi-valued="true">
<name>${root.keypair.param}</name>
<#assign x=root.keypair.values.length()>
<#list 1..x as i>
<value>${root.keypair.values.get(.vars.i - 1).get("value")}</value>
</#list>
</data-value>
</update-data>
</message>
```

take-action

A take-action message may be applied to an authorization or delivery task, to change the status of the task. The action attribute of the take-action tag identifies the action to be taken. Valid actions are summarized in table below..

Table 5-12 Take-action Messages Table

Action	Task Type	Description
done	Delivery task	Mark the delivery task as completed.
ok	Review task	Mark the review as completed.
reject	Authorization task	Reject the authorization.
approve	Authorization task	Approve the authorization.

When the last delivery task in a task plan is marked as done, the requisition is closed (completed). An approval task can be marked as Approved or Rejected, by setting the “action” attribute of the take-action tag to the corresponding value.

Example JSON Payload

```
{
"key":"19E2A5E6-40EA-43AC-9D73-5FDEFF8C28FB",
"action":”ok”, "action":”approve”, "action":”reject
}
```

Corresponding Example Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="${root.key}">
<take-action action="${root.action}">
</take-action>
</message>
```

Where,

- `${root}` represents *org.json.JSONObject* of the incoming string.
- `${root.key}` corresponds to the channel-id/key attribute from the JSON payload.
- `${root.action}` is the task type.

Transformations and nsXML

Outbound nsXML messages will typically be quite large and complex, often in excess of 500 KB. Although it is not mandatory to use transformations to alter the message format, it is unlikely that external systems would be configured to read nsXML. Consequently using transformations to alter the outbound message formats is normally unavoidable.

However as formats for inbound messages will probably be negotiated with those responsible for the third-party system, it is quite possible that a specification could be agreed that aligns closely to the nsXML message formats. If this is the case, the Inbound transformation could be much simpler than the corresponding outbound one.

Although we refer to XSL Transformations (XSLT), the technology used is actually called eXtensible Stylesheet Language and also includes XPATH. XPATH is a language for finding information and navigating through elements and attributes in an XML document. XPATH includes built-in functions for string values, numeric values, date and time comparison, sequence manipulation, Boolean values and other methods.

Monitoring Service Link Transactions

There are multiple ways to monitor Service Link usage:

- The Service Link home page shows a graph of message volume over the last 30 days and provides Common Tasks and the View Transactions tab to access other monitoring options.

- The option to view Recent Failed Messages, also on the Service Link home page, shows all messages that could not be delivered.
- The option to view Messages, accessible from the View Transactions tab, shows all messages sent to or received by Service Link, and allows administrators to filter and search to show messages of interest.
- The option to view External Tasks, accessible from the View Transactions tab, shows all tasks that remain ongoing because a Service Link message could not be delivered, and allows administrators to filter and search to show tasks of interest.

All Service Link monitoring/administration pages are displayed using configurable "data tables". The appearance of these tables (the columns displayed, the width of each column and the order in which data is presented) can be customized. In addition, Filter and Search capabilities allow administrators to view only those rows which are of interest.

Rate limiting easily throttle REST calls, and prevents distributed denial-of-service (DDoS) security threats from malicious users. Application level rate limiting also effectively balances load. See Configuring Rate Limits for REST API Requests.

Viewing Messages from the Service Link Home Page

The Recent Failed Messages pane of the Service Link home page displays Service Link messages that could not be delivered to their destination within the past 30 days. By default, messages are displayed in reverse chronological order based on the date and time when they were sent.

Figure 5-19 Failed Messages

Recent Failed Messages

Message Type	Req ID	Agent	Task Subject	Date	Resent On	Status Text
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:55 PM		Inbound Message I
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:54 PM		Inbound Message I
Composite	181	Douglas_DB_Agent	configurable task d...	12/05/2011 03:51 PM		Inbound Message I
Composite	180	Douglas_DB_Agent	DT1	12/05/2011 03:51 PM		Inbound Message I
Composite	179	Douglas_DB_Agent	configurable task d...	12/05/2011 03:50 PM		Inbound Message I
Composite	178	Douglas_DB_Agent	configurable task d...	12/05/2011 03:49 PM		Inbound Message I
Composite	177	Douglas_DB_Agent	configurable task d...	12/05/2011 03:47 PM		Inbound Message I
Unknown		Douglas_DB_Agent		12/05/2011 03:21 PM		Unknown Channel
Take Action	172	Douglas_DB_Agent	configurable task d...	12/02/2011 02:58 PM		Inbound Message I
Take Action	171	Douglas_DB_Agent	configurable task d...	12/02/2011 02:56 PM		Inbound Message I
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:06 PM		Inbound Message I
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:05 PM		Inbound Message I
Take Action	169	Douglas_DB_Agent	configurable task d...	12/02/2011 02:05 PM		Inbound Message I

Click one of the column links in the Failed Messages grid to view associated information:

Table 5-13 Service Link Failed Messages Clickable Columns

Column	Link
Message Type	Message details on Service Link Message Details popup pages
Req ID	Requisition details
Agent	Agent details in Service Link Agents page

The Messages page, available from the View Transactions tab, allows you to view all messages, both inbound and outbound, regardless of their status; to explicitly filter the messages that appear on the page; and to search messages which fit specified search criteria.

Viewing Messages

The Messages page displays all or selected Service Link messages, depending on which filters have been set. By default, completed messages are not displayed. To display the Messages page, click the **View Transactions** tab from the Service Link home page. Then click the **Messages** subtab. The View Failed Messages link in the Common Tasks area of the Service Link home page also displays the Messages page, with a filter set to show only messages with a status of "Failed".

The Messages page appears, as shown below.

Figure 5-20 Messages Page

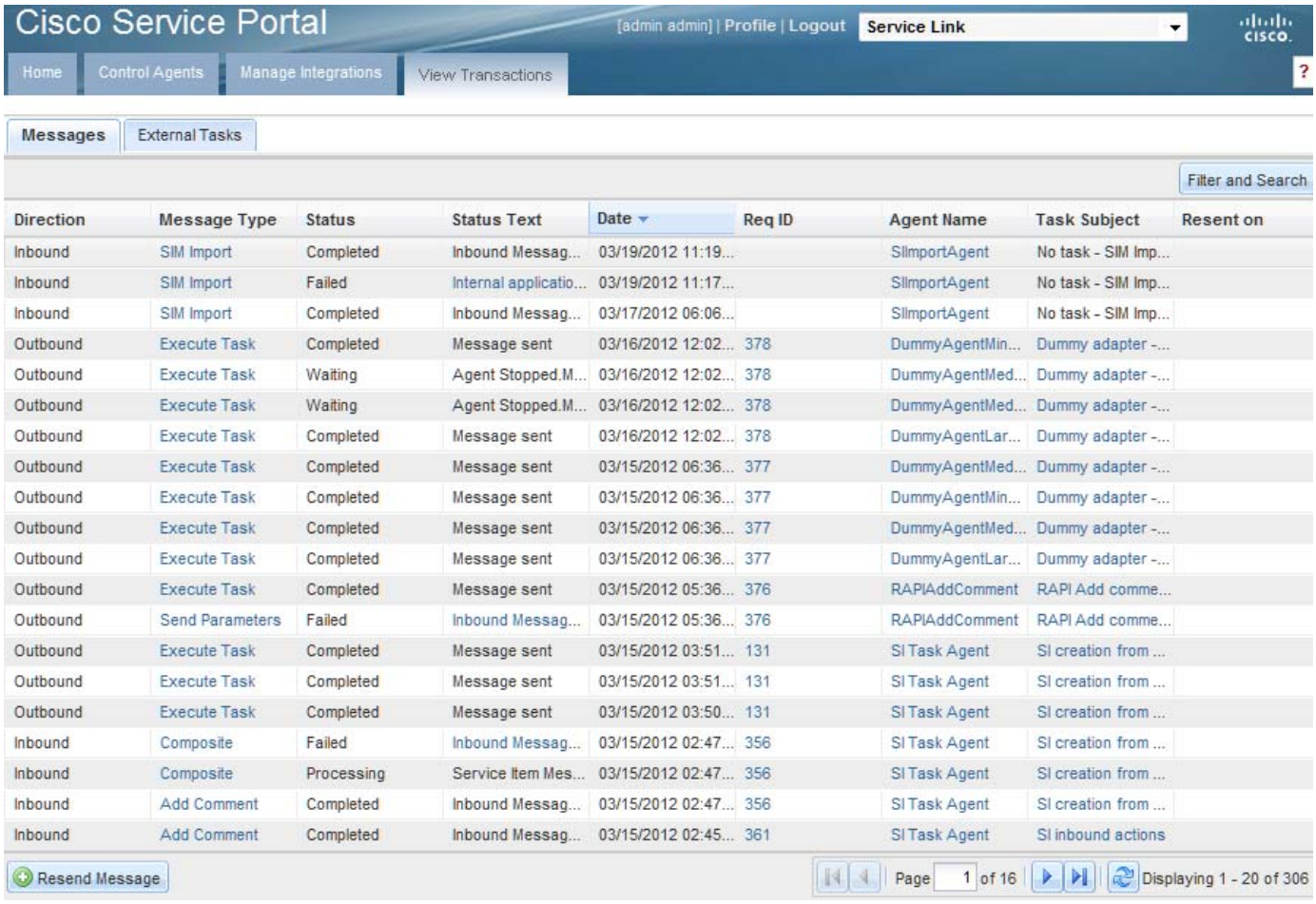


Table 5-14 Service Link Messages Clickable Columns

Column	Link
Message Type	Message details on the Service Link Message Details popup pages.

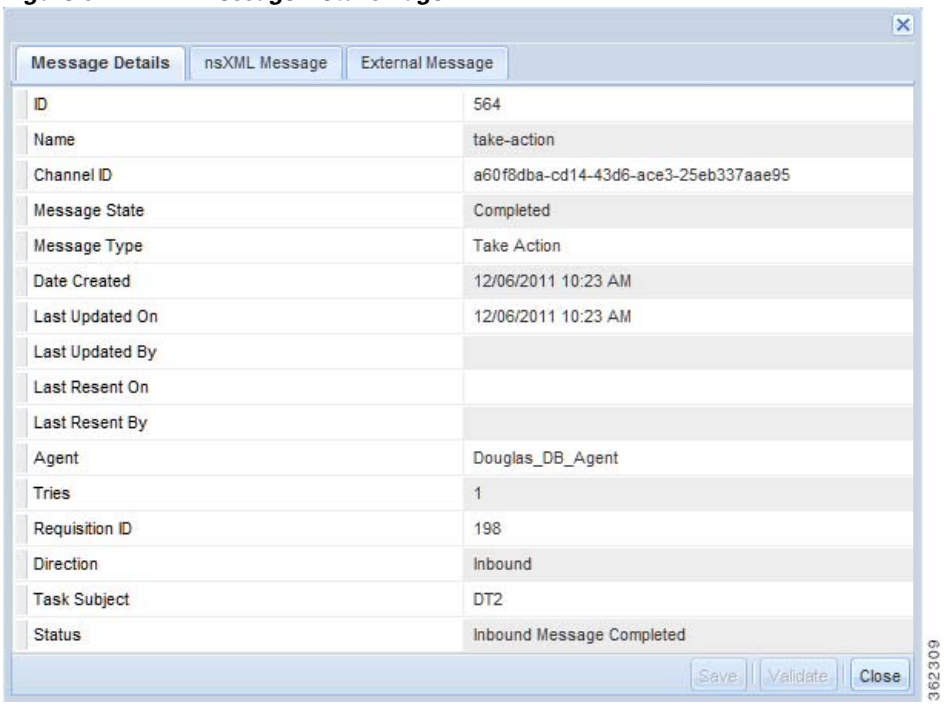
Table 5-14 Service Link Messages Clickable Columns

Status Text	For failed messages, a link is available to the error messages written to the adapter-specific log file and the server log. See the Service Link Troubleshooting and Administration for more details.
Req ID	Requisition details.
Agent	Agent details in the Service Link Agents page.
Task Subject	Task details in Service Manager.

Message Details

The Message Details popup pages allows you to view both the Service Catalog and external messages. This page also displays the channel Id, which uniquely identifies the task in this requisition. You can use this Id when working out issues with the third-party system.

Figure 5-21 Message Details Page



Click one of the tabs on the Message Details popup page to view associated information.

Table 5-15 Service Link Message Details Subtabs

Column	Link
Message Details	Details about the message.

Table 5-15 Service Link Message Details Subtabs

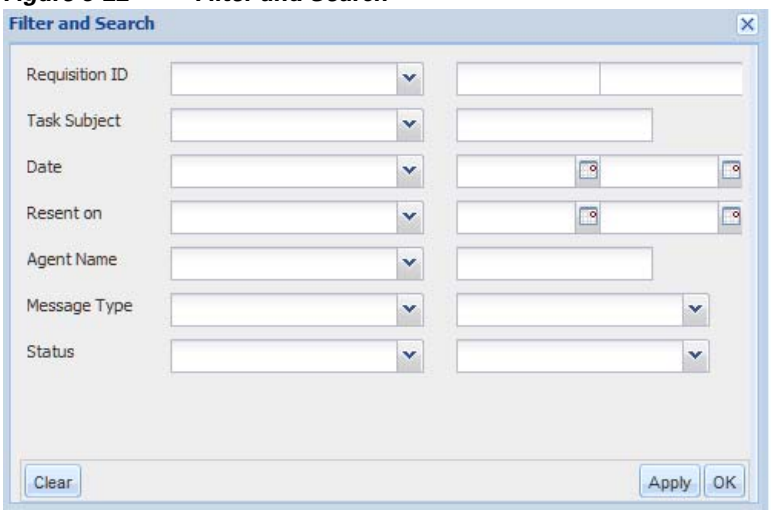
nsXML Message	For outbound messages, the message produced by the Business Engine, to be processed (transformed) by the Service Link agent; for inbound messages, the message received from the external system, transformed by the agent transformation (if any), and to be processed by the Business Engine.
External System Message	For outbound messages, the message after the transformation associated with the agent has been applied; for inbound messages, the message as it was received from the external system.

Filter and Search

You can use the search functionality to view a subset of messages, for example, all messages with a **Failed** status. Search allows you to specify one of the columns in the Messages window as the search target and to select or type a value to be matched.

Click **Filter and Search** (at the top of the Messages page).

Figure 5-22 Filter and Search



The Filter and Search dialog box also allows you to:

- Filter a particular column by using any relational operator appropriate for the semantics of that column. For example, a date range may be chosen, or any status not equivalent to the specified status can be chosen.
- Filter by the logical ‘AND’ of all criteria specified for columns.

The Filter and Search dialog box is non modal. You can fill out the desired criteria and click **Apply** to view the results of the current settings. If required, simply adjust the settings and **Apply** again. Remember that you can also display the messages in ascending or descending order by any column, or change the columns that are displayed by using the techniques.

Resending Failed Messages

During Service Link development, you may generate many messages that fail to be delivered because of errors in the agent or transformation configuration. These messages should not be resent. Similarly, messages generated via a Service Item Import task should not be resent—the import file format should be adjusted, and the import task tried again.

In a production environment, however, messages may fail to be delivered because of an outage of the external system or other external factor that can be corrected. Once the cause of the delivery failure has been corrected, failed messages can be resent.

To resend failed messages:

Step 1 In the Messages page of the View Transactions tab, click the row containing the Failed message or messages.

Step 2 In the bottom left corner of the Messages page, click **Resend Message.**

Service Link will attempt to resend the message to its designated destination. If the resend succeeds, the message status and date are updated, and the resend date is recorded and displayed in the Resent On column.

Transformations are not reapplied while resending a message. The agent tries to send the already transformed message to its destination.

Resending of failed inbound messages for service item operations is not supported. The process attempts to retry task actions. Hence the destination for those messages is the Business Engine, not the Service Item import processor.

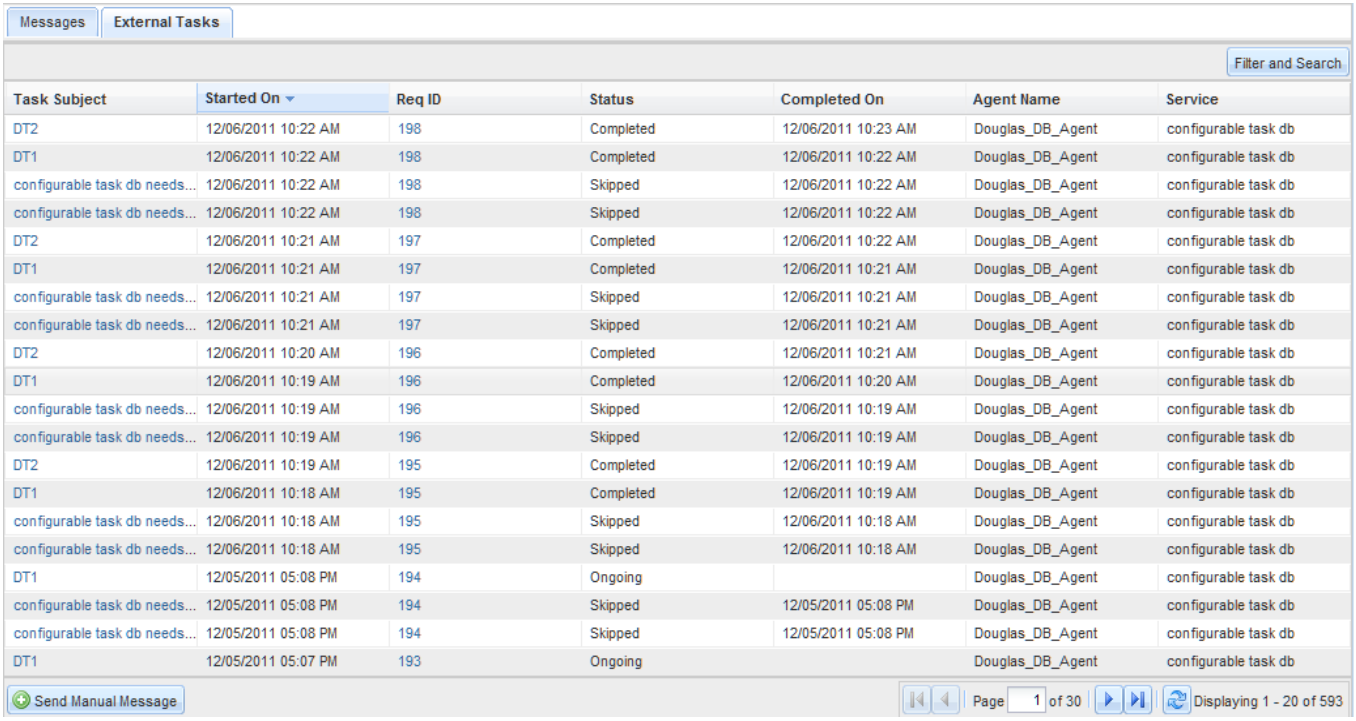
Viewing External Tasks

To view External Tasks:

Step 1 From the Service Link home page, click **View Transactions**. Then click the **External Tasks** subtab.

The External Tasks page appears, as shown below.

Figure 5-23 External Tasks Page



Step 2 Click one of the following column links to view associated information.

:

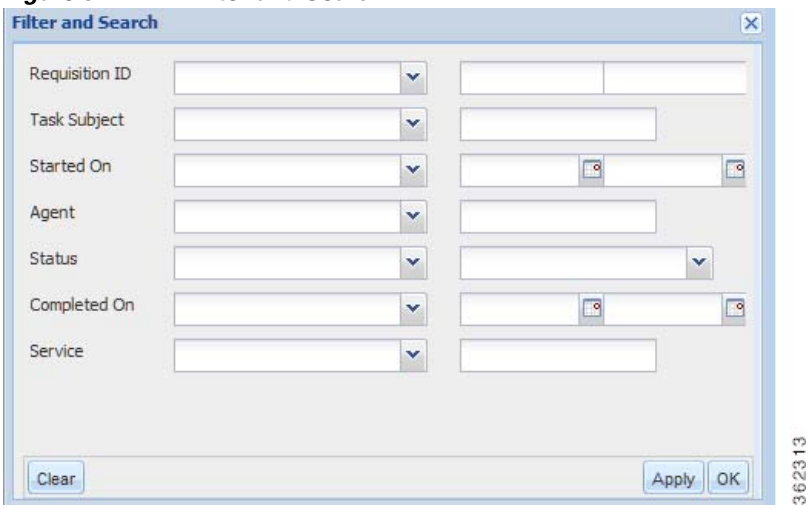
Table 5-16 Service Link External Tasks Clickable Columns

Column	Link
Task Subject	Task details in Service Manager
Req ID	Requisition overview in My Services
Agent Name	Agent details in the Service Link Agents page

Filter and Search

Like the Messages display, the External Tasks page offers the ability to customize the columns and order of data displayed in the data table and to filter and search on that data.

Figure 5-24 filter and Search



Sending a Manual Message

A task that has been started and is expecting to receive an inbound message is in an “Ongoing” state. The incoming message will typically update the task or change its status. No subsequent tasks in the requisition's delivery plan can be performed until a message is received and the task is completed. If you suspect (or can confirm by conferring with administrators of the external system) that the expected message has already been sent, but has somehow been “lost”, you can emulate receipt of the message by sending a manual message.

Manual messages cannot be used to emulate failed service item operations.

Note Use this feature carefully. This feature overrides all the communication protocols in the system, and using it may leave artifacts in the third-party system to which Service Link may no longer be able to respond. Also, if you use this feature to cancel a requisition, for example, Service Link will not notify the interested parties, so you will have to follow up on your own.

To send a manual message to the Business Engine:

Step 1 From the Service Link Home page, click **View Transactions**. Then click **External Tasks**.

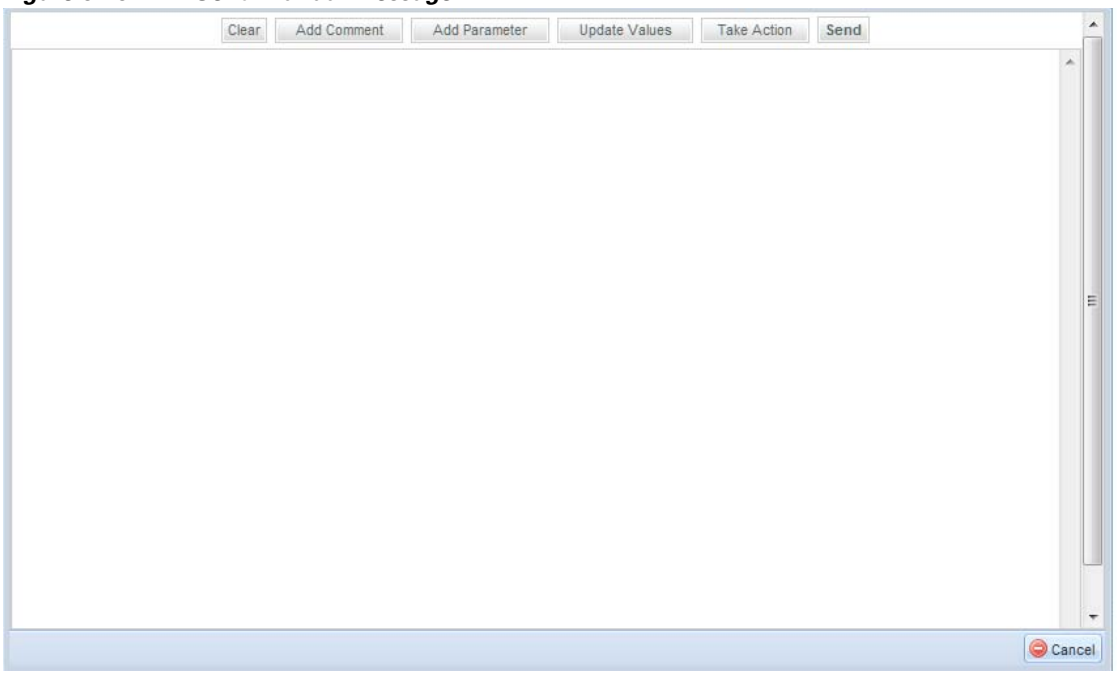
The External Tasks page appears.

Step 2 In the bottom left corner of the External Tasks page, click the line containing the task for which you want to send a manual message.

Step 3 Click **Send Manual Message**.

The Send Manual Message dialog box appears, as shown below.

Figure 5-25 Send Manual Message



Step 4 Click the button corresponding to the type of message you want to send—**Add Comment**, **Add Parameter**, **Update Values** or **Take Action**.

Table 5-17 Action and Description

Action	Description
Add Comment	Send an add-comments message, to add a system comment to the requisition.
Add Parameter	Send a send-parameters message. Modify one or more inbound agent parameter values and the value of the corresponding dictionary field to which the parameter is bound.
Update Values	Send a message to modify the contents of the specified dictionary field. (This message type is provided primarily for backward compatibility with previous versions; field contents should typically be updated via inbound agent parameters.)
Take Action	Send a take-action message. Mark the task as done (completed) or canceled; approve or reject an authorization; or mark a review as OK.

Step 5 Respond to the associated popup dialog boxes (turn off your popup blocker) for the message type chosen. This will populate the message window with a well-formed XML message of the appropriate type. An <add-comments> message will also be included, to indicate that this message was not received through normal channels, but manually generated.

Step 6 If desired, you may edit the generated message. When you have constructed the entire message, click **Send**. An inbound message is sent to the Business Engine.

Republishing Service Link Messages

In the rare occasion of extended outage or incorrect configurations of the Service Link application, external tasks might not have corresponding outbound messages created in Service Link.

Once the underlying issue is resolved in Service Link and the application is up and running again, the problem external tasks can be republished to Service Link to allow the outbound messages to be created and the delivery process to resume.

To republish outbound messages:

Step 1 From the Service Link Home page, click **View Transactions**. Then click **Message Republish**.

Step 2 On the left-hand pane, enter the Requisition ID for the requests which have one or more missing outbound Service Link messages. All authorization and delivery tasks associated with the requisition are evaluated and only those tasks that require republishing are processed for outbound message creation. Up to 20 requisitions can be entered at a time.

Step 3 Click **Republish**.

Step 4 Review the processing status on the right-hand pane once the republish process is completed.

Managing Service Link Adapters

All Service Link adapters support nsXML as the data exchange format. For more information about the nsXML format, see Designing Integration with Adapter Development Kit.

All poller-based adapters support processing on only one message per invocation.

The Service Link Adapters installed in all application instances are:

- Dummy Adapter
- Database Adapter
- File Adapter
- HTTP/WS Adapter
- JMS Adapter
- MQ Adapter
- Service Item Listener Adapter
- Web Service Listener Adapter
- Cloud Resource Manager Adapter

In addition to these adapters, Service Link supports an Auto-Complete Adapter.

Additional adapters may be installed and configured using the Service Link Adapter Development Kit (ADK). Any such custom adapters also appear on the Adapters page, and their properties may be reviewed. For details on building and installing custom adapters, see Designing Integration with Adapter Development Kit.

The following sections describe these adapters.

Auto-Complete Adapter

The Auto-Complete adapter allows an agent to send an outbound message and to mark the task as complete without waiting to receive an acknowledgement from the external system. If the outbound message is successfully sent (for example, a file is written to the specified directory by an outbound file adapter), the auto-complete adapter generates an incoming message for the same task. That incoming message has the message type “take-action”. This message is processed normally by the Business Engine, marking the action as done and completing the external task.

Dummy Adapter

The Dummy Adapter is a placeholder. It can be used in several processing scenarios:

- Using the dummy adapter as the inbound adapter allows an external task initiated by Service Link to remain in Ongoing status.
- Using the dummy adapter as an outbound adapter and the auto-complete adapter as the inbound adapter allows service designers to implement Auto-Complete Agents in external tasks. The task can then be used in part of the workflow, for example, to generate an email to participants, or to close a request which has no other tasks. This combination can also be used to verify if communications between Service Catalog and Service Link are working correctly.

Database Adapter

The Database (DB) adapter uses one or more tables in a database to pass data between Service Catalog and external applications.

Database Connection

Inbound and outbound database adapters are capable of communicating with any JDBC-compliant relational database that supports ANSI-standard SQL. Valid connection criteria must be provided, as well as the JDBC URL, and a database driver. If the external database is SQLServer or Oracle, Cisco-provided drivers may be used. Drivers available from Cisco are:

```
com.microsoft.sqlserver.jdbc.SQLServerDriver
oracle.jdbc.OracleDriver
```

The JDBC URL has the format:

```
jdbc:sqlserver://<host>:<port>;databaseName=<db_name>;selectMethod=direct;sendStringParame
tersAsUnicode=true
jdbc:oracle:thin:@<host>:<port>:<SID>
jdbc:oracle:thin:@//<host>:<port>/<service_name>
```

where

- dbtype is sqlserver or oracle
- host is the name of the database server
- port is the port through which to connect to the database; typically 1433 for SQLServer and 1521 for Oracle
- The database name must be specified for SQLServer; the SID (System Identifier) and service name must be specified for Oracle

EXAMPLES:

```
jdbc:sqlserver://mysqlserver.cisco.com:1433;databaseName=RequestCenter;selectMethod=direct
;sendStringParametersAsUnicode=true

jdbc:oracle:thin:@myoracle.cisco.com:1521:DEVRC

jdbc:oracle:thin:@//myoracle.cisco.com:1521/PRODRC
```

A user-supplied driver may be used if supporting jar files are installed on the directory ISEE.war/WEB-INF/lib in the Service Catalog directory structure.

Step 1 Obtain the appropriate third-party JDBC driver. For example, the Sybase JDBC Driver can be downloaded from Sybase's website.

Step 2 Copy any required jars to the ISEE.war/WEB-INF/lib folder.

Step 3 Modify the Agent settings to use the custom driver and the correct JDBC URL format. For example, the format for the JDBC URL for the Sybase driver is:

```
jdbc:sybase:Tds:host:port/database
```

Step 4 Restart the Service Link and Service Catalog services.

The format of the JDBC Url may also be influenced by the application server on which Service Link is deployed. For example, a possible JDBC URL to establish a connection to SQLServer database from a WebSphere application server would be:

```
jdbc:sqlserver://<host>:<port>;databaseName=<db_name>;selectMethod=direct;sendStringParame
tersAsUnicode=true
```

Inbound Properties

When the database adapter is used as an inbound adapter, the agent properties include a SQL statement to be executed against the specified database connection. The SQL is typically a select command which returns a set of rows. These rows are then formatted into an external XML message. The message must be transformed via an inbound transformation (specified in the agent) into a valid nsXML inbound message. That message is, in turn, passed to the Business Engine. If the Business Engine finds an open task identified by the Channel ID specified in the inbound message, the inbound message is processed and the specified action taken.

The Property sheet for the database inbound adapter prefixes the property names given below with “DBInboundAdapter”.

Table 5-18 DB Adapter Inbound Properties

Property	Description
DBPassword	Password for the specified user.
DBUserName	Database user name.
InboundSql	The SQL statement to be executed for the inbound transaction. This should be a SELECT statement that returns a set of rows. Transactional SQL (that is, a procedure) is not supported.

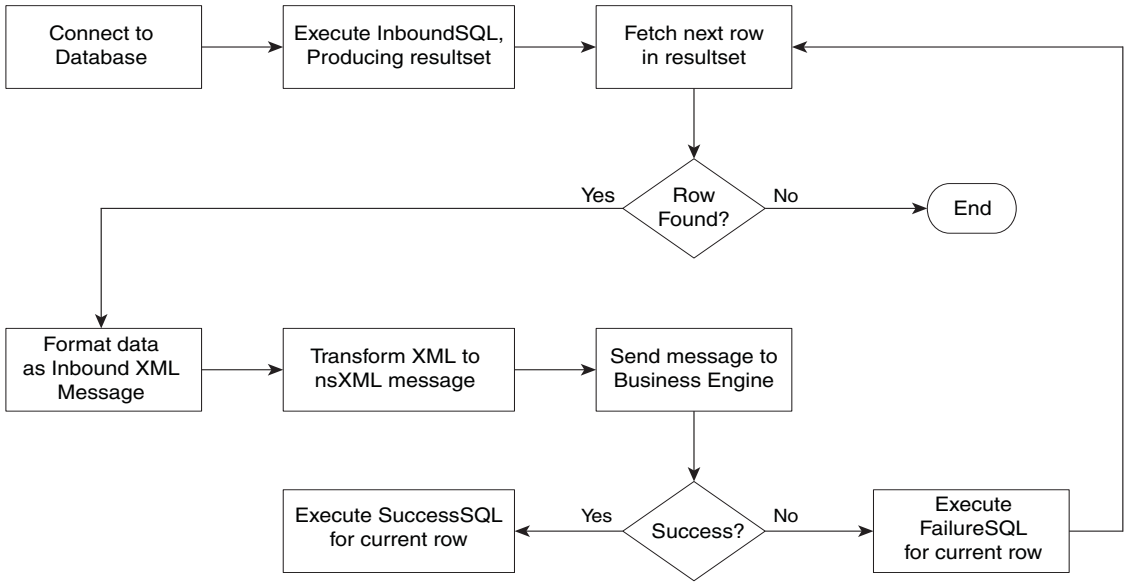
Table 5-18 DB Adapter Inbound Properties (continued)

Property	Description
InboundSuccessSql	The SQL to be executed on success of the inbound transaction, typically a SQL update or delete statement which marks the current row as successfully processed.
InboundFailureSql	The SQL to be executed on failure of the inbound transaction typically a SQL update or delete statement which marks the current row as successfully processed.
JDBCUrl	JDBC URL to connect to the database.
JDBCDriverClass	The class name of the driver to be used to connect to the database.

Inbound Message and Work Flow

The process flow for the inbound database adapter is shown below:

Figure 5-26 Process Flow for Inbound Database Adapter



For each row in the result set, the adapter generates an XML message with the following structure:

- The root element of the message is <inbound-results>.
- The required child element is <row>. Each message has exactly one <row> element.
- Each <row> element has multiple <column> elements, one for each column included in the InboundSQL statement specified for the adapter.
- The <row> element has attributes for the column name (<name>) and JDBC data type (<type>; 12 for character and 1 for numeric).
- The value of each <column> element is the value returned for the corresponding column in the SQL statement.

For example, the SQL statement

```
SELECT channel-id, task, status, processType FROM rcInterface
  WHERE status = 'UPDATED'
```

might yield an XML stream like the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<inbound-results>
    <row>
       <column name="channel-id" type="12" >
            "3F2504E0-4F89-11D3-9A0C-0305E82C3301"
       </column>
       <column name="task" type="12" >Task</column>
       <column name="status" type="12" >UPDATED</column>
       <column name="processtype" type="1" >null</column>
    </row>
</inbound-results>
```

A transformation must then be applied to this XML stream to produce a valid nsXML inbound message. For example, a transformation which would complete an ongoing task might include the following code:

```
<xsl:template match="/inbound-results/row">
   <xsl:variable name="status"  select="column[@name='status']" />
   <xsl:choose>http://www.w3schools.com/xsl
      <xsl:when test="$status='Complete'">http://www.w3schools.com/xsl
      <message>http://training2.cisco.com/RequestCenter
      <xsl:attribute name="channel-id">
          <xsl:value-of select="column[@name='channel-id']" />
      </xsl:attribute>http://training2.cisco.com/ServiceLink
      <take-action action="done" />
      </message>
   <xsl:otherwise>
     .       .          .
```

The Business Engine processes the resultant nsXML message. If the message was applied successfully, the SuccessSQL specified in the agent is executed. The SuccessSQL typically updates the columns in the source table that caused the row to be selected for processing, so that the row will not be found again in the next polling interval. To specify that Service Link should update the current row, identify the column or columns that comprise the row's unique identifier. Those columns must have been included in the inbound SQL statement. For example:

```
UPDATE rcInterface
   SET status = 'DONE'
 WHERE channel-id = #channel-id#
```

Similarly, the FailureSQL is executed if the Business Engine failed to apply the nsXML message-for example, if an error occurred during processing of the message. The FailureSQL typically updates the status of the current row to indicate that the row was not correctly processed. For example:

```
UPDATE rcInterface
   SET status = 'FAILED'
 WHERE channel-id = #channel-id#
```

Outbound Properties

When the database adapter is used as an outbound adapter, it provides a “staging table” style interface between Service Catalog and the external system. The nsXML outbound message which is provided to the agent by the Business Engine must be transformed into an external message containing one or more SQL statements. These SQL statements are then executed in the specified database, using the specified connection.

The Property sheet for the DB outbound adapter prefixes the property names given below with “DBOutboundAdapter”.

Table 5-19 DB Adapter Outbound Properties

Property	Description
DBPassword	Database User password
DBUserName	Database user name
JDBCUrl	JDBC URL to connect to the database
JDBCDriverClass	The class name of the specific driver to be used to connect to the database

Outbound Message and Workflow

The outbound message produced by the XSLT transformation must have the format:

```
<?xml version="1.0" encoding="UTF-8"?>
<outbound-message>
    <execute-sql-list>
        <execute-sql> SQLStatement
       </execute-sql>
    </execute-sql-list>
</outbound-message>
```

The message can contain multiple SQL statements, each within an <execute-sql> tag. These statements typically insert or update rows in SQL tables. Any SQL statement supported by the JDBC driver specified for the adapter can be used. Stored procedures (in SQLServer Transact-SQL or Oracle PL/SQL) are not supported, although the SQL statement can include user-defined functions. Since each external task is uniquely identified by a Channel ID, the target table for the outbound SQL statement must include a column for the Channel ID in order for that task to be updateable by an inbound message.

File Adapter

The File Adapter provides support for reading files from a specified directory or writing files to a specified directory.

- The adapter cannot be configured for processing files from multiple directories or sub directories of a specified directory.
- The oldest file of a set of files of a directory would be processed by an inbound file adapter when invoked.
- Only one agent should be configured for a specified directory to process the files.
- The directories (locations) specified must be on the file system of the application server where Service Catalog is installed or accessible from the application server. All directories must be on the same physical device, since files are moved from one directory to another as Service Link processing proceeds.

File Adapter Inbound Properties

Following are the properties with the default values for the File Adapter.

The Property sheet for the File inbound adapter prefixes the property names given below with “FileInboundAdapter”.

Table 5-20 File Adapter Inbound Properties

Property	Description
BackupLocation	Location where the files are backed up after they have been processed, if the Final Resolution or OnError property is "Preserve".
BackupSuffix	File extension for the backup files; default is .bak.
FileLocation	Location (directory) that is polled for inbound files to be read; a unique location should be used for each inbound file adapter.
FileNameDateFormat	Date format for the files; default is .yyyyMMddHHmmssSSS.
FinalResolution	Action to take on the file after transaction completion. Options are: • Preserve – Moves the file to the backup location • Delete – Deletes the file default is Preserve.
OnError	Action to take on the file when an error occurs. Options are: • Preserve – Moves the file to the backup location • Delete – Deletes the file default is Preserve.
TempLocation	Temporary folder used for processing inbound files.

File Adapter Outbound Properties

The outbound file adapter produces an XML file on the specified file location. The name of the file contains the channel-id, a unique identifier for the external task that included the agent and created the message. The file name ends with the date format specified as an outbound property.

The Property sheet for the File outbound adapter prefixes the property names given below with "FileOutboundAdapter".

Table 5-21 File Adapter Outbound Properties

Property	Description
BackupLocation	Outbound files backup location; may be any valid file system directory accessible from the application server.
BackupSuffix	File extension for the backup files; default is .bak.
ConflictResolution	Action to take in case of conflict in the outbound file names. Options are: • Preserve – Moves the file to the backup location • Delete – Deletes the file Default is Rename.
FileLocation	Location to which the file is written; may be any directory accessible from the application server.
FileNameDateFormat	Date format for the file name; default is: .yyyyMMddHHmmssSSS.

Table 5-21 File Adapter Outbound Properties

OnError	Action to take on the file when an error occurs. Options are: • Preserve – Moves the file to the backup location • Delete – Deletes the file Default is Preserve.
TempLocation	Temporary folder used for processing the outbound file.

HTTP/WS Adapter

The HTTP/WS adapter is used to send or receive HTTP requests or web service requests and responses. HTTPS is also supported.

The use of a proxy server in connecting to the web service is not supported.

When used to call web services, only synchronous calls are possible. The outbound transformation must be written in a way to produce an external message that is compliant to the web service standard. For SOAP-based web services, appropriate SOAP header and SOAP body elements should be included

Outbound Properties

The HTTP/WS Adapter outbound properties specify the behavior of the outbound adapter.

The HTTP/WS Adapter Outbound Properties table for the http/ws outbound adapter prefixes the property names given below with “HttpOutboundAdapter”.

Table 5-22 HTTP/WS Adapter Outbound Properties

Property	Description
WsdlURL	The URL of the wsdl that includes the operation to be performed; used only with the Integration Wizard.
WsdlOperation	The operation to be performed by the web service; documentation only except when using the Integration Wizard. A drop-down list of all operations included in the specified WSDL is available.
RoutingURL	URL to route all outbound messages to be posted; the web service end point.
AcceptUntrustedURL	Option to allow accepting untrusted certificates from external systems; default is true.
ContentType	Content type; default is text/xml; charset=ISO- 8859-1.
TimeOut	Timeout to get the http url connection; default is 180,000 microseconds.
ProcessResponse	Option to treat the result of the post or the response to a SOAP message as an inbound message; default is false.

Table 5-22 HTTP/WS Adapter Outbound Properties (continued)

Property	Description
RequestHeaders	Any custom header parameters that must be included in the HTTP request header, typically in the form of name-value pairs, with parameters separated by ampersands (&). For example, a SOAPAction might be entered in the format: SOAPAction=Op A SOAPAction and a custom header called "referrer" might be entered in the format: SOAPAction=Op&referrer=www.test.com
AuthenticationScheme	The type of authentication to be used to request the web service or post to the URL; options are basic, anonymous, digest, or NTLM; details explanations of these options are given below.
AuthenticationScopeHost	The host to which the authentication credentials apply. May be left empty if credentials are applicable to any host.
AuthenticationScopePort	The port to which the authentication credentials apply. May be left empty if credentials are applicable to any port.
AuthenticationScopeRealm	The realm to which the authentication credentials apply. May be left empty if credentials are applicable to any realm.
Username	User name for authentication to the target system.
Password	Password for authentication to the target system.
Host	Host credential that may be required for some authentication schemas (like NTLM).
Domain	Domain credential that may be required for some authentication schemas. NTLM does not use the concept of realms. The authentication domain should be specified as the value of the 'realm' attribute. May be left empty if credentials are applicable to any domain.
SaveRefField	Boolean used when *ProcessResponse* is true. Indicates that the response will contain a field which the external system uses as a unique identifier (or TopicID) for this task; see the Response to the http/ws Request for more information.
RefFieldXPath	The XPath expression in the response that identifies the reference field (Topic ID).
RefFieldPattern	A regular expression to be applied to be reference field.
CancelIdentifierXPath	The XPath expression whose presence specifies that an ongoing task should be canceled.
method	The HTTP method to be applied for sending the message. The four supported methods are: POST, GET, DELETE, PUT. For SOAP-based web services, always use POST.
AppendUsernamePassword	This setting is used for RESTful web services only. When it is set to true, the username and password are appended to the request header for HTTP authentication. This setting should not be used when Single Sign-On is enabled. In the case of SSO, configure the authentication scheme and scope parameters instead.

Table 5-22 HTTP/WS Adapter Outbound Properties (continued)

Property	Description
UsernameAlias	The argument name to be used in the request header for username.
PasswordAlias	The argument name to be used in the request header for password.
PublicKey	Enter the Public Key of the external system. **Encrypt** attributes sent out through Service Link are encrypted using the public key of the external system that is configured in agents. You can either use the existing public key in the drop-down list or create a new one while configuring agents in **Service Link > Manage Integrations > Agents for HTTP adapter** in the HttpOutboundAdapter.PublicKey. While adding a new public key, ensure that you enter the modulus and exponent of the new public key. If there are multiple PO instances each having a different public key to which the outbound messages have to be distributed, then separate agents have to be configured with different secure keys and each such agent is mapped to different external tasks. For more information about secure data transaction, see Securing Sensitive Data.

EncryptStringFormat	If you want to integrate Prime Service Catalog with any external system other than Process Orchestrator, the encrypted string must be passed through JSON or XML format.
	Prime Service Catalog integrates with Process Orchestrator automatically if you use Intelligent Automation format. For more information about Intelligent Automation format, see Cisco Intelligent Automation for Cloud Documentation.
	JSON format:
	`{"encryptStringFormat":{"initVector":"DCI4Tg==","saltHash":"LUNZWg==","payload":"UGF2YW4gQ29uZmlkZW50aWFsIQ=="}}`
	XML format:
	`<encryptStringFormat><initVector>DCI4Tg==</initVector><payload>UGF2YW4gQ29uZmlkZW50aWFsIQ==</payload><saltHash>LUNZWg==</saltHash></encryptStringFormat>`

Disable Unused HTTP Methods

In Prime Service Catalog only certain HTTP methods are allowed for REST APIs. Using any other method may cause unexpected results and you may want to restrict user access to the unsupported methods.

Supported HTTP methods are:

- GET
- HEAD
- POST
- PUT
- DELETE

- TRACE
- OPTIONS

The Connect and Patch methods are not allowed.

On apache server use the <LimitExcept> directive to limit access controls to all HTTP methods except the named methods. For more information see Apache documentation at *https://httpd.apache.org*. Below is an example of the LimitExcept directive:

```
<LimitExcept GET HEAD POST PUT DELETE TRACE OPTIONS>
  Require valid-user
</LimitExcept>
```

On the IIS server, the <verbs> element specifies which HTTP verbs are allowed or denied to limit the type of HTTP requests that are allowed by the Web server. For information see IIS server documentation at *https://docs.microsoft.com/en-us/iis.*

The following example configures two options: it configures IIS to deny HTTP PUT requests, and also configures request filtering to allow WebDAV access to all HTTP verbs.

```
<configuration>
  <system.webServer>
       <security>
          <requestFiltering>
             <verbs applyToWebDAV="false">
                <add verb="PUT" allowed="false" />
             </verbs>
          </requestFiltering>
       </security>
    </system.webServer>
</configuration>
```

Authentication Schemes

Some properties of the outbound http/ws adapter are required only for certain authentication schemes and, then, perhaps only for web servers with customized authentication. Authentication Schemes table summarizes authentication schemes supported by the outbound http/ws adapter.

Table 5-23 Authentication Schemes

Authentication Type	Description
Anonymous	The request is not required to supply user credentials; access to the web server is typically via a service account.
Basic	User name and password are required; password is sent in clear text.
Digest	User name and password are required, but password is transmitted as an MD5 hash.
NTLM	Integrated Windows Authentication on Windows 2003.
NTLMv2	Integrated Windows Authentication on Windows 2008 or latter.

Response to the http/ws Request

When a request is posted to a web site or a message sent to a web service, the target site typically sends a response to the message originator. If that response is unlikely to contain information useful to Service Link, you may set the *Process Response* property to false, to instruct Service Link to ignore any such messages. However, such responses might include additional information, such as the external system's ticket number or case number assigned to the task that originated in Service Catalog. In this case, you can set both the Process Response and Save Ref Field properties to true and specify the xpath for the Reference field for Service Link to capture the reference from the web service response. In addition, a transformation can be applied to the response to invoke actions to update the service form with information from the external system.

Using Agent Parameter in Outbound Property Values

Outbound properties may now contain agent parameter namespace.This allows a single agent to be used for multiple operations if they are routed to the same external system and differ only in the routing URL segments or request header values. The syntax for agent parameter namespace is $ParamName$

Reference Field and TopicID

External systems generally have their own means for identifying incidents, requests, or other objects, whether opened by a third-party system or maintained via the product's user interface. A designated Reference Field (TopicID) allows Service Catalog to maintain a cross-reference between the external system's unique identifier and the Service Catalog channel-id. Once the TopicID is identified in the initial response to the web service request and saved, further messages from the external system, received via the web services listener adapter, can use the TopicID to identify the Service Catalog external task.

Inbound Properties

- The HTTP/WS inbound adapter is a listener adapter and does not support polling based invocations.
- Only one HTTP/WS inbound agent should be configured for a given URL. Either the http or https protocol may be used.

No properties may be specified for an inbound http adapter. All http posts should be directed to the Integration Server's URL:

<ServerName>:<Port>/IntegrationServer/ishttplistener?channel-id=<channel-id>

where

- <ServerName> is the Service Catalog application server.
- <Port> is the port on which Service Catalog is listening.
- <channel-id> is the channel ID which uniquely identifies the task to be affected by the inbound message. Error 503 (Application Error) is returned to the third-party system if the channel-id does not apply to an ongoing task.

Web Service Invocation

A web service is not-so-simply "XML over HTTP". For an outbound adapter, an XML message is sent via http (or https) to a web service. The message, created by application of a transformation to the outbound message, must be enclosed within a SOAP envelope. A sample XML message to a web service might look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<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"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Header
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <AuthenticationInfo>
    <userName>ns28sbd</userName>
    <password>09rbc19</password>
  </AuthenticationInfo>
</soap:Header>
<soap:Body
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <Op>
    <Assigned_To_Group>CSCC</Assigned_To_Group>
    <Case_Type>Problem</Case_Type>
    <Category>Computer/Printer/Server</Category>

<!-- additional tags as required />

    <txt_internalticketid/>
    <txt_requestid >40</txt_requestid >
    <Type>New Hardware Request</Type>
  </Op>
</soap:Body>
</soap:Envelope>
```

JMS Adapter

The JMS inbound adapter is a listener adapter and does not support polling based invocations.

The JMS adapter can read and write messages from a queue or publish/subscribe to a particular topic. Only one JMS inbound agent should be configured for a given queue. It is not possible to use the same agent to subscribe to multiple topics. The topic must be fully specified; for example, "topic.sample.exported".

Inbound Adapter Properties

The Property sheet for the JMS inbound adapter prefixes the property names given below with "JMSInboundAdapter".

Table 5-24 JMS Adapter Inbound Properties

Name	Description
JndiProviderUrl	JNDI provider URL for looking up JMS administered objects for the inbound agent; default is jnp://localhost:4099.
JndiFactory	JNDI Naming factory for inbound agent; default is org.jnp.interfaces.NamingContextFactory.
JmsTopicFactory	Topic Connection factory for getting JMS Topic Connection for inbound agent; not used.
JmsQueueFactory	Queue Connection factory for getting JMS Queue Connection for inbound agent; default is ConnectionFactory.
MessageMode	Whether JMS destination is Queue or Topic. Valid values are Queue or Topic; default is Queue.
JmsQueue	Named JMS queue if message mode is Queue for inbound agent.
JmsTopic	Named JMS topic if message mode is Topic for inbound agent.
MessageType	Type of the message for the inbound agent. Valid value is Text
Publisher.isAdapter	If the publisher is adapter; default is True.
Listener.UseCallback	Whether to use callbacks; default is True.
UserName	User name for JNDI Security credentials for the inbound agent.
Password	Password for JNDI Security credentials for the inbound agent.

Outbound Adapter Properties

The Property sheet for the JMS outbound adapter prefixes the property names given below with "JMSOutboundAdapter".

Table 5-25 JMS Adapter Outbound Properties

Name	Description
JndiProviderUrl	JNDI provider URL for looking up JMS administered objects for the outbound agent; default is jnp://localhost:4099.
JndiFactory	JNDI Naming factory for the outbound agent; default is org.jnp.interfaces.NamingContextFactory.

Table 5-25 JMS Adapter Outbound Properties (continued)

Name	Description
JmsTopicFactory	Topic Connection factory for getting JMS Topic Connection for outbound agent; not used.
JmsQueueFactory	Queue Connection factory for getting JMS Queue Connection for outbound agent; default is ConnectionFactory.
MessageMode	Whether JMS destination is Queue or Topic. Valid values are Queue | Topic.
JmsQueue	Named JMS queue if message mode is Queue for outbound agent.
JmsTopic	Named JMS topic if message mode is Topic for outbound agent.
MessageType	Type of the Message for the outbound agent. Valid value is Text.
Publisher.isAdapter	If the publisher is adapter; default is True.
UserName	User name for JNDI Security credentials for outbound agent.
Password	Password for JNDI Security credentials for outbound agent.

MQ Adapter

The MQ inbound adapter is a poller adapter which uses the IBM WebSphere Message Queue (MQ) system. The adapter supports IBM MQ Series versions 5.x and above. It uses IBM MQ Series Java API for the integration. IBM MQ software is not included with Service Catalog, and a license must be obtained from IBM.

Inbound Properties.

The Property sheet for the MQ inbound adapter prefixes the property names given below with "MQInboundAdapter".

Table 5-26 IBM MQ Inbound Adapter Properties

Name	Description
ManagerName	Name of the IBM MQ Manager
HostName	Host name of the IBM MQ Server
Port	Port for the IBM MQ Server for Inbound
UserName	User Name for authentication
Password	Password for authentication
ChannelName	IBM MQ Channel Name for inbound messages
QueueName	Queue Name for inbound messages
MsgFormat	Message Format for inbound messages; default is Text

Outbound Properties

The Property sheet for the MQ outbound adapter prefixes the property names given below with "MQOutboundAdapter".

Table 5-27 IBM MQ Adapter Properties

Name	Description
ManagerName	Name of the IBM MQ Manager for outbound messages
HostName	Host name of the IBM MQ Server
Port	Port for the IBM MQ Server
UserName	User Name for authentication
Password	Password for authentication
ChannelName	IBM MQ Channel Name for outbound messages
QueueName	Queue Name for outbound messages
MsgFormat	Message Format for Outbound; default value is Text

Service Item Listener Adapter

Similar to the Web Service Listener Adapter (see the Web Service Listener Adapter), the Service Item Listener Adapter provides a Web service (SOAP) end point to be used by external systems to send updates to external tasks. In addition to task updates, the adapter allows the creation, update, and deletion of service items in Lifecycle Center as part of the inbound SOAP message. The adapter also allows the retrieval of service item metadata and the data for service item instances.

The SOAP message sent by an external system must invoke the “processMessage” operation. The message content within the soap body is transformed into a message that Service Link understands, then segregated based on the operation type, and forwarded to the Business Engine and Service Item Import processor, respectively. Up to two messages may result in the View Transactions page for an inbound SOAP message—one for task update operations (take-action, add-comments, send-parameters) and one for service item operations (, update, delete). The latter has “Service Item” as the message type.

Authentication for inbound messages can be enabled optionally by turning on the site setting "Inbound HTTP Requests Authentication" in the Administration module. For more information, see Web Service Listener Adapter.

Inbound Properties

The Property sheet for the Service Item Listener inbound adapter prefixes the property names given below with “ServiceItemListenerInboundAdapter”.

Table 5-28 Service Item Listener Inbound Adapter Properties

Property	Description
Wsdl	The URL of the wsdl that describes the Service Catalog inbound Service Item for the current installation has the format: <Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl where: <Protocol> is either http or https. <ServerName> is the server where Service Link is installed. <Port> is the communication port specified for Service Link. For example, http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl This property is read-only. It is made available so that designers can consult the WSDL, which is useful in understanding the services and writing a webservice client.
RoutingURL	The URL to which the SOAP message should be sent has the format: <Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService This property is read-only. It is made available so that external systems integrators can write clients that post SOAP messages to this URL.

Outbound Properties

The Service Item Listener Adapter is unidirectional—inbound only. Therefore, there are no Outbound Properties.

Web Service Listener Adapter

The Web Service Listener Adapter provides a Web service (SOAP) end point to be used by external systems to send updates to external tasks. The SOAP message sent by an external system must invoke the “processMessage” operation. The message content within the soap body is transformed into a message that Service Link understands, then forwarded to the HTTP/WS inbound adapter to be processed further.

The Web service Listener Adapter uses an underlying Web Service Listener. Authentication for inbound messages can be enabled optionally by turning on the site setting "Inbound HTTP Request Authentication" in the Administration module. Once enabled, a valid username and the corresponding password are required to be passed in the request header for the inbound message to be processed. If desired, the "Accept Encrypted Password" setting can be enabled to enforce the use of encrypted password only. 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

Inbound Properties.

The Property sheet for the Web Service Listener inbound adapter prefixes the property names given below with “WSListenerInboundAdapter”.

Table 5-29 Web Service Listener Inbound Adapter Properties

Property	Description
WsdlURL	The URL of the wsdl that describes the Service Catalog inbound Web Service for the current installation has the format: <Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/TaskService.wsdl where: <Protocol> is either http or https. <ServerName> is the server where Service Link is installed. <Port> is the communication port specified for Service Link. For example, http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/TaskService.wsdl This property is read-only. It is made available so that designers can consult the WSDL, which is useful in understanding the services and writing a webservice client.
WsdlRoutingURL	The URL to which the SOAP message should be sent has the format: <Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService This property is read-only. It is made available so that external systems integrators can write clients that post SOAP messages to this URL

Outbound Properties

The Web Service Listener Adapter is unidirectional—inbound only. Therefore, there are no Outbound Properties.

Securing Sensitive Data

Prime Service Catalog often contains data that must be secured when it is viewed/accessed within the product as well as when it is exchanged with external systems. Therefore, you can use the encryption methods during secure data transaction.

For information about encrypting dictionary attributes within Prime Service Catalog, see “Configuring Dictionaries” in Cisco Prime Service Catalog Designer Guide.

Encrypt attributes sent out through Service Link agents based on HTTP/WS and AMQP task adapters are secured or encrypted using the 1024/2048-bit RSA Public Key of the external system that is configured in the agents. You must configure the **Public Key** and **EncryptStringFormat** of the external system in Http/WS outbound properties page of the adapter and then configure the agents properties. For more information, see Managing Adapters.

For example, the encrypt attribute of the outbound message to Process Orchestrator (PO) is converted in the format below:

- 4 bytes of magic number
- 1 byte of Init Vector (IV) length
- IV
- 2 bytes for Salt length
- Salt- Secured using PO's public key
- 4 bytes for encrypted payload length
- Encrypted payload

When there is a public key update in the external systems, then you must update the agent accordingly. For example, when there is a public key update in PO, then the PO will accept the old key pair for 3 days as grace period so the old messages in Service Link could still pass through. After three days of grace period PO will return an error if the agent in Prime Service Catalog has not been updated. Therefore the public key needs to be manually re-configured in agents once again when re-key happens in the external systems.

Note If the Service Link agent does not have the public key of the external system configured, then dictionary field attributes that are defined to be stored and encrypted are decrypted and clear-text values for these attributes are sent over the outbound message only when the agent uses HTTPs/SSL protocol. If SSL is not configured, then the dictionary field attributes are sent as encrypted format only.

Note Prime Service Catalog supports AES algorithm and 128/256 bit symmetric keys. Therefore, ensure that you install "Java Unlimited Strength Crypto Policy" file manually on Java Development Kit (JDK) during Service Catalog installation. Java Unlimited Strength Crypto Policy file enables the external system to perform 2048 bit encryption. If you do not install Unlimited Strength Crypto Policy the installer displays an error message during server startup. For instructions about installing the unlimited strength crypto policy file, see Oracle Website. For more information about installing Cisco Prime Service Catalog, see Cisco Prime Service Catalog Installation and Upgrade Guide.

Cloud Resource Manager Adapter

The Cloud Resource Manager adapter supports outbound communications from Service Link to UCSD. The Cloud Resource Manager adapter supports polling interval attribute which determines the frequency at which the inbound poller is activated.

Note Cloud Resource Manager adapter is provided with predefined configuration and it is recommended that you deploy this adapter using the default configuration.

Using the Integration Wizard in Service Designer

The Integration Wizard automates many of the steps involved in creating an integration between Service Catalog and SOAP web services. The Integration Wizard works by retrieving the wsdl and operation to be invoked by the web service integration. Based on that definition of the integration, the integration wizard displays all components required to support the integration.

- The Service Link agent that can be used in an external task to perform the integration is created and referenced in the delivery plan of the current service.
- A transformation to transform nsXML into the SOAP message required by the web service is created and referenced in the Outbound Adapter of the agent.
- Agent parameters for all data required both in the initial web service request and the response are added to the agent definition.
- A dictionary containing fields to hold agent parameter values is created, and dictionary fields are mapped to corresponding agent parameters.
- An active form component containing the dictionary is created and included in the current service.

The Integration Wizard uses some default options in defining the authentication method and behavior of the integration. If these settings are not appropriate, or if the integration must be modified after it has been created, the advanced configuration options available in Service Link Manage Integration pages can be used to edit the agent definition.

The Integration Wizard is available only to those service designers who have been granted a role that allows creation of Service Link agents and transformations.

Wsdl's to be accessed by the Integration Wizard must comply with Web Services Operability (WS-I) best practices.

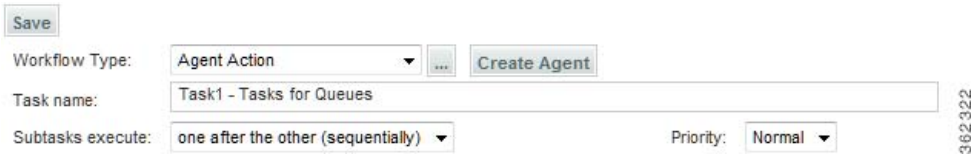
To use the Integration Wizard:

Step 1 Edit the service in Service Designer.

Step 2 Go to the **General** subtab of the Plan tab for the service. Optionally fill in other data relating to the task.

Step 3 Click **Agent**.

Figure 5-27 Agent

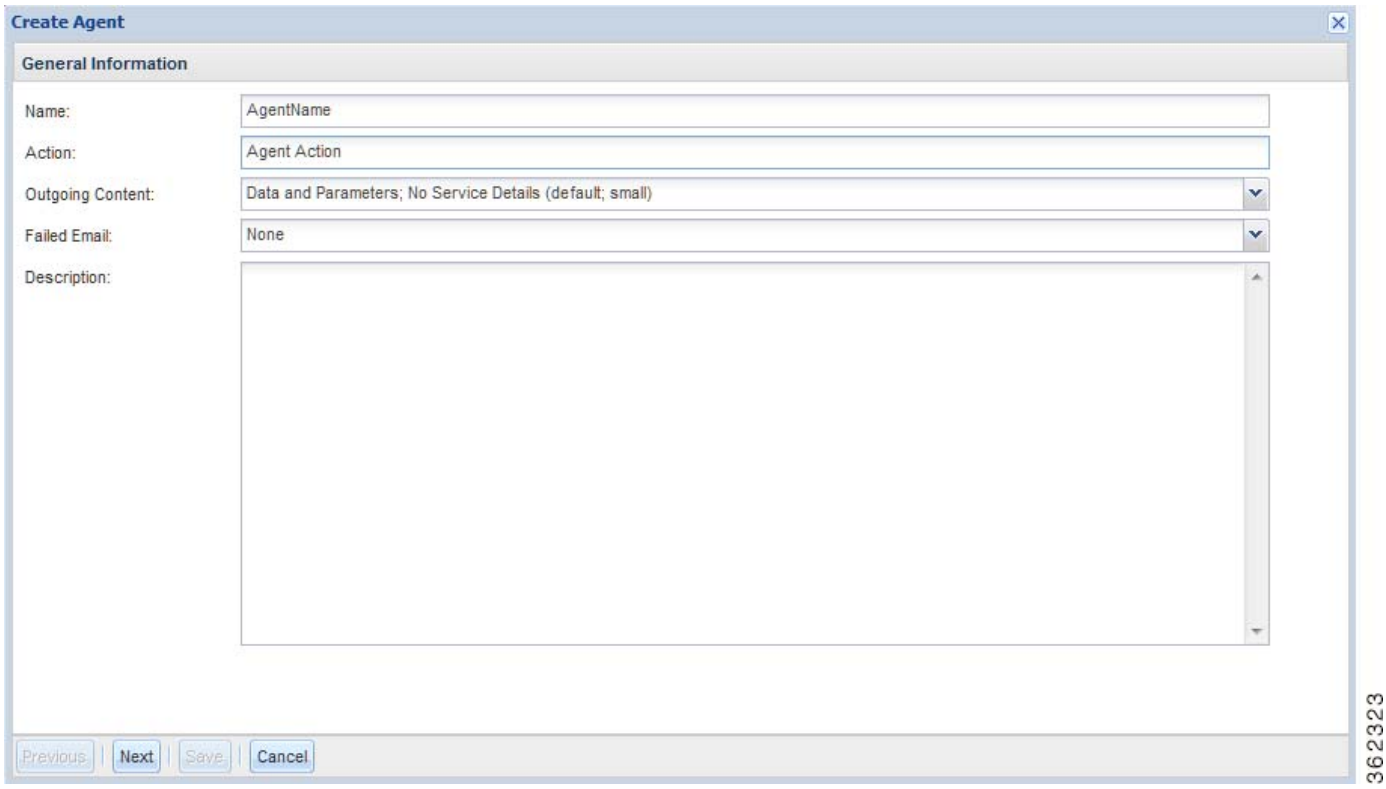


The first page of the Integration Wizard appears. The wizard may consist of up to eight pages, depending on how the agent is configured. As each page is completed, click **Next** to advance to the next page, or **Previous** to return to a previous page. When you are finished, click **Save** to save the definition of the agent (and other design components) or **Cancel** to exit without saving your work.

General Information

Start by specifying general information about the agent:

Figure 5-28 Create Agent



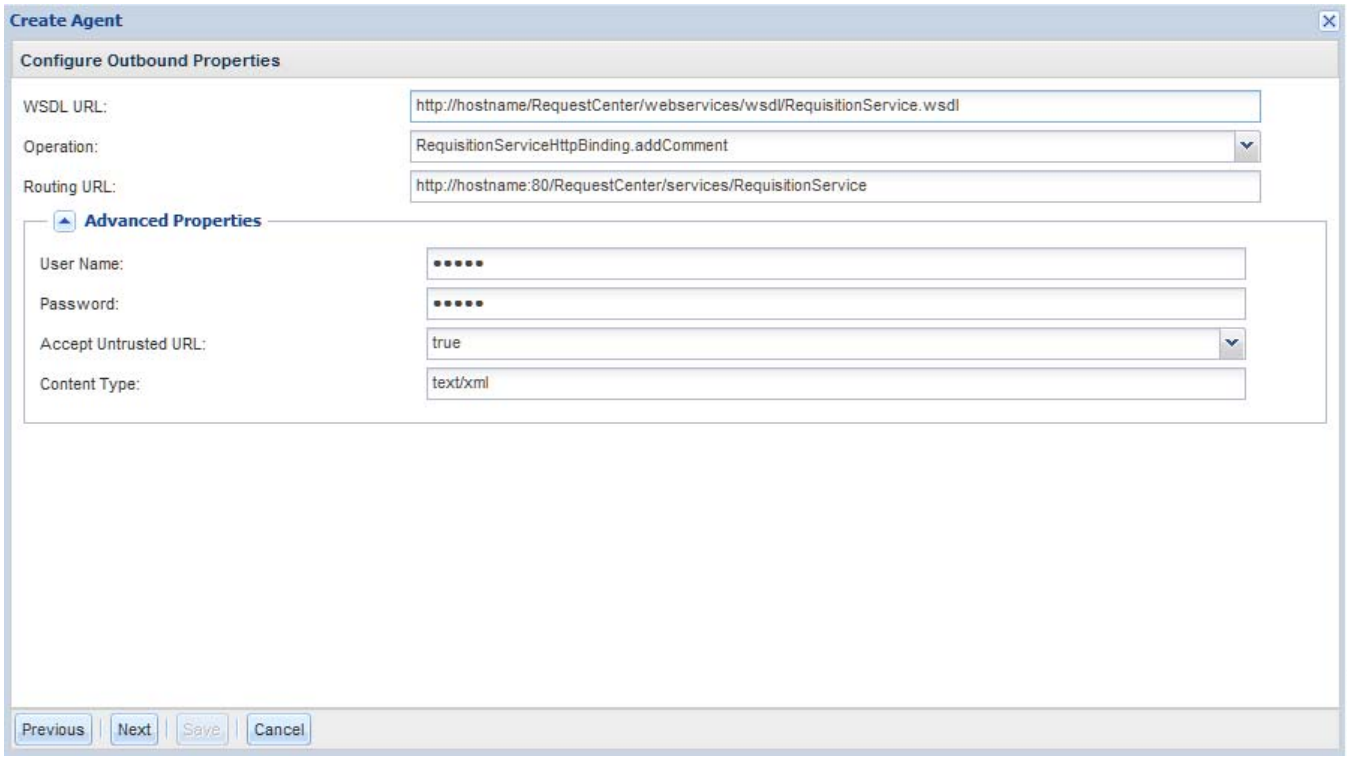
The dictionary and active form component to be created will have the same name as the agent. Therefore, since naming standards for dictionaries are more stringent than for agents, the agent name can contain only letters, numbers and the underscore, and cannot start with a number.

All other settings on this screen match those available in the Agents page of Service Link.

Click **Next** to proceed to the next page of the wizard.

Outbound Properties

Figure 5-29 Create Agent Outbound Properties



Enter the location of the WSDL containing the operation to be performed by the integration. This will typically be the URL where the WSDL resides.

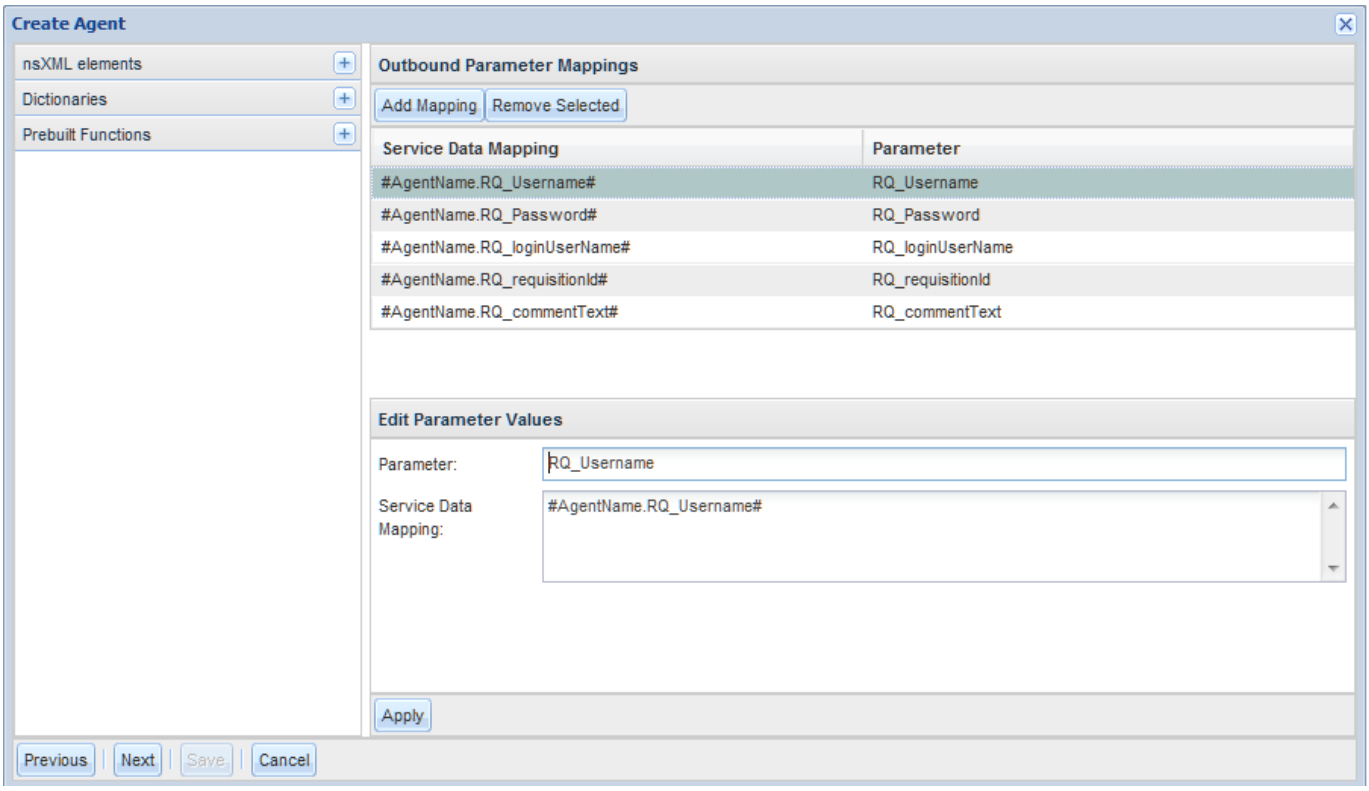
The Integration Wizard reads the wizard and displays a list of supported operations. Select the desired operation. The attributes specified for the operation will drive the definition of agent parameters on subsequent pages of the wizard.

If the wsdl includes a routing url, that, too, is displayed.

If desired, click the **Advanced Properties** drop-down button to display additional settings for the integration. These may be entered now or specified later via Service Link. Only basic authentication can be specified via the wizard.

Outbound Request Parameter Mappings

Figure 5-30 Outbound Request Parameter Mappings



The wizard parses the wsdl and, in the sample shown, determines that it includes two attributes that must be used in the web service outbound request. Therefore, it s two agent parameters whose names match the names of the attributes in the wsdl.

The agent parameters are mapped to dictionary fields. The field names match the names of attributes in the wsdl, and the dictionary name matches the agent name. This dictionary is automatically created when you save the agent.

If desired, you can change the Service Data Mapping to refer to a dictionary and field that have previously been defined in Service Designer. This effectively changes the agent parameter mapping. However, the dictionary created by the wizard will still contain the original field. You may remove this by editing the dictionary definition.

A short digression might be useful here about structuring and using dictionaries in services. The primary purpose of a Service Catalog dictionary is to structure the data to be shown to users on a service form. Therefore, service designers typically design dictionaries with the user interface in mind, grouping and arranging fields to optimize the experience of both customers and service team members.

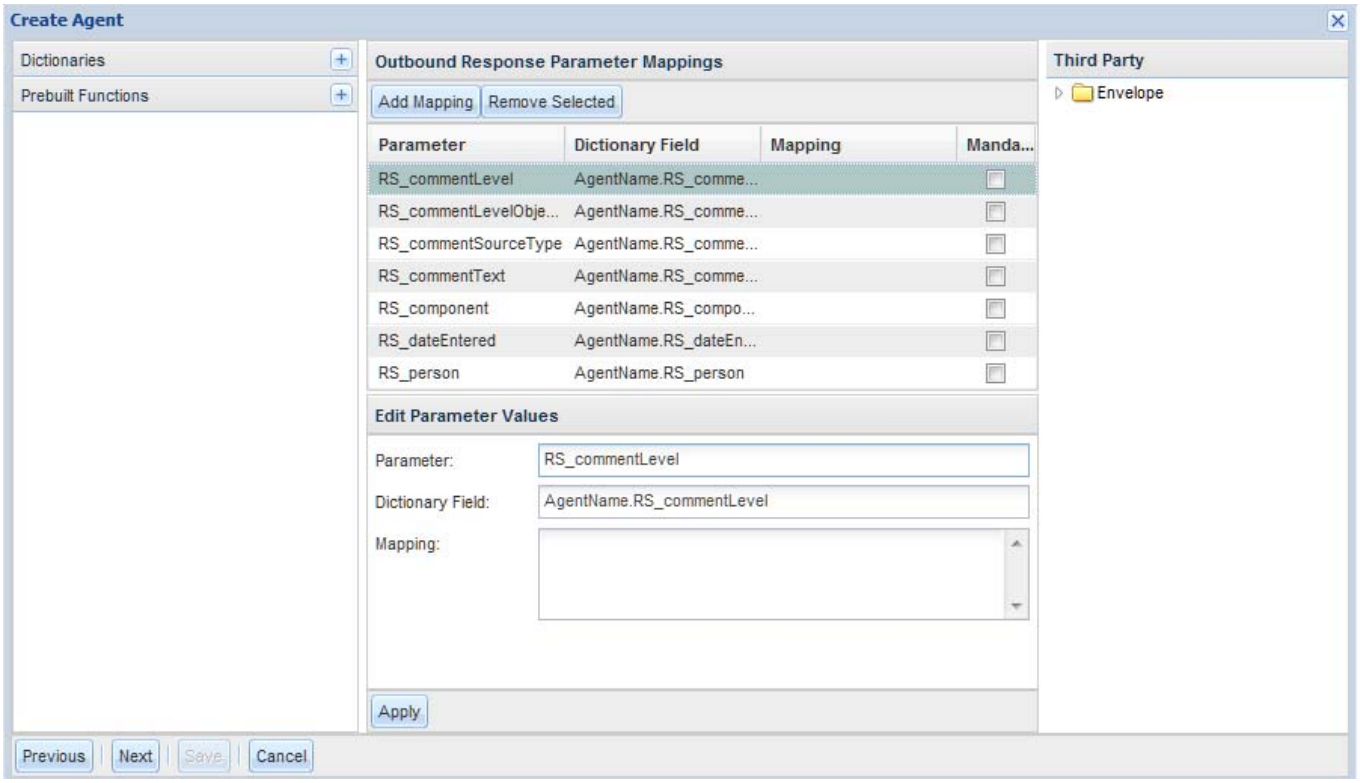
In principle, an outbound message might need to include data that has been entered (or defaulted or computed) in fields in many dictionaries. However, this would make maintaining the agent parameter mappings more complicated and prone to error—integration designers would have to be well acquainted with the design of the service form and its dictionaries. Therefore, it is recommended practice to a dictionary solely for the purpose of containing integration data. Some fields in the dictionary may be redundant with fields displayed on the service form. In this case, the service designer should supply

conditional rules to copy the value of the field from the displayed dictionary to the integration dictionary. Further, the integration dictionary is not displayed on the service form (it is typically hidden via an active form rule); however, it can be kept visible during development to facilitate debugging.

Outbound Response Parameter Mappings

By default, the Integration Wizard assumes that a response received from the target system will be processed. Any attributes sent in the response have corresponding agent attributes that are mapped to dictionary fields.

Figure 5-31 Outbound Response Parameter Mappings



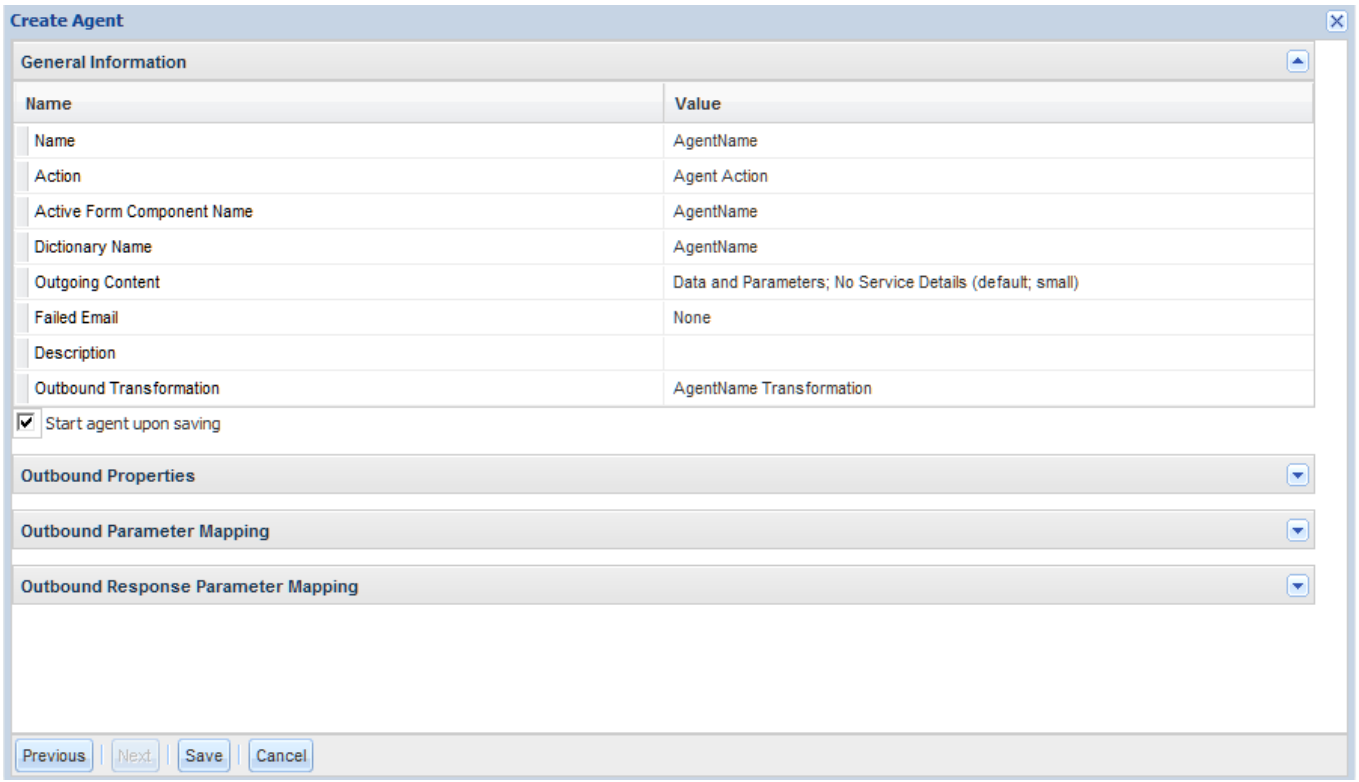
In addition to a agent-to-field correspondence, the mapping may include simple XSLT operations, available via the Prebuilt Functions drop-down arrow to the left of the page.

As for outbound parameters, the inbound parameter could also be mapped to an alternative dictionary field. All dictionaries can be browsed via the Dictionaries drop-down arrow to the left of the page.

Integration Summary

The last page of the wizard summarizes the integration as defined. You may return to any previous page to make corrections or click **Save** to save the agent and all other integration components created. By default, the agent is started when the integration is saved. You can alter this behavior by unchecking the "Start agent upon saving" check box.

Figure 5-32 Integration Summary



All components are now available for editing via Service Link and Service Designer screens. These components are shown in Integration Components table.

Table 5-30 Integration Components

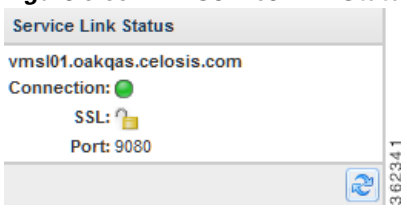
Integration Component	Description
Agent	An agent using an http/ws outbound adapter, with agent parameters and an associated transformation.
Transformation	The transformation required to convert outbound nsXML to the expected format for the WSDL operation selected.
Dictionary	Dictionary containing fields corresponding to all outbound and inbound parameters. The dictionary is created in the Integration group; it can be moved if desired.
Active Form Component	Active form component containing the dictionary created. The form component is created in the Integration group; it can be moved if desired.

Service Link Troubleshooting and Administration

Checking Service Link Status

The starting point for checking the operational status of Service Link is the Service Link Status display. The Service Link status is always displayed beneath the Common Tasks area of the Service Link Home page.

Figure 5-33 Service Link Status



This feature helps you verify that Service Link is communicating with the Service Catalog service via its assigned port.

The Service Link Status display indicates whether the Service Link connection status is operational, and shows the port and protocol being used.

Starting and Stopping Agents

You must stop a service link agent modify the agent properties and restart again for it to function. Agents can be started and stopped individually by using the Control Agents page.

Note If the Service Link service is stopped and restarted, all agents that were running when the service was stopped are automatically restarted.

Logging

All adapters log their activities into the server log file.

In addition, each standard adapter has its own log file on the Service Link\log directory. The degree of detail written to the log is configurable; instructions for doing so are application-server specific.

For details on managing both server and adapter-specific log files, see the *Cisco Prime Service Catalog Administration and Operations Guide*.

JBoss Logging

In a JBoss installation, Service Link adapter logs can be segregated from the server log into separate files by modifying the logging.properties file under the "*<JBOSS_DIR>*\standalone\configuration" directory. Examples of such configurations can be found in the sample property files "\preinstall\jboss\templates" directory in the product package.

WebLogic Logging

WebLogic does not allow separation of log files per adapter, and the IntegrationServer component is configured to use the WebLogic logger by default. If separation of logs is desired, edit the file newscalelog.properties under ISEE.war/WEB-INF/classes. Uncomment the line that specifies commons logging as the logging mechanism. It is also very important that you uncomment and set a valid value for logger.directory to a valid and existing directory in the system, where the user that is used to run IntegrationServer has full write access. The file newscalelog.properties has additional instructions. In addition, if additional settings for other adapters are desired, edit the file log4j.xml and use the FILE_ADAPTER appender and category as a base and adjust the appender name and reference, the package of the appender and the file name.

WebSphere Logging

WebSphere logging of Service Link is based by default on log4j as included in the WebSphere application server. The log4j implementation in WebSphere is powerful and configurable through the administration console and other tools. However, it does not allow for easy separation of log files. If you want to separate log files per adapter in WebSphere, follow the steps below:

Step 1 Under “ISEE.war/WEB-INF/classes/config”, locate the file newscalelog.properties and open it with an editor.

Step 2 Uncomment the line:

```
logger.class.name=com.newscale.bfw.logging.LogUtilCommonsImpl
```

Step 3 Locate the line for logger.directory. Specify the log directory; for example:

```
logger.directory=I:/logfiles/servicelinkserver2
```

It is very important that you enter a valid directory where these log files reside and the user that is used to run IntegrationServer has full write access to it.

Step 4 Under the “ISEE.war/META-INF/” directory, manually a folder named “services”.

Step 5 Under this services folder, manually a text file named “org.apache.commons.logging.LogFactory”. Within the file, add one line as follows:

```
org.apache.commons.logging.impl.Log4jFactory
```

Message Purging

As Service Link messages may occupy a significant amount of space in the database and they are no longer referenced once a request is completed, there are benefits in purging them from the database on a regular basis to reduce the database size. The purge utility can be accessed in the Administration module under the Utilities tab. The utility does not actually remove the message record entries. Instead it replaces the XML content of the message with "Message has been purged" for all completed or failed message older than the retention period specified. For more information, see the Purge Utilities section in the Cisco Prime Service Catalog Administration and Operations Guide.

Application Server Configuration Files

You can analyze the following files when troubleshooting.

- **rcjms.properties file** – This file contains the information about the integration outbound JMS queue and can be located in the "/RequestCenter.war/WEB-INF/classes/config" directory. The business engine puts the message in the queue specified in this file. The values for the following properties should match with those in the integrationserver.properties file in the ISEE.war file:

  ISEEOutbound.JndiProviderUrl

  ISEEOutbound.JndiFactory

  ISEEOutbound.JmsTopicFactory

  ISEEOutbound.JmsQueueFactory

  ISEEOutbound.JmsQueue

  ISEEOutbound.JmsTopic

- **integrationserver.properties** file – This file contains the information about inbound and outbound JMS queues and can be located in the "/ISEE.war/WEB- INF/classes/config" directory. Verify the JMS properties specified in this folder.
- **newscale.properties** file – This file contains the property for **isee.base.url**. Ensure that it points to the Service Link server url.

Online Error Log

In addition to the server log file and adapter-specific log files, any errors detected by Service Link can also be viewed online. The message text for a failed message shown on the Messages page is a hyperlink to the detailed error for that message.

The error messages are exactly those that appear in the server logs and may be highly technical. Some sample error messages, and an explanation, are given below.

```
com.newscale.is.core.RoutingException: Routing exception found: Reference Field not
retrieved from response
```

An outbound web services message was sent, but the inbound response could not be processed because the specified referenced field was not in the response message.

```
com.newscale.is.core.TransformationException: javax.xml.transform.TransformerException:
javax.xml.transform.TransformerException: Tag is not allowed in this position in the
stylesheet!
```

The transformation produced an invalid XML message.

Prebuilt Functions

Prebuilt functions provide the ability to manipulate the values of agent parameters included in a nsXML message.

Prebuilt functions were developed using the FreeMarker template engine, version 2.3.12, available as open source software and developed by the Visigoth Software Society. Cisco has certified only those functions documented below and available in the drop-down list when building agent parameters. Other functions supported by the FreeMarker framework may be used, but should be extensively tested.

Function Usage

Basic function usage consists of applying to the function to an expression, specifying an argument list for the function if required. In general terms:

```
${Expression?function(argumentList)}
```

For Service Link, the expression is typically either a dictionary field, specified via lightweight namespace syntax, or an nsXML element. It must be enclosed in quotes:

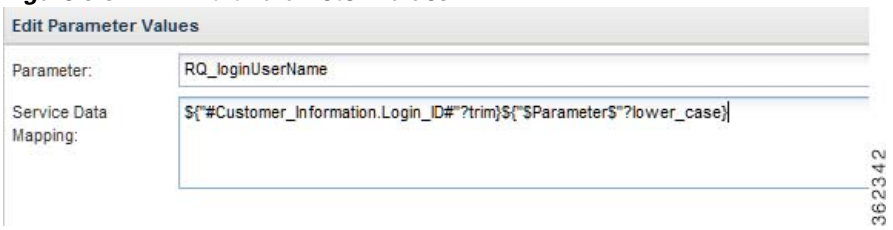
```
${"#Customer_Information.Login_ID#"?upper_case}
```

Two or more functions can be chained-applied to the same expression-by using the syntax:

```
${Expression?function1(argumentList)}${"$Parameter$"?function2}
```

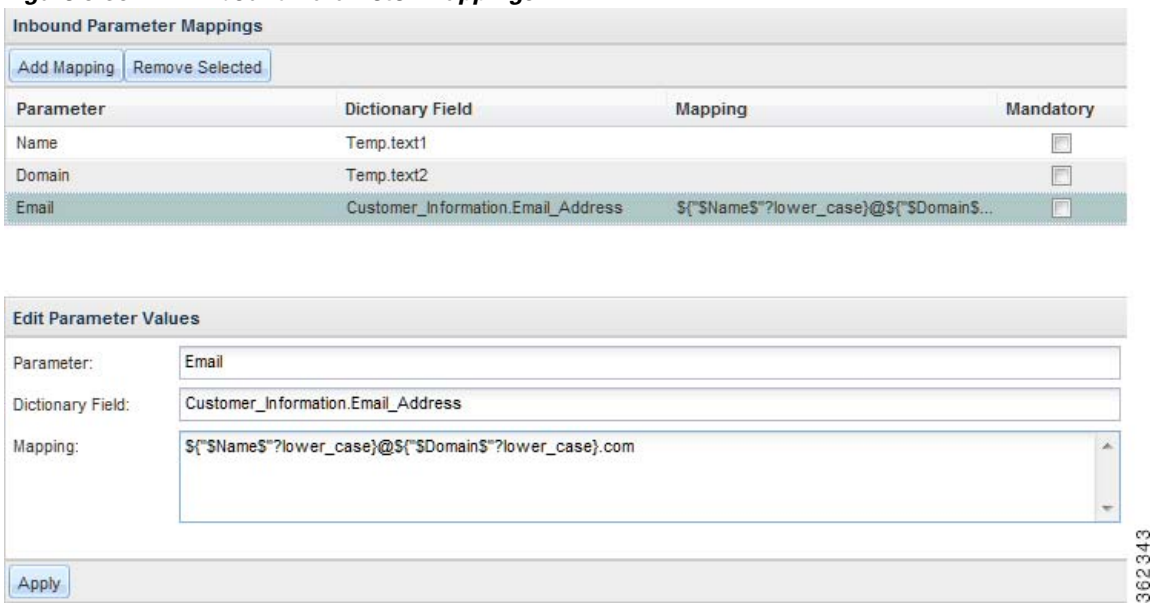
For example, the service data mapping below first trims any leading or trailing spaces from the designated dictionary field, then converts the result to lower case.

Figure 5-34 Edit Parameter Values



Multiple elements can be combined in one mapping, as shown below. The elements are implicitly concatenated together to form one string.

Figure 5-35 Inbound Parameter Mappings



This scenario also shows another coding technique—the use of "temporary" fields to hold input values so they can be used in a mapping expression.

Function Synopsis

substring

The substring function has the syntax:

```
exp?substring(from, toExclusive), also callable as exp?substring(from)
```

A substring of the string. *from* is the index of the first character. It must be a number that is at least 0 and less than or equal with *toExclusive*, or else an error will abort the template processing. The *toExclusive* is the index of the character position after the last character of the substring, or with other words, it is one greater than the index of the last character. It must be a number that is at least 0 and less than or equal to the length of the string, or else an error will abort the template processing. If the *toExclusive* is omitted, then it defaults to the length of the string. If a parameter is a number that is not an integer, only the integer part of the number is used.

index_of

Returns the index within this string of the first occurrence of the specified substring. For example, *"abcabc"?index_of("bc")* will return 1 (don't forget that the index of the first character is 0). Also, you can specify the index to start the search from: "*abcabc*"*?index_of(*"*bc*"*, 2)* will return 4. There is no restriction on the numerical value of the second parameter: if it is negative, it has the same effect as if it were zero, and if it is greater than the length of this string, it has the same effect as if it were equal to the length of this string. Decimal values are truncated to integers.

If the 1st parameter does not occur as a substring in this string (starting from the given index, if you use the second parameter), then it returns –1.

last_index_of

Returns the index within this string of the last (rightmost) occurrence of the specified substring. It returns the index of the first (leftmost) character of the substring. For example: "*abcabc*"*?last_index_of(*"*ab*"*)* will return 3. Also, you can specify the index to start the search from. For example,

```
"abcabc"?last_index_of("ab", 2)
```

will return 0. Note that the second parameter indicates the maximum index of the start of the substring. There is no restriction on the numerical value of the second parameter: if it is negative, it has the same effect as if it were zero, and if it is greater than the length of this string, it has the same effect as if it were equal to the length of this string. Decimal values are truncated to integers.

If the first parameter does not occur as a substring in this string (before the given index, if you use the second parameter), then it returns –1.

length

The number of characters in the string.

lower_case

The lower case version of the string. For example, "*GrEeN MoUsE*" becomes "*green mouse*".

replace

It is used to replace all occurrences of a string in the original string with another string. It does not deal with word boundaries. For example:

```
${"this is a car acarus"?replace("car", "bulldozer")}
```

will print:

```
 this is a bulldozer abulldozerus
```

The replacing occurs in left-to-right order. This means that this:

```
${"aaaaa"?replace("aaa", "X")}
```

will print:

```
    Xaa
```

If the first parameter is an empty string, then all occurrences of the empty string are replaced, like `"foo"?replace("","|")` will evaluate to `"|f|o|o|"`.

replace accepts an optional **flags parameter**, as its third parameter.

upper_case

The upper case version of the string. For example, "*GrEeN MoUsE*" becomes "*GREEN MOUSE*".

Designing Integration with Adapter Development Kit

You can use the Service Link Adapter Development Kit (ADK) to develop Service Link adapters. The ADK is the set of components that allow the production of adapters for the Service Link subsystem of Service Catalog. Service Link provides external communications for Service Catalog and provides for coordinated externalization of workflow tasks with other systems.

To achieve this communication, Service Link supports installable adapters. Service Link ships with standard adapters, but developers can create other adapters. This chapter describes the process of writing adapters.

This chapter is intended for:

- **Administrator**. The administrator has access to the product packages and can install Service Catalog products in a customer system.
- **Adapter Developer**. The adapter developer is a person that is well versed in Java technologies, including ANT, and it is a subject matter expert of the integration required.

For more support on building custom adapter, contact *Cisco TAC*.

Getting Started

This section describes the installation of the ADK, its structure, compiling adapters, and adapter deployment.

Installing the JDK

Follow the instructions from Sun or IBM to install the Java Development Kit. Service Catalog is certified with Sun JDK 6 for installation on WebLogic 12.2.1.2 or JBoss 7.1.1, and with IBM Java 1.6 for installation on WebSphere 7.0.0.17.

Installing the ADK

To install the ADK:

1. **Administrator**: Expand the context of the product packages, locate the adk.zip under the image/isee/dist folder, and inform the adapter developer of the location.

2. **Adapter developer**: Obtain the file adk.zip (or adk.tar.gz) from the administrator.
3. **Adapter developer**: Expand the ADK package in a local machine, in C:\ADK. It does not have to be the C drive, nor the ADK directory. However, the examples in this chapter use C:\ADK.

ADK Structure

After installing the ADK, the following subdirectories exist:

Table 6-1 Subdirectories for ADK

Directory	Description
<root>	Contains the build procedure files.
ant	Complete ANT build system. This ANT is the standard Apache ANT build system, with some added extensions.
doc	Contains the java doc subdirectory. You may place the ADK documentation here.
doc\javadoc	The help for the ADK in javadoc format.
example	Contains our example adapter.
example\src	Contains the source java files for the example adapters.
example\deploy	Contains adapter.xml, which describes this adapter.
lib	Contains the ADK libraries needed for compilation.

An adapter is a subdirectory in the main ADK structure. After installing, **example** is one such adapter. Adapter code has to be structured in the following way

:

Table 6-2 Adapter Code Structuring Table

Directory	Description
<adapter>	The short name of the adapter. In the case of the example adapter, this is **example**.
<adapter>\src	Mandatory. The root for the source java files.
<adapter>\deploy	Mandatory. The deployment directory. This should contain a file called **adapter.xml**.
<adapter>\lib	Optional. Additional libraries to be added to the lib directory of **ISEE.war**.
<adapter>\config	Optional. Additional files to be copied to the classes directory of **ISEE.war**.

In the example provided, only **src** and **deploy** exist.

After compiling the files, create a staging directory (see the following sections for a description on how to build adapters). The staging directory can be deleted and recreated with the build procedure later.

Table 6-3 Directory Description Table

Directory	Description
staging	The root of the production directory.
staging\classes	The compiled java classes for the adapters.

Table 6-3 Directory Description Table

staging\adapters	Contains the built jars for each of the adapters. The adapters appear with the name adapter_<adaptershortname>.jar.
staging\config	The files from each config subdirectories for each adapter.
staging\deploy	The files from each of the deploy subdirectories, renamed as <adaptershortname>.xml.
staging\lib	The files from each of the lib subdirectories for each adapter.
staging\dist	The final isee.adapters deployable file.

Creating Adapter Source Structures

To create new adapters:

Step 1 Create the directory structure as defined above.

Step 2 Create the source and place it in the structure.

Step 3 Create adapter.xml and place it in the deploy directory. For more information, see the Understanding the adapter.xml Descriptor.

Step 4 Optionally add additional libraries and configuration files.

Step 5 Modify build.properties and add your adapter to the adapters line. This configures ANT to look for the created directories.

The compilation steps allow for adding the build to version control, and later compiled before installation.

Compiling Adapters

After creating the adapter, build it by executing:

build.cmd (or **./build.sh** for unix systems)

The final product appears under **staging/dist/isee.adapters**. This file needs to be provided to the administrator for deployment.

Deploying Adapters

To deploy an adapter, the administrator performs the following procedures:

Step 1 Obtain the Service Link custom adapter package. It should be in the form of a zip file.

Step 2 Unzip the adapter to a temporary directory (for example, c:\temp\adapter). This directory is hereinafter referred to as <AH>.

Step 3 Copy the <AH>/adapters/<ADAPTER_NAME>.jar to the deployed “ISEE.war/WEB-INF/lib” directory.

Step 4 Copy the <AH>/lib/* files (if any) to the deployed “ISEE.war/WEB-INF/lib” directory.

Step 5 Copy the <AH>/config/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.

Step 6 Copy the <AH>/udk/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.

Step 7 If the custom adapter is not developed internally using the Adapter Development Kit, obtain the adk.zip from the "<ServicePortal_Software_Dir>/adk" folder, where <ServicePortal_Software_Dir> is the extracted software image of the Service Catalog application. Extract the adk.zip to c:\adk (for Windows) or /opt/adk (for UNIX/Linux). This directory is hereinafter referred to as <ADK>.

Step 8 Set the JAVA_HOME environment variable if it is not already configured in the environment.

Step 9 Open a command window and cd into the <ADK>/lib folder. Execute adapter_dbinstaller.sh or adapter_dbinstaller.cmd as appropriate to your environment. Use --help or -? as the argument to the adapter installer to see the list of required input arguments. When prompted for the Adapter Deployment Descriptor file, enter the xml file name under the <AH>/deploy directory with the full path (for example, /opt/<AH>/deploy/custom_adapter.xml).

Step 10 For each udk file that was installed (Step 6), add the file’s name to the “UDConfig” property inside the integrationserver.properties file. The UDConfig property is a comma-delimited list of all udconfig files. Append the adapters udconfig files to this list.

Step 11 Start the Service Catalog and Service Link servers. Verify the new adapter exists.

Implementing an Adapter?

An adapter is the vehicle by which Service Link connects with external systems (often referred as third-party systems). Adapters are composed of three pieces:

- An inbound piece, referred to as the **inbound adapter**
- An outbound piece, referred to as the **outbound adapter**
- An error handler

The inbound adapter manages incoming communications into Service Catalog. It processes the XML messages coming into the system. There are two types of inbound adapters: pollers and listeners. A poller is a thread that periodically wakes up and looks for incoming messages, while a listener waits and is awakened by an external event. An example of a poller is the inbound file adapter, which needs to periodically check for messages. An example of a listener is the HTTP adapter which waits until an HTTP XML event is posted.

Outbound adapters manage the XML messages coming out of Service Catalog. There is only one type of outbound adapter.

An agent is a logical element that protects service designers from having to know all the complexities of adapter and connection properties. And agent defines an inbound adapter and an outbound adapter. The inbound adapter is optional and can be specified as “Auto complete”. “Auto complete” is a mode whereby the system does not need a reply from a third party for the workflow to proceed, and is mostly associated with unreliable, or shoot-and-forget protocols, such as an email-based system. The administrator configures agents and their associations with adapters for the service designers to use.

In addition, XML transformations can be applied to messages before they go to a third-party system, or after they are received from a third-party system and delivered to Service Link.

The message system uses a common XML dialect known as nsXML, which is a schema that defines the valid XML that Service Link can process or produce. nsXML currently consists of six operations:

- task-started – outgoing
- task-cancelled – outgoing
- take-action – incoming
- send-parameters – incoming
- add-comment – incoming

When outgoing, Service Link can transform these operations to a destination. The same is true for incoming messages, and the XSL transformations can convert the external format into the nsXML dialect.

For more information about nsXML operations, see Understanding Communication Message Content and Structure.

Types of Adapters

The adapters are of two types:

- Transport Adapters

  Transport adapters are specific to a given transport, such as HTTP, file, JMS, or some proprietary network socket implementation.

- Application Adapters

  Application adapters have an element of transport but are better understood by the specific third-party application, such as Remedy and Siebel. In many cases native APIs are provided through jars. In this version of Service Link, transport adapters cannot yet be extended to create application adapters.

Agents may use different adapters for inbound and outbound messages.

Adapter Components

In addition to java code, an adapter is composed of:

- Jar libraries (for example, Remedy java API)
- Static configuration files. We do not recommend changing of text files once deployed.
- Deployment descriptor. An XML file that describes the adapter.

Connection Properties

In order to connect to third-party systems, adapters may expose connection properties that the Service Link module exposes to administrators. They are described in the XML deployment descriptor, and their values can be retrieved by the java code to a well established API.

Example: Implementing a File Adapter

This section illustrates how to implement a simple adapter. The example adapter is a file adapter that communicates with the external world.

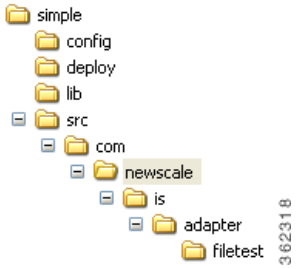
The simple file adapter contains:

- An outbound adapter that creates a file, whose file name is specified through adapter properties.
- An Inbound adapter that reads a file, whose file name is specified through adapter properties.
- A simple exception handler.

Creating Directory Structure

First, create the adapter's directory structure. In the ADK directory structure, create a directory named **simple**, and create the following directory structure under it:

Figure 6-1 Directory Structure



Under **src** notice the source package representing the java package **com.newscale.is.adapter.filetest**. Any other package can be used, but this example uses this one.

Creating Outbound Adapter Class

Secondly, create the outbound adapter class. The name is **FileOutboundAdapter** and this class file should be placed in the package described in step one. Its skeleton is shown below, without the implementation of the methods.

```
import com.newscale.is.adk.AdapterContext;
import com.newscale.is.adk.base.OutboundAdapter;
import com.newscale.is.adk.exceptions.AdapterException;
public class  FileOutboundAdapter extends OutboundAdapter {
  public FileOutboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public void processMessage (String message, String channelId)throws AdapterException {
  }
  public void terminate () throws AdapterException {
  }
  public void commit() throws AdapterException {
  }
  public void rollback() throws AdapterException {
  }
}
```

To create an outbound adapter, the class needs to extend the class **com.newscale.is.adk.base.OutboundAdapter** as shown above.

Implement a constructor that receives a **com.newscale.is.adk.AdapterContext** as a parameter. The recommended way to implement this constructor is also shown above: calling the super constructor.

Implement the initiate method as shown above. This method is called when an agent using the adapter is started. If this method is empty, you can omit it.

Implement the **processMessage** method. This method is called when a message is ready to be sent. If a transformer is specified in the agent, the transformer has transformed the message.

Implement the **terminate** method. Call this method is when the agent stops. If this method is empty, you can omit it.

Implement the **commit** method. Call this method when the agent is about to complete its transaction. If this method is empty, you can omit it. This method is used so that a transaction can be started in the **processMessage**, and later completed.

Implement the **rollback** method. This method is called when the agent is about to rollback its transaction. If this method is to be left empty, it can be omitted. This method is used so that a transaction can be started in the **processMessage**, and later recalled.

For more information about transaction support, see Configuring Transaction Notification.

In our case, the file outbound class writes a file with the contents of the xml. To achieve that, first set up a variable that keeps the file name where the file is stored. For this purpose, use the agent properties.

```
String path = null;
 public void initiate (AdapterContext context)
   throws AdapterException {
   Properties properties = context.getProperties();
   this.path = properties.getProperty("OB_FILE_DIR") + "/" +
       properties.getProperty("OB_FILE_NAME");
 }
```

When the string is received, writing it to the file is trivial.

```
  public void processMessage (String message, String channelId)
    throws AdapterException {
    try {
      Writer w = new FileWriter(path);
      w.write(message);
      w.close();
    } catch (Exception e) {
      e.printStackTrace();
      throw new AdapterException(1, "Problem while writing to a file: " +
          e.getMessage());
    }
}
```

Of course, this code has been oversimplified for the sake of explanation.

Creating Poller Inbound Adapter Class

The skeleton for our inbound adapter is as follows:

```
public class FileInboundAdapter extends InboundAdapter {
  public FileInboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public String receiveMessage () throws AdapterException {
      return null;
    }
  }
  public void terminate () throws AdapterException {
  }
  public void commit()
    throws AdapterException {
  }
```

```
  public void rollback()
    throws AdapterException {
  }
}
```

The semantics of the methods are just like the ones in the outbound adapter. The only exception is the **receiveMessage** method. The **receiveMessage** method is called periodically in the case of a poller adapter. If data is found, then the method returns a valid xml in third-party format. If no data is found, null is returned. The code for the inbound adapter is as follows (just like the outbound adapter, the initialization is done with the correct parameters):

```
String path = null;
public void initiate (AdapterContext context)
  throws AdapterException {
  Properties properties = context.getProperties();
  this.path = properties.getProperty("IB_FILE_DIR") + "/" +
      properties.getProperty("IB_FILE_NAME");
}
```

Processing of the file is done as follows:

```
public String receiveMessage () throws AdapterException {
   String receivedMessage = "";
   char data[] = {};
   try {
     StringBuffer buffer = new StringBuffer();
     FileInputStream fis = new FileInputStream(path);
     InputStreamReader isr = new InputStreamReader(fis, "UTF8");
     Reader in = new BufferedReader(isr);
     int ch;
     while ((ch = in.read()) > -1) {
       buffer.append((char) ch);
     }
     in.close();
     String requestString = buffer.toString();
     boolean success = (new File(path)).delete();
     return requestString;
   } catch (Exception e) {
     return null;
   }
}
```

Creating Listener Inbound Adapter

A listener adapter is created by virtue of an ad-hoc process. Two classes are in play: the inbound adapter class, and an actual receiver class, like a servlet. The receiver class is required to obtain the channel ID. The receiver class locates the InboundAdapter class as follows:

```
ChannelInfoVO chVo = AgentDAO.getInstance().getChannelInfo(channelId);
 if(chVo != null){
   Adapter adapter =
       AgentManager.getInstance().getAdapter(chVo.getAgentId());
   ((InboundAdapter).receiverProcess(xml);
 }
```

The inbound adapter has a method called **receiverProcess** that should be called with the message, or an object whose **toString()** method returns the text of the message. The example does not provide a listener inbound adapter.

Implementing Exception Handler

Once the two adapters are done, the exception handler needs to be implemented. In our case it is a very simple class, where all we do is output the error. The complete class is shown here:

```
public class FileExceptionHandler implements ITransportExceptionHandler {
  public FileExceptionHandler () {
  }
  public void handleException (Map props, String message) {
    System.out.println("Outbound Message Failed to deliver: " + message);
  }
}
```

Configuring Transaction Notification

Transaction support has been provided to the adapters so that agents get notified before they undo their own transactions. The methods **commit** and **rollback** have been added for that purpose.

Note No logic code should be added to these methods, as the system is in the middle of committing or rolling back a transaction. These methods should only rollback or commit their resources.

To track resources to be committed or rolled back, an adapter can use this common technique:

Create a static map. Once the processing method is called (either processMessage or receiveMessage) the method can add:

```
private static Map resources = new HashMap();
public void processMessage (String message, String channelId)
  throws AdapterException {
  Connection con = … // obtain a connection to external resource
  Map.put(Thread.getCurrentThread(), con);
}

public void commit() throws AdapterException {
  con = (Connection) map.get(Thread.getCurrentThread());
  map.remove(Thread.getCurrentThread());
  con.commit();
}
```

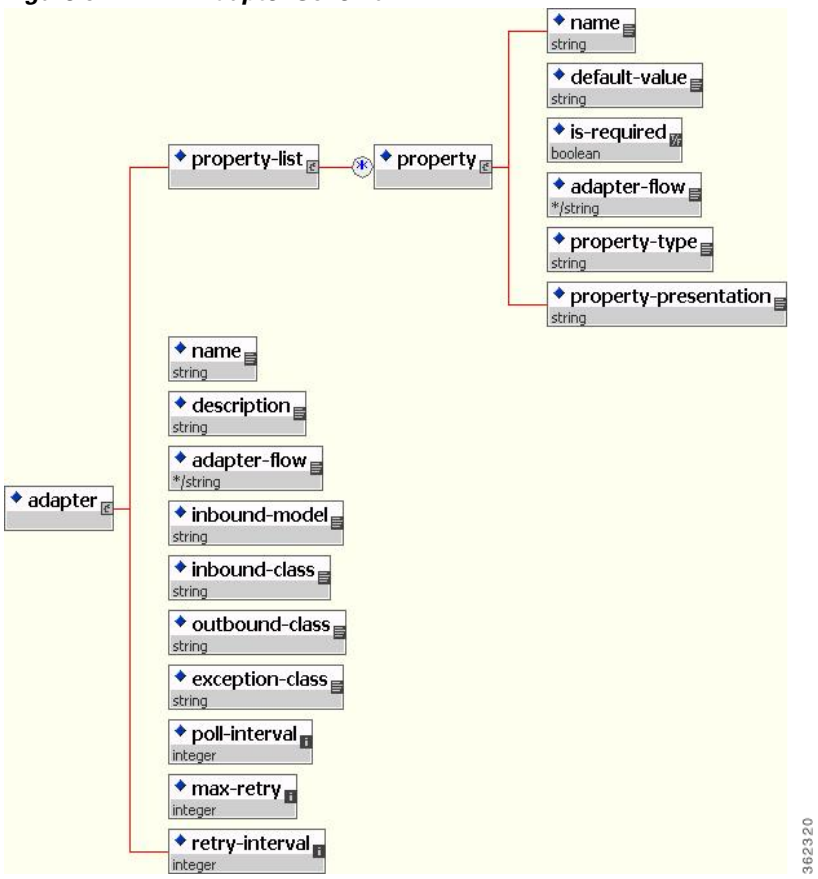
Understanding the adapter.xml Descriptor

The adapter descriptor contains information for the deployment of the adapter and its properties.

The Adapter Schema

The adapter schema is as follows:

Figure 6-2 Adapter Schema



Description of "adapter" Element Fields

name: Name of the adapter

description: Description of the adapter

adapter-flow:

Valid values for this are:

- "inbound"
- "outbound"

inbound-model:

Valid values are:

- "listener"
- "poller"
- "extendedpoller"

inbound-class: Absolute class name of inbound adapter

outbound-class: Absolute class name of outbound adapter

exception-class: Absolute class name of exception handler for this adapter

poll-interval: Poll interval (applicable for “poller” type adapter) in milliseconds

max-retry: Max number of retries in case of message failure

retry-interval: Interval between retries in milliseconds

Description of “property” (Adapter Properties) Element Fields

name: Name of the adapter property

default-value: Default value for the property

is-required: Whether this is a mandatory or optional property. Valid values are “true” or “false”

property-type: The type of property. Valid values are “string” for now.

property-presentation: Valid values are “text” and “password”

adapter-flow:

Valid values are:

- “inbound”
- “outbound”

Adapter.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter>
<property-list>
<property>
<name>InboundFinalResolution</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OnError</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
```

```
<name>BackupSuffix</name>
<default-value>.bak</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>FileNameDateFormat</name>
<default-value>.yyyyMMddHHmmssSSS</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundConflictResolution</name>
<default-value>Rename</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
</property-list>
<name>File Adapter</name>
<description>Read/write the external data from/to external file system</description>
<adapter-flow>inbound</adapter-flow>
<inbound-model>poller</inbound-model>
<inbound-class>com.newscale.is.adapter.file.FileInboundAdapter</inbound-class>
<outbound-class>com.newscale.is.adapter.file.FileOutboundAdapter</outbound-class>
<exception-class>com.newscale.is.adapter.file.FileExceptionHandler</exception-class>
<poll-interval>10000</poll-interval>
```

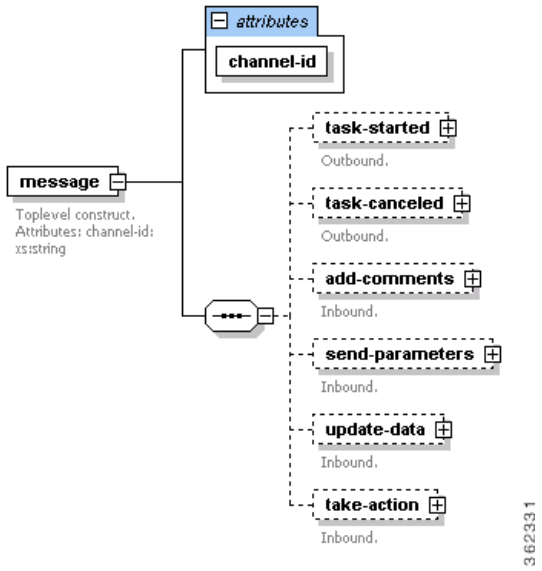
```
<max-retry>0</max-retry>
<retry-interval>0</retry-interval>
</adapter>
```

Understanding Communication Message Content and Structure

This section describes the details of the communication message content and structure. The message content is encapsulated in XML documents which are sent between Service Catalog and third-party systems over various carrier protocols such as HTTP, SOAP, or JMS. For easier understanding of the structures and substructures of messages, a graphical notation is used.

Message

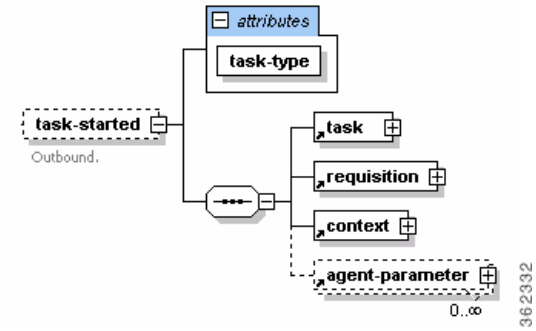
Figure 6-3 Message



The outbound \has a top level element message which contains the element “task-started” or “task-canceled”. The inbound document has a top level element message which contains one or many elements “add-comments,” “send-parameters,” or “take-action”. The message tag has a mandatory attribute which is called channel-id and is of type string. It is a unique string value created by Service Link for any outbound message created. The third-party system needs to reply back the message with the corresponding channel-id. This ID has to be carried on both the Service Catalog and third-party system sides.

Task Started or Task Cancelled

Figure 6-4 Task Started or Task Cancelled



Task started kicks off an external activity in the third-party system. The design strategy followed for this operation is to incorporate all the information that may be required by the third-party system to execute the task. This element holds all the required details about the task and the requisition it belongs to. It may also contain one or more optional agent parameters. The context element describes the task in context of service delivery plan. This node does not contain values for Requisition-level reviews and approvals.

Task

Figure 6-5 Task

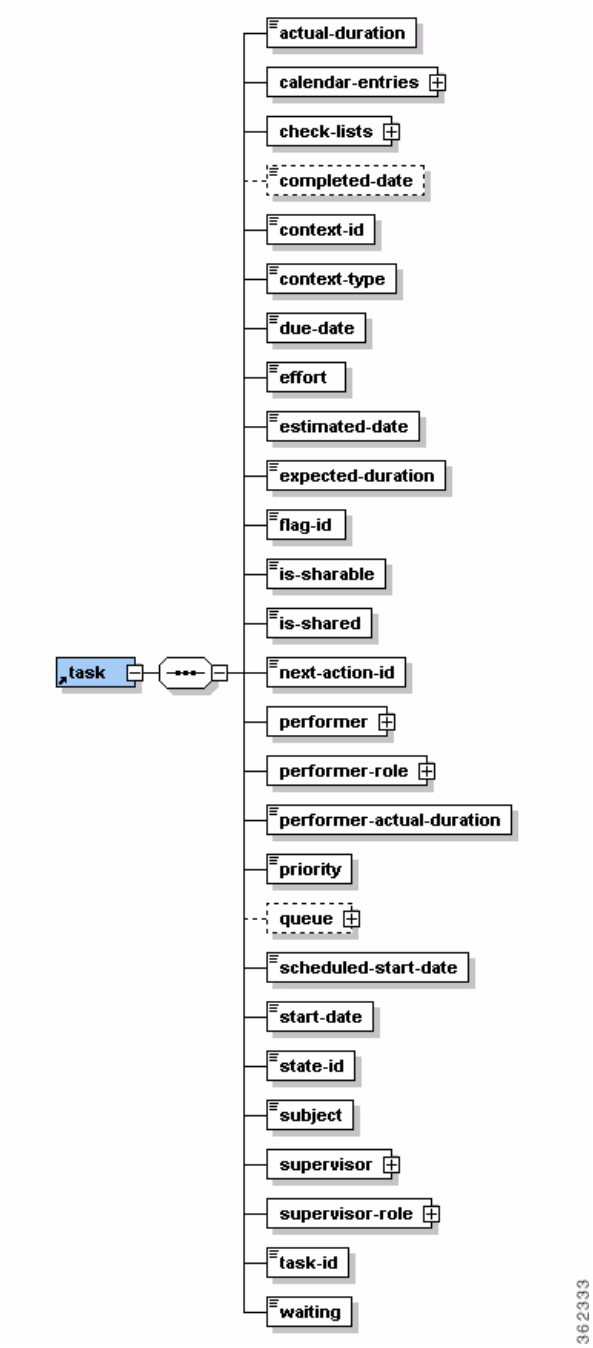
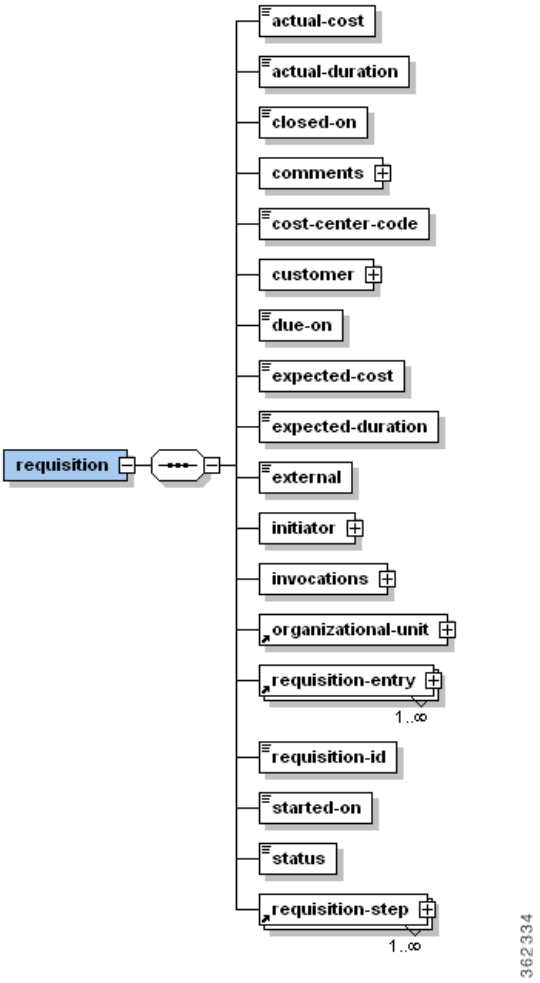


Table 6-4 Task Element and Description

Element	Description
actual-duration	Empty because the task has just become ongoing.
calendar-entries	The calendar entry of the person who requested a service.
check-lists	Task check list.
completed-date	Empty because the task has not yet been completed.
context-id	The object id of the context object in which the task gets initiated.
context-type	The object context, for example, Requisition Entry.
due-date	The due date for the task.
effort	Expected task effort in hours (how many working hours are expected to be required by one person to complete the task?).
estimated-date	Estimated completion date and time of the task.
expected-duration	Expected task duration in hours (how many working hours are expected to pass from the beginning of the task to the end?)/.
flag-id	Color indicator for the UI.
is-sharable	Boolean value indicating whether the task is sharable or not.
is-shared	Boolean value indicating whether the task is shared or not.
next-action-id	What is the next possible action for the task?
performer	The performer of the task. The performer element has an associated person object/.
performer-actual-duration	Empty because the task has not yet been completed/.
performer-role	What is the process role the performer is fulfilling?
priority	Task priority: 1, 2 or 3 for high, medium, or low, respectively/.
queue	Description of the queue to which this task has been assigned.
scheduled-start-date	Date on which the task is scheduled to be started/.
start-date	Date on which the task was started/.
state-id	Which state the task is in/.
subject	Subject of the task.The subject changes with the service definition, not with the requisition entry/.
supervisor	The supervisor of the task.The supervisor element has an associated person object.
supervisor-role	The process role the supervisor is fulfilling.
task-id	An integer used to uniquely identify this task instance. A new number is generated for each task of a requisition entry.
waiting	An indicator that represents the dependencies of this task including sub tasks and third-party systems.

Requisition

Figure 6-6 Requisition



The requisition element encapsulates all requisition and requisition entry data that can be used for integration purposes when executing an external activity.

Table 6-5 Requisition Element Description Table

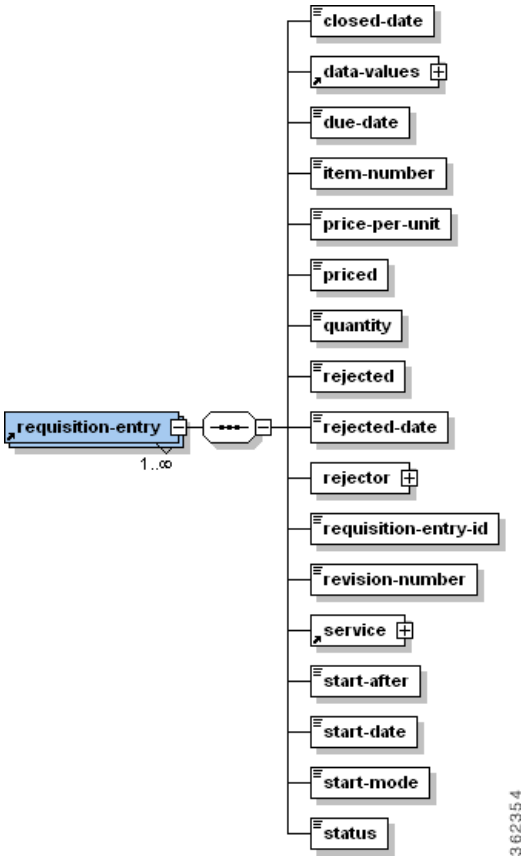
Element	Description
services	Number of services (or requisition entries) requested.
actual-cost	Actual cost of the requisition.
actual-duration	Actual duration of the requisition.
closed-on	Empty, as the requisition has not yet been completed.
comments	Comment on the requisition.
cost-center-code	Not used.
customer	The person for whom the requisition was ordered. It holds the person object data.
due-on	Date and time when the delivery of the requisition is due.
expected-cost	Expected cost of the requisition.

Table 6-5 Requisition Element Description Table

Element	Description
expected-duration	Expected duration in hours for handling the whole requisition.
external	Boolean value indicating whether the requisition has been initiated from an external system.
initiator	The person who ordered this requisition. It holds the person object data.
invocations	Attributes set up through RAPI (Requisition API).
organizational-unit	The organizational unit of the requestor (initiator).
requisition-entry	The requisition entry data.
requisition-id	Integer id of the submitted requisition. This is the same ID that can be seen in My Services and Service Manager after submitting a requisition manually.
requisition-step	The requisition authorization/delivery steps and status.
started-on	The date on which the requisition started.
status	State the requisition is currently in. While executing an external activity, this is ongoing.

Requisition Entry

Figure 6-7 Requisition Entry



This tag encapsulates all the data of one requisition entry that can be used for integration purposes.

Table 6-6 Requisition Entry Element Description Table

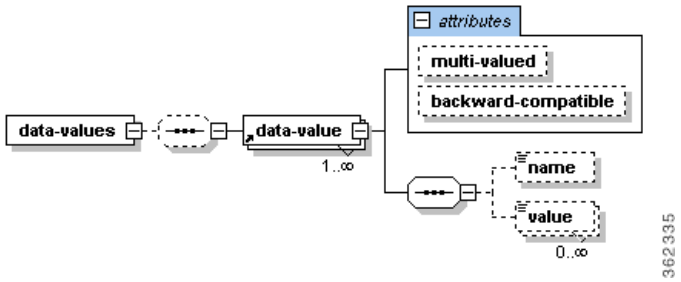
Element	Description
closed-date	Date and time when the requisition entry's status was changed from ongoing to completed. It is empty because the requisition is not closed when the task is ongoing.
data-values	Requisition entry data value.
due-date	Date on which this requisition is supposed to finish.
item-number	Item number of the requisition entry within the requisition.
price-per-unit	Unit price of the service requested.
priced	True if the price has been established and false if pricing is not done on the requisition.
quantity	Quantity ordered.
rejected	Indicates whether the requisition entry is approved or rejected.
rejected-date	If it is rejected, on what date.
rejector	Indicates the person who rejected the requisition.

Table 6-6 Requisition Entry Element Description Table

requisition-entry-id	Entry ID.
revision-number	If the revision is made it indicates the revision number.
service	Element related to the service which this requisition entry belongs to.
start-after	Delayed start date.
start-date	The date on which the requisition entry got started.
start-mode	Specifies if the requisition entry starts immediately or late.
status	Status of the requisition entry: closed or ongoing.

Data Values

Figure 6-8 Data Values



The data-values element can have one or more data values comprised of dictionary data. The data-value name indicates the “Dictionaryname.FieldName” and value is the value entered by the user while ordering the service. If the value is a multi-select drop down list, then one data-value element can have multiple values.

Service

Figure 6-9 Service

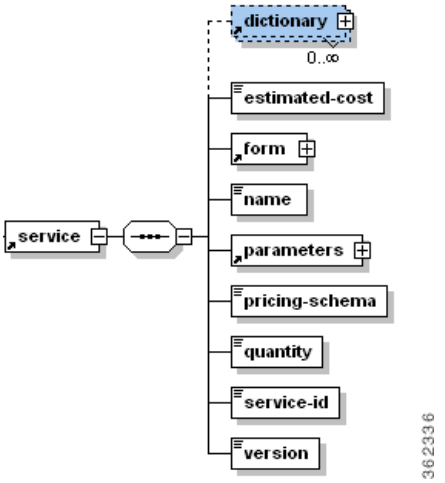


Table 6-7 Service Element Description Table

Element	Description
dictionary	A service element can have zero or more dictionaries.
estimated-cost	The estimated cost of the service.
form	Element which holds all the field elements of the service form.
name	Name of the service.
parameters	Parameters defined for this service.
pricing-schema	Specifies if the service is a bid, pricing task or fixed price.
quantity	How many quantities were ordered?
service-id	Id of the service in Service Catalog.
version	Last modified version number of the service.
standard-duration	Standard duration for the service.

Dictionary

Figure 6-10 Dictionary

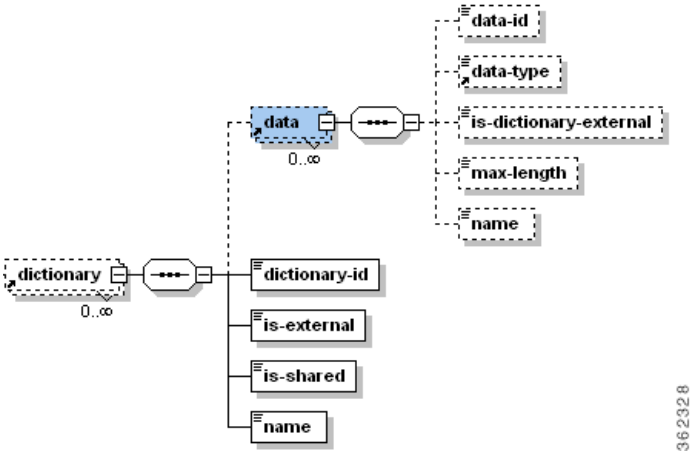


Table 6-8 Dictionary Element Description Table

Element	Description
caption	A string value containing the caption data within the dictionary.
data	The data elements within the dictionary. The data element holds values for the data element name, maximum length, data type and other metadata.
dictionary-id	The dictionary id of a dictionary within Service Catalog.
dictionary-template-type-id	The template used for creating the dictionary (for example, 2 for person-based dictionaries).
classification-id	The dictionary classification (applicable to Service Item dictionaries only).
mdr-data-type-id	The dictionary service item type (applicable to Service Item dictionaries only).
display-order	An integer value containing the display order of the dictionary.
is-external	A Boolean value which indicates whether the dictionary is an internal Service Catalog dictionary or is external.
is-reportable	A Boolean value stating whether the dictionary has been marked as reportable for use with the Advanced Reporting module's Ad-Hoc reporting feature.
is-shared	A Boolean value which indicates whether the dictionary is a shared dictionary or not.
is-template	A Boolean value which indicates whether the dictionary is a template; the value is always false.
logic-name	Internal name of the dictionary (applicable to reserved dictionaries only).
name	Name of the dictionary.

Form

Figure 6-11 Form

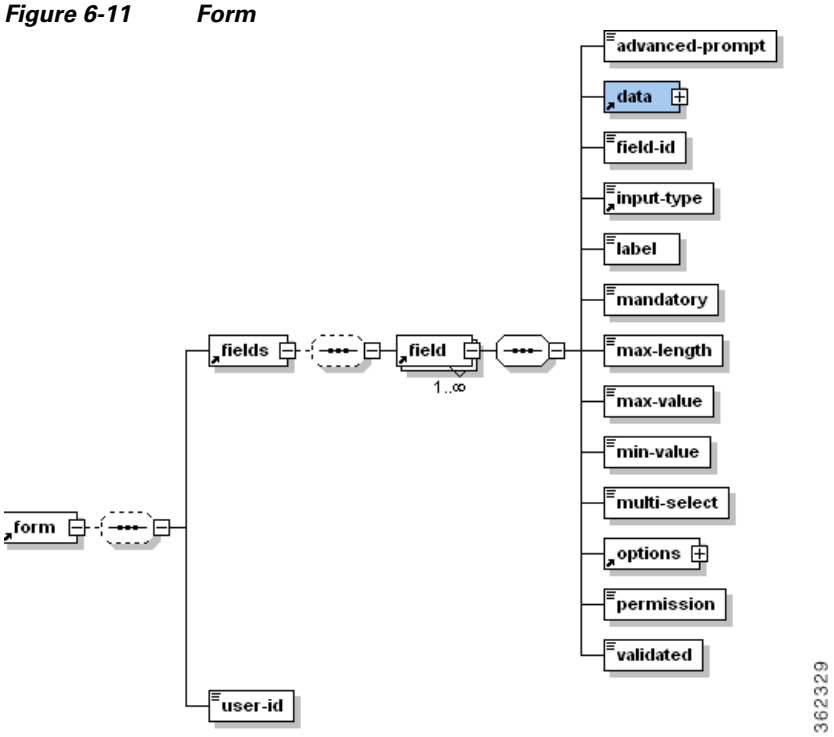
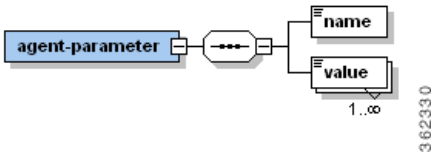


Table 6-9 Form Element Description Table

Element	Description
fields	Fields have one or many field elements inside a requisition form. advanced-prompt – Rich html prompt. data – holds the data for the field which has data type, name, length, and so on. dictionary-display-order – The value for dictionary-display-order is the value of DefObjectDictionaries.DisplayOrder for the Dictionary associated with the DataElement associated with the Field. display-order – the value for **display-order** is the value of DefObjectDataHTML.DisplayOrder for the field. field-id – Field Id within the Service Catalog database. input-type – Input type of the field (for example, text, option, and so on). label – Label specified for the field. mandatory – The field data is mandatory in the service. max-length – Maximum length specified for the field. max-value – If it is a number range specified. min-value multi-select – Whether the input type is a multi-select box. options – The option list available for this data field. permission – validated – Should it be validated or not.
user-id	

Agent Parameter

Figure 6-12 Agent Parameter



The agent parameter represents the external parameters specified for the agent. It has the Boolean attribute called multi-valued which is either true or false based on whether this parameter has multiple values selected by user. The name represents the name of the agent parameter and value represents its value.

Examples: Inbound and Outbound Documents

task-started or task-canceled (outgoing)

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1124919814742:-32752"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <task-started task-type="task">
        <task>
            <actual-duration>0.0</actual-duration>
            <calendar-entries>
                <calendar-entry>
                    <calendar-entry-id>2</calendar-entry-id>
                    <date>Thu Aug 25 17:00:00 PDT 2005</date>
                    <end-time>Fri Aug 26 21:40:37 PDT 2005</end-time>
                    <is-blocked>false</is-blocked>
                    <is-break>false</is-break>
                    <is-read>false</is-read>
                    <person>
                        <company-address/>
                        <email>admin@company.com</email>
                        <fax/>
                        <first-name>admin</first-name>
                        <home-ou>
                            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                            <organizational-unit-id>1</organizational-unit-id>
                        </home-ou>
                        <home-phone/>
                        <last-name/>
                        <login-name>admin</login-name>
                        <person-id>1</person-id>
                        <personal-address/>
                        <supervisor-name/>
                        <timezone>Pacific Standard Time</timezone>
                        <work-phone/>
                    </person>
                    <sequence>0</sequence>
                    <start-time>Thu Aug 25 21:40:37 PDT 2005</start-time>
                    <subject>External Task</subject>
                </calendar-entry>
            </calendar-entries>
            <check-lists>
                <check-list-entry>
                    <display-order>1</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you wake up</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>2</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you take a shower</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>3</display-order>
```

```
                        <is-mandatory>true</is-mandatory>
                        <last-date/>
                        <last-person/>
                        <name>Make sure you have breakfast</name>
                        <status>false</status>
                    </check-list-entry>
                </check-lists>
                <completed-date/>
                <context-id>1</context-id>
                <context-type>Requisition Entry</context-type>
                <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
                <effort>10.0</effort>
                <estimated-date/>
                <expected-duration>10.0</expected-duration>
                <flag-id>0</flag-id>
                <is-sharable>false</is-sharable>
                <is-shared>false</is-shared>
                <next-action-id>2</next-action-id>
                <performer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </performer>
                <performer-actual-duration>0.0</performer-actual-duration>
                <priority>2</priority>
                <scheduled-start-date>Thu Aug 25 21:40:37 PDT 2005</scheduled-start-date>
                <start-date>Wed Aug 24 21:42:15 PDT 2005</start-date>
                <state-id>2</state-id>
                <subject>External Task</subject>
                <supervisor>
                    <company-address>Foo Bar 25 Suite 300 Foo City CA  94404
USA</company-address>
                    <email>internal@company.com</email>
                    <fax/>
                    <first-name>Monkey</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name>McBride</last-name>
                    <login-name>monkey</login-name>
                    <person-id>3</person-id>
                    <personal-address>Fuchi Caca 16 Apartment C Fuchi Minn OR  78787
USA</personal-address>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </supervisor>
                <task-id>3</task-id>
                <waiting>1</waiting>
```

```
            </task>
            <requisition>
                <actual-cost>0.0</actual-cost>
                <actual-duration>0.0</actual-duration>
                <closed-on/>
                <comments>
                    <comment>
                        <comment-date>Wed Aug 24 21:42:06 PDT 2005</comment-date>
                        <comment-id>1</comment-id>
                        <comment-text>I am adding a comment and I cannot think of a better
comment</comment-text>
                        <component-id>3</component-id>
                        <component-name>Request Center Component</component-name>
                        <person>
                            <company-address/>
                            <email>admin@company.com</email>
                            <fax/>
                            <first-name>admin</first-name>
                            <home-ou>
                                <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                                <organizational-unit-id>1</organizational-unit-id>
                            </home-ou>
                            <home-phone/>
                            <last-name/>
                            <login-name>admin</login-name>
                            <person-id>1</person-id>
                            <personal-address/>
                            <supervisor-name/>
                            <timezone>Pacific Standard Time</timezone>
                            <work-phone/>
                        </person>
                        <source-object-id>2</source-object-id>
                        <source-object-inst-id>1</source-object-inst-id>
                    </comment>
                </comments>
                <cost-center-code/>
                <customer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </customer>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <expected-cost>0.0</expected-cost>
                <expected-duration>0.0</expected-duration>
                <external>false</external>
                <initiator>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
```

```
            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
            <organizational-unit-id>1</organizational-unit-id>
        </home-ou>
        <home-phone/>
        <last-name/>
        <login-name>admin</login-name>
        <person-id>1</person-id>
        <personal-address/>
        <supervisor-name/>
        <timezone>Pacific Standard Time</timezone>
        <work-phone/>
    </initiator>
    <invocations/>
    <organizational-unit>
        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
        <organizational-unit-id>1</organizational-unit-id>
    </organizational-unit>
    <requisition-entry>
        <closed-date/>
        <data-values>
            <data-value>
                <name>Requester</name>
                <value>John McGarzafi</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.TicketID</name>
                <value>None yet</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.AssetNumber</name>
                <value>123456789</value>
            </data-value>
        </data-values>
        <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
        <item-number>1</item-number>
        <price-per-unit>0.0</price-per-unit>
        <priced>true</priced>
        <quantity>1</quantity>
        <rejected>false</rejected>
        <rejected-date/>
        <rejector>
            <company-address/>
            <email/>
            <fax/>
            <first-name/>
            <home-phone/>
            <last-name/>
            <login-name/>
            <person-id>0</person-id>
            <personal-address/>
            <supervisor-name/>
            <timezone/>
            <work-phone/>
        </rejector>
        <requisition-entry-id>1</requisition-entry-id>
        <revision-number>5</revision-number>
        <service>
            <dictionary>
                <data>
                    <data-id>3</data-id>
                    <data-type>Person</data-type>
                    <is-dictionary-external>false</is-dictionary-external>
                    <max-length>100</max-length>
                    <name>Requester</name>
```

```
    </data>
    <dictionary-id>1</dictionary-id>
    <is-external>false</is-external>
    <is-shared>false</is-shared>
    <name>Monkey Service (private)</name>
</dictionary>
<dictionary>
    <data>
        <data-id>1</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>TicketID</name>
    </data>
    <data>
        <data-id>2</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>AssetNumber</name>
    </data>
    <dictionary-id>2</dictionary-id>
    <is-external>false</is-external>
    <is-shared>true</is-shared>
    <name>RemedyStuff</name>
</dictionary>
<estimated-cost>0.0</estimated-cost>
<form>
    <fields>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>2</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>AssetNumber</name>
            </data>
            <field-id>2</field-id>
            <input-type>text</input-type>
            <label>AssetNumber</label>
            <mandatory>false</mandatory>
            <max-length>50</max-length>
            <max-value>0.0</max-value>
            <min-value>0.0</min-value>
            <multi-select>false</multi-select>
            <options>
                <available-keys/>
                <available-labels/>
                <current-values/>
                <multivalued>false</multivalued>
            </options>
            <permission>4</permission>
            <validated>true</validated>
        </field>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>1</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>TicketID</name>
            </data>
```

```
                        <field-id>1</field-id>
                        <input-type>text</input-type>
                        <label>TicketID</label>
                        <mandatory>false</mandatory>
                        <max-length>50</max-length>
                        <max-value>0.0</max-value>
                        <min-value>0.0</min-value>
                        <multi-select>false</multi-select>
                        <options>
                            <available-keys/>
                            <available-labels/>
                            <current-values/>
                            <multivalued>false</multivalued>
                        </options>
                        <permission>4</permission>
                        <validated>true</validated>
                    </field>
                    <field>
                        <advanced-prompt>Give the name!</advanced-prompt>
                        <data>
                            <data-id>3</data-id>
                            <data-type>Person</data-type>
                            <is-dictionary-external>false</is-dictionary-external>
                            <max-length>100</max-length>
                            <name>Requester</name>
                        </data>
                        <field-id>3</field-id>
                        <input-type>text</input-type>
                        <label>Requester Name</label>
                        <mandatory>false</mandatory>
                        <max-length>100</max-length>
                        <max-value>0.0</max-value>
                        <min-value>0.0</min-value>
                        <multi-select>false</multi-select>
                        <options>
                            <available-keys/>
                            <available-labels/>
                            <current-values>
                                <string>John McGarzafi</string>
                            </current-values>
                            <multivalued>false</multivalued>
                        </options>
                        <permission>4</permission>
                        <validated>true</validated>
                    </field>
                </fields>
                <user-id>0</user-id>
            </form>
            <name>Monkey Service</name>
            <parameters>
                <default-duration>0.0</default-duration>
                <priority>2</priority>
                <start-date/>
                <start-mode>0</start-mode>
            </parameters>
            <pricing-schema>0</pricing-schema>
            <quantity>1</quantity>
            <service-id>1</service-id>
            <version>5</version>
        </service>
        <start-after/>
        <start-date>Wed Aug 24 21:40:50 PDT 2005</start-date>
        <start-mode>0</start-mode>
        <status>1</status>
```

```
            </requisition-entry>
            <requisition-id>1</requisition-id>
            <started-on>Wed Aug 24 21:40:32 PDT 2005</started-on>
            <status>1</status>
            <requisition-step>
                <completed-on/>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <estimated-on>Fri Aug 26 21:40:37 PDT 2005</estimated-on>
                <name>Delivery project for Monkey Service</name>
                <status>2</status>
            </requisition-step>
        </requisition>
        <agent-parameter multi-valued="false">
            <name>Ticket</name>
            <value>None yet</value>
        </agent-parameter>
        <agent-parameter multi-valued="false">
            <name>Asset</name>
            <value>123456789</value>
        </agent-parameter>
    </task-started>
</message>
```

take-action (incoming)

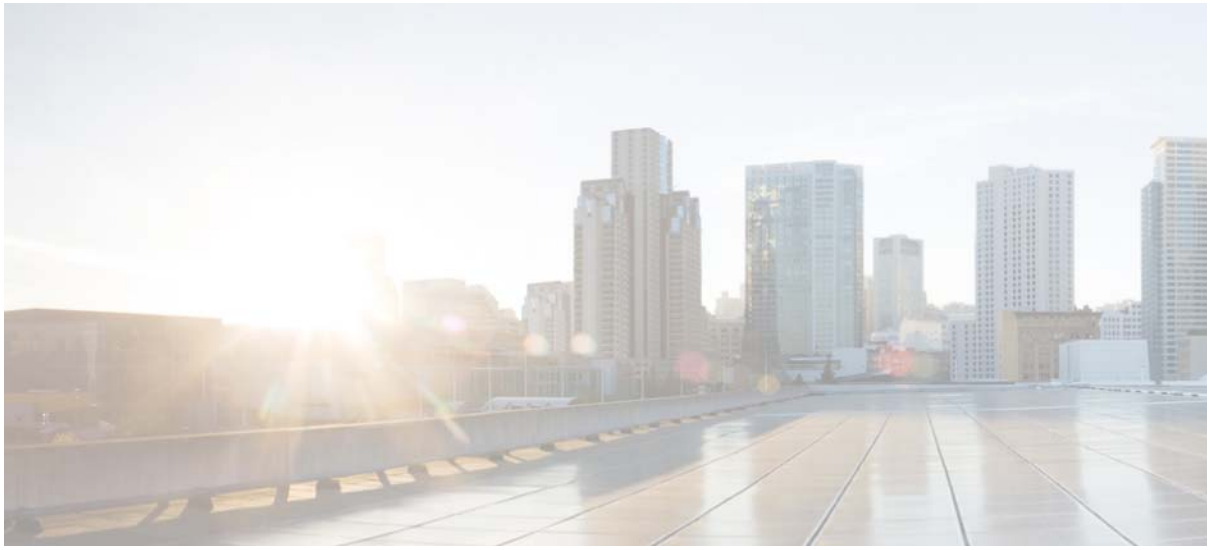
```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1124919814742:-32752">
            <take-action action="done"/>
</message>
```

send-parameters (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1116468068789:-32360">
    <send-parameters>
      <agent-parameter>
         <name>Param1</name>
             <value>cat</value>
      </agent-parameter>
      <agent-parameter>
         <name>Param2</name>
             <value>catlitter</value>
       </agent-parameter>
     </send-parameters>
</message>
```

add-comments (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="32580443:1116276793649:-32629">
            <add-comments>
               <comment>Test Comment</comment>
            </add-comments>
</message>
```



PART 3

Integrating with External Systems

Developing Integration with External Systems Using JSR Portlets

The Portal Designer solution within Service Catalog provides a rich platform for integrating with external applications through JSR Portlets. The portal front-end uses Apache Pluto 1.1 libraries as the framework. Portlets developed using APIs which meet the Java Portlet Specification (JSR168, JSR286) standards may be deployed along with Service Catalog. Once deployed, these will appear in Portal Designer as "Third-Party Portlets" and can be added to portal pages. For more information on how to maintain JSR portlets and other content in the Portal Designer solution, see the *Cisco Prime Service Catalog Designer Guide*.

This chapter covers some guidelines on the development and deployment of JSR portlets for the Portal Designer solution. A sample portlet named "MyJSR" is used throughout the chapter as an illustration. The portlet is developed with Spring 3.0 Annotation-based Controller and Sencha's Ext JS—the JavaScript framework for the portal front-end.

Portlet Structure and Packaging

The portlet files should be packaged according to the JSR 168 or 286 specifications, in the form of web application (war) files appropriate for the application server used. A typical portlet war file may include servlets, resource bundles, images, html, jsp, css files, and so on.

JBoss Application Server

Here is the anatomy of a simple portlet named "MyJSR.war":

```
1. css
    MyJSR.css
2. images
    <Custom Images that the Portlet needs can be placed here>
3. js
    MyJSRCreatePersonView.js
    MyJSREdit.js
    MyJSRHelp.js
    MyJSRView.js
4. WEB-INF
    classes
        com
            myjsr
                MyJSRController.class
        config
```

```
        spring
            MyJSRApplicationContext.xml
    jsrportlet.properties
    log4j.properties
jsp
    MyJSREdit.jsp
    MyJSRHelp.jsp
    MyJSRView_listperson.jsp
    MyJSRView_updateperson.jsp
lib
    newscale_appclient.jar
    newscale_core.jar
    cxf-2.2.12.jar
    cxf-rt-transports-http-2.2.12.jar
    pluto-portal-driver-2. 0.2.jar
    org.springframework.aop-3.1.0.RELEASE.jar
    org.springframework.asm-3.1.0.RELEASE.jar
    org.springframework.aspects-3.1.0.RELEASE.jar
    org.springframework.beans-3.1.0.RELEASE.jar
    org.springframework.context-3.1.0.RELEASE.jar
    org.springframework.context.support-3.1.0.RELEASE.jar
    org.springframework.core-3.1.0.RELEASE.jar
    org.springframework.expression-3.1.0.RELEASE.jar
    org.springframework.instrument-3.1.0.RELEASE.jar
    org.springframework.instrument.tomcat-3.1.0.RELEASE.jar
    org.springframework.jdbc-3.1.0.RELEASE.jar
    org.springframework.jms-3.1.0.RELEASE.jar
    org.springframework.orm-3.1.0.RELEASE.jar
    org.springframework.oxm-3.1.0.RELEASE.jar
    org.springframework.test-3.1.0.RELEASE.jar
    org.springframework.transaction-3.1.0.RELEASE.jar
    org.springframework.web-3.1.0.RELEASE.jar
    org.springframework.web.portlet-3.1.0.RELEASE.jar
    org.springframework.web.servlet-3.1.0.RELEASE.jar
    org.springframework.web.struts-3.1.0.RELEASE.jar
tld
    c.tld
    pluto.tld
    portlet.tld
    portlet_2_0.tld
    portlet-el.tld
    portlet-el_2_0.tld
portlet.xml
web.xml
jboss-deployment-structure.xml
```

In this sample portlet, the nsAPI java client—newscale_appclient.jar—is included in the lib folder as the portlet invokes the REST API to retrieve data from Service Catalog. Pluto libraries and other libraries that the nsAPI java client depends on need to be included in the lib folder. In addition, the jboss-deployment-structure.xml is included to describe the dependencies on the JBoss modules.

An additional descriptor—portlet.xml—must be present to specify the portlet-related configurations.

Weblogic Application Server

On WebLogic, the JSR portlet war structure is similar except that it must contain all the libraries the portlet uses because it is deployed outside of the Service Catalog application.

Some of the libraries included below are required for invoking the nsAPI java client and to read portlet common settings. Certain libraries are needed here because the portlet uses JSTL tags in JSPs and Spring portlets MVC.

```
1. css
    MyJSR.css
2. images
    <Custom Images that the Portlet needs can be placed here>
3. js
    MyJSRCreatePersonView.js
    MyJSREdit.js
    MyJSRHelp.js
    MyJSRView.js
4. WEB-INF
    classes
        com
            myjsr
                MyJSRController.class
        config
            spring
                MyJSRApplicationContext.xml
        jsrportlet.properties
        log4j.properties
    jsp
        MyJSREdit.jsp
        MyJSRHelp.jsp
        MyJSRView_listperson.jsp
        MyJSRView_updateperson.jsp
    lib
        newscale_appclient.jar
        newscale_core.jar
        newscale_udkernel.jar
        newscale_compbeans.jar
        newscale_conf.jar
        castor-0.9.5.4.jar
        commons-beanutils-1.8.3.jar
        commons-httpclient-3.1.jar
        commons-logging-1.0.4.jar
        cxf-2.2.12.jar
        ezmorph-1.0.4.jar
        json-lib-2.2.2-jdk13.jar
        jsr311-api-1.0.jar
        neethi-2.0.4.jar
        oscache-2.4.jar
        standard.jar
        wsdl4j-1.6.1.jar
        XmlSchema-1.4.6.jar
        commons-collections-3.2.1.jar
        commons-lang-2.4.jar
        neethi-2.0.4.jar
        castor-0.9.5.4.jar
        jstl.jar
        org.springframework.aop-3.1.0.RELEASE.jar
        org.springframework.asm-3.1.0.RELEASE.jar
        org.springframework.aspects-3.1.0.RELEASE.jar
        org.springframework.beans-3.1.0.RELEASE.jar
        org.springframework.context-3.1.0.RELEASE.jar
        org.springframework.context.support-3.1.0.RELEASE.jar
        org.springframework.core-3.1.0.RELEASE.jar
        org.springframework.expression-3.1.0.RELEASE.jar
        org.springframework.instrument-3.1.0.RELEASE.jar
        org.springframework.instrument.tomcat-3.1.0.RELEASE.jar
        org.springframework.jdbc-3.1.0.RELEASE.jar
        org.springframework.jms-3.1.0.RELEASE.jar
        org.springframework.spring-library-3.1.0.RELEASE.libd
        org.springframework.test-3.1.0.RELEASE.jar
        org.springframework.transaction-3.1.0.RELEASE.jar
        org.springframework.web-3.1.0.RELEASE.jar
```

```
    org.springframework.web.portlet-3.1.0.RELEASE.jar
    org.springframework.web.servlet-3.1.0.RELEASE.jar
    org.springframework.web.struts-3.1.0.RELEASE.jar
tld
    c.tld
    pluto.tld
    portlet.tld
    portlet_2_0.tld
    portlet-el.tld
    portlet-el_2_0.tld
portlet.xml
web.xml
```

Dependent Libraries

The set of libraries required for inclusion in the JSR Portlet war file are available in either the deployed RequestCenter application on the application server or the Service Catalog installer image:

-pluto-taglib-2.0.2.jar: under the "preinstall" folder of the product image

-all other files: RequestCenter.war/WEB-INF/lib

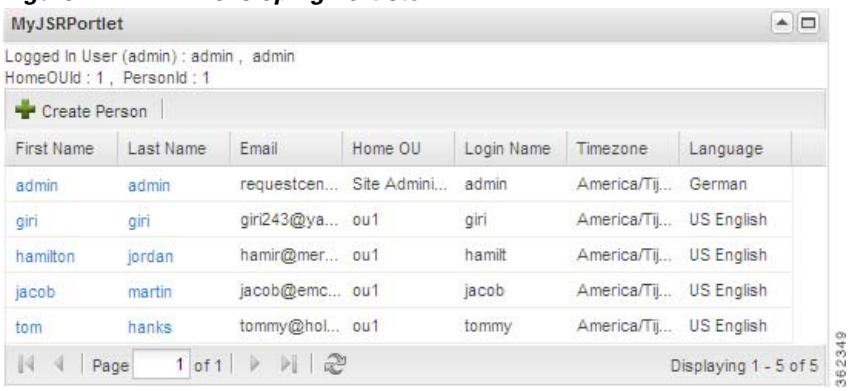
Developing Portlets

A typical JSR portlet should cover the three rendering modes —View, Edit, and Help. In addition, the portlet would support different window states—Normal, Minimized, and Maximized.

The MyJSR portlet example shown below provides a user interface that supports two functionalities:

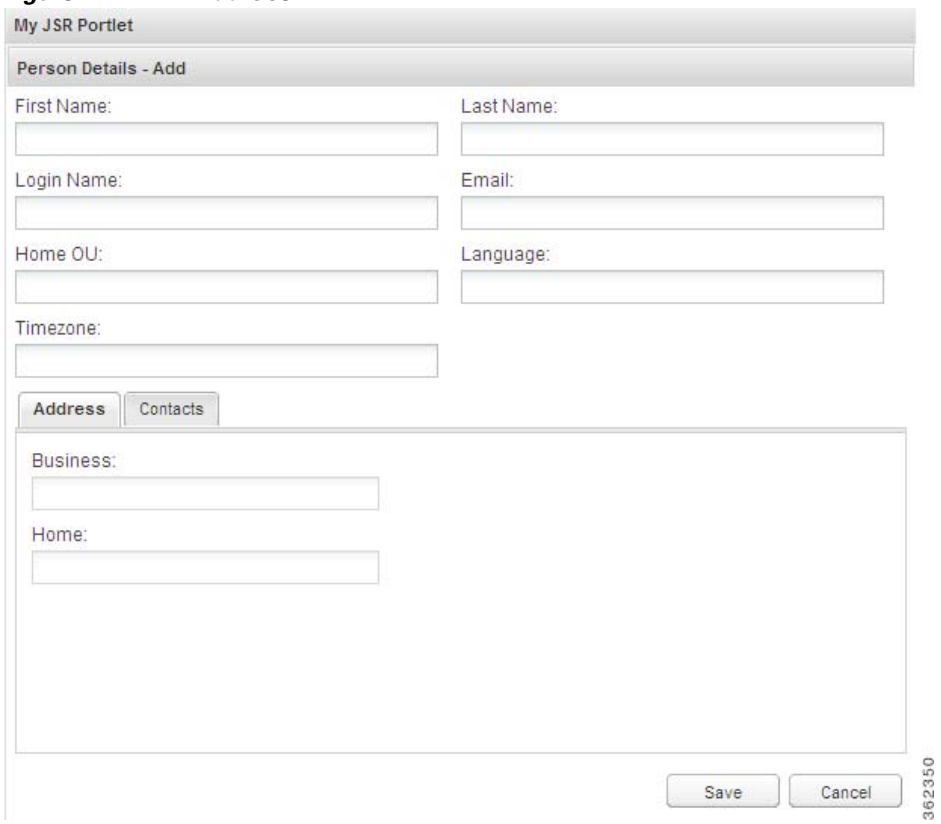
1. List Service Catalog users in a grid.

Figure 7-1 Developing Portlets



2. Allow user to be added/updated.

Figure 7-2 Add User



To achieve the above requirements, the sample code that follows includes these high-level operations:

- Retrieval of Service Catalog users using nsAPI java client
- Returning the user details in JSON format to the user interface
- Rendering the list of users in a Ext JS grid on the browser
- Display/entry of user details in a form designed using Ext JS
- Adding/updating user details in the Service Catalog repository using nsAPI java client

MyJSR.css

#Code Custom Styles for the portlet can be designed here.

Now let us examine the content of each of the components within the MyJSR.war.

MyJSRCreatePersonView.js

Example code using Ext JS to display a form for creating a user.

```
/* Code custom JavaScript for the portlet here */

Ext.onReady(function() {
var tab2 = new Ext.FormPanel({
id : 'personEditForm',
labelAlign : 'top',
```

```
title : 'Person Details - Add',
bodyStyle : 'padding:5px',
width : 600,
renderTo : MyJSREditDiv,
items : [{
layout : 'column',
border : false,
items : [{
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'First Name',
               value : personListObj.firstName,
name : 'firstName',
anchor : '95%'
                                }, {
xtype : 'textfield',
fieldLabel : 'Login Name',
               value : personListObj.login,
name : 'login',
anchor : '95%'
                                }, {
xtype : 'textfield',
fieldLabel : 'Home OU',
name : 'homeOrganizationalUnitName',
               value : personListObj.homeOrganizationalUnitName,
anchor : '95%'
                                }, {
xtype : 'textfield',
fieldLabel : 'Timezone',
name : 'timeZoneName',
               value : personListObj.timeZoneName,
anchor : '95%'
                                }]
                        }, {
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'Last Name',
name : 'lastName',
               value : personListObj.lastName,
anchor : '95%'
                                },{
xtype : 'textfield',
fieldLabel : 'Email',
name : 'email',
               value : personListObj.email,
vtype : 'email',
anchor : '95%'
                                }, {
xtype : 'textfield',
fieldLabel : 'Language',
name : 'languageName',
               value : personListObj.languageName,
anchor : '95%'
                                }]
                        }]
            }, {
xtype : 'tabpanel',
plain : true,
```

```
activeTab : 0,
height : 235,
defaults : {
bodyStyle : 'padding:10px'
              },
items : [{
title : 'Address',
layout : 'form',
defaults : {
width : 230
                        },
defaultType : 'textfield',
items : [{
fieldLabel : 'Business',
name : 'businessAddress',
disabled : true
                                }, {
fieldLabel : 'Home',
name : 'homeAddress',
disabled : true
                                }]
                    }, {
title : 'Contacts',
layout : 'form',
defaults : {
width : 230
                        },
defaultType : 'textfield',
items : [{
fieldLabel : 'Business',
name : 'businessPhone',
disabled : true
                                }, {
fieldLabel : 'Home',
name : 'homePhone',
disabled : true
                                }, {
fieldLabel : 'Mobile',
name : 'mobilePhone',
disabled : true
                                }, {
fieldLabel : 'Fax',
name : 'faxNumber',
disabled : true
                                }]
                    }]
            }],

buttons : [{
text : 'Save',
handler : function() {
Ext.getCmp("personEditForm").getForm().submit({
url : addPersonActionUrl,
params : {},
success : function(form, action) {
var responseObj = Ext.util.JSON.decode(action.response.responseText);
if(responseObj.success == "true")
Ext.Msg.alert('Success', responseObj.successMsg);
else
Ext.Msg.alert('Error', responseObj.errorMsg);
                    }
                  });
               }
             },{
```

```
text : 'Cancel',
handler : function() {
window.location=viewPersonUrl;
                 }
                 }]
   });
});
```

MyJSREdit.js

This JavaScript can be used to add any custom code for portlet edit mode.

```
/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});
```

MyJSRHelp.js

This JavaScript can be used to add any custom code for portlet edit mode.

```
/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});
```

MyJSRView.js

Example JavaScript to display users in Ext JS grid.

```
/* Code custom JavaScript for the Portlet here */
Ext.onReady(function() {
// Demonstrates how to getUser info from Java Script and set it to div
varLogin=document.getElementById('MyJSRLoginNameDiv');
  Login.innerHTML=nsAPP_CurrentUserLoginName;
varFirstName=document.getElementById('MyJSRFirstNameDiv');
  FirstName.innerHTML=nsAPP_CurrentUserFirstName;
varLastName=document.getElementById('MyJSRLastNameDiv');
  LastName.innerHTML=nsAPP_CurrentUserLastName;
varHomeOU=document.getElementById('MyJSRHomeOUDiv');
  HomeOU.innerHTML=nsAPP_CurrentUserHomeOuId;
var PersonID=document.getElementById('MyJSRPersonIDDiv');
  PersonID.innerHTML=nsAPP_CurrentUserId;

var pid = portletId.substr(pidPrefix.length);
if (Ext.getCmp(pid).height && Ext.getCmp(pid).height >= 29) {
var gridHeight = Ext.getCmp(pid).height - 29;
  }

var gridStore = new Ext.data.JsonStore({
proxy : new Ext.data.HttpProxy({
url : pagingUrl,
timeout : connectionTimeOut
```

```
            }),
autoLoad: {params:{start: 0, limit: defaultRecordSize}},
root: 'rows',
totalProperty: 'results',
fields : [{
name : 'firstName',
type : 'string'
                }, {
name : 'lastName',
type : 'string'
                }, {
name : 'email',
type : 'string'
                }, {
name : 'homeOrganizationalUnitName',
type : 'string'
                }, {
name : 'login',
type : 'string'
                }, {
name : 'timeZoneName',
type : 'string'
                }, {
name : 'languageName',
type : 'string'
                }, {
name : 'businessPhone',
type : 'string'
                }, {
name : 'homePhone',
type : 'string'
                }, {
name : 'mobilePhone',
type : 'string'
                }, {
name : 'faxNumber',
type : 'string'
                }, {
name : 'businessAddress',
type : 'string'
                }, {
name : 'homeAddress',
type : 'string'
                }]
        });
gridStore.load();

var expander = new Ext.ux.grid.RowExpander({
tpl : new Ext.Template(
'<h2 class="title">Address</h2><table>',
'<tr><td width=400><b>Business</b> {businessAddress}</td>',
'<td width=400><b>Home</b>{homeAddress}</td></tr></table>',
'<h2 class="title">Contact</h2><table>',
'<tr><td width=400><b>Business</b> {businessPhone}</td>',
'<td width=400><b>Home</b> {homePhone}</td></tr>',
'<tr><td width=400><b>Mobile</b> {mobilePhone}</td>',
'<td width=400><b>Fax</b> {faxNumber}</td></tr></table>')
        });

var gridColModel = new Ext.grid.ColumnModel({
defaults : {
sortable : true,
autoWidth : true
           },
```

```
columns : [{
header : "First Name",
dataIndex : 'firstName'
                  }, {
header : "Last Name",
dataIndex : 'lastName'
                  }, {
header : "Email",
dataIndex : 'email'
                  }, {
header : "Home OU",
dataIndex : 'homeOrganizationalUnitName'
                  }, {
header : "Login Name",
dataIndex : 'login'
                  }, {
header : "Timezone",
dataIndex : 'timeZoneName'
                  }, {
header : "Language",
dataIndex : 'languageName'
                  }]
         });

var gridConfig = {
renderTo :  MyJSREditDiv,
width : "100%",
layout : 'fit',
store : gridStore,
cm : gridColModel,
loadMask: true,
autoWidth : true,
plugins : expander,
tbar : [{
text : 'Create Person',
iconCls : 'add',
handler : function() {
window.location=createNewPersonActionUrl;
                 }
                 },'-'],
bbar : new Ext.PagingToolbar({
pageSize : defaultRecordSize,
store : gridStore,
displayInfo : true,
params:{
start: 0,
limit: defaultRecordSize
                            }
                  })
         };

if ('maximized' == portletWindowState) {
gridConfig.height = document.documentElement.clientHeight - 188;
   } elseif ('normal' == portletWindowState) {
var viewConfig = {
forceFit : true
    };
gridConfig.viewConfig = viewConfig;

if (gridHeight && gridHeight > -1) {
gridConfig.height = gridHeight;
    } else {
gridConfig.autoHeight = true;
    }
```

```
    }

var grid = new Ext.grid.GridPanel(gridConfig);
});
```

portlet.xml

Example of the portlet specification. The portlet-class needs to be set along with a pair of init-params—contextConfigLocation and nsContentPortlet (nsContentPortlet is always set to “false”).

```
<?xmlversion="1.0"encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License.  You may obtain a copy of the License at

http:www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed  under the  License is distributed on an "AS IS" BASIS,
WITHOUT  WARRANTIES OR CONDITIONS  OF ANY KIND, either  express  or
implied.

See the License for the specific language governing permissions and
limitations under the License.
-->
<portlet-app
xmlns="http:java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
version="1.0"
xmlns:xsi="http:www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http:java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
                        http:java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
<portlet>
<description>MyJSR Description</description>
<portlet-name>nsMyJSR</portlet-name>
<display-name>My JSR Portlet</display-name>
<portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
<init-param>
<name>contextConfigLocation</name>
<value>/WEB-INF/classes/config/spring/MyJSRApplicationContext.xml</value>
</init-param>
<init-param>
<name>nsContentPortlet</name>
<value>false</value>
</init-param>
<expiration-cache>-1</expiration-cache>
<supports>
<mime-type>text/html</mime-type>
<portlet-mode>VIEW</portlet-mode>
<portlet-mode>EDIT</portlet-mode>
<portlet-mode>HELP</portlet-mode>
</supports>
<portlet-info>
<title>My JSR Portlet</title>
</portlet-info>
</portlet>
</portlet-app>
```

web.xml

Example deployment descriptor with the servlet and servlet mapping is required by the portal server; in this case, Apache Pluto.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<!DOCTYPEweb-appPUBLIC"-Sun Microsystems, Inc. DTD Web Application
2.3EN""http:java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<display-name>My JSR Portlet Application</display-name>
<description>My JSR Portlet</description>

<!-- Resources bundle base class -->
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
         /WEB-INF/classes/config/spring/MyJSRApplicationContext.xml
</param-value>
</context-param>

<context-param>
<param-name>parameter-name</param-name>
<param-value>parameter-value</param-value>
</context-param>

<servlet>
<servlet-name>ViewRendererServlet</servlet-name>
<servlet-class>
org.springframework.web.servlet.ViewRendererServlet
</servlet-class>
</servlet>

<servlet>
<servlet-name>MyJSR</servlet-name>
<servlet-class>org.apache.pluto.container.driver.PortletServlet</servlet-class>
<init-param>
<param-name>portlet-name</param-name>
<param-value>MyJSR</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>ViewRendererServlet</servlet-name>
<url-pattern>/WEB-INF/servlet/view</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>MyJSR</servlet-name>
<url-pattern>/PlutoInvoker/nsMyJSR</url-pattern>
</servlet-mapping>

<!- Declare Tag libraries that are used in which are going to use in JSP pages-->
<taglib>
<taglib-uri>http://portals.apache.org/pluto</taglib-uri>
<taglib-location>/WEB-INF/tld/pluto.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet_2_0.tld</taglib-location>
</taglib>
```

```
<taglib>
<taglib-uri>/WEB-INF/tld/c.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el_2_0.tld</taglib-location>
</taglib>

</web-app>
```

MyJSREdit.jsp

JSP for portlet edit mode.

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSREdit.js") %>';
```

```
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Edit here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRHelp.jsp

JSP for portlet help mode.

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>
```

```
<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRHelp.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Help here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<divid="MyJSRHelpDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSRHelpDiv = 'MyJSRHelpDiv-<portlet:namespace/>';
</script>
```

MyJSRView_listperson.jsp

JSP code for portlet view mode.

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
var portletId = 'portlet-container-<c:out value="${portlet}"/>';
var pidPrefix = "portlet-container-";
var portletWindowState = "<c:out value='${portletWindowState}'/>";
```

```
var portletMode = "<c:out value='${portletMode}'/>";
var defaultRecordSize = <c:out value='${defaultRecordSize}'/>;
var nsappPersonViewDiv = 'nsappPersonViewDiv-<portlet:namespace/>';
var pagingUrl = '<portlet2:resourceURL escapeXml="false" />';
var connectionTimeOut = <c:out value='${connectionTimeOut}'/>;
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRView.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Edit here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
<!--PortletMode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/>-->
</c:if>

<!-Un-comment this to get User Info from the Portlet Controller that sets user info in
model
Logged In User (<c:out value="${userName}" escapeXml="false"/>): <c:out
value="${firstName}" escapeXml="false"/> ,  <c:out value="${lastName}"
escapeXml="false"/> , <c:out value="${PersonId}"
escapeXml="false"/> , <c:out value="${HomeOUId}" escapeXml="false"/>
-->

<!-Write declare divs to display user info -->
<div>Logged In User (<div id="MyJSRLoginNameDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>) : <div id="MyJSRFirstNameDiv"
style="text-align:right;display:inline;width:100%;"><span></span></div> , 
<divid="MyJSRLastNameDiv" style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>
</div>
<div>HomeOUId : <div id="MyJSRHomeOUDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div> , 
PersonId : <div id="MyJSRPortletPersonIDDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span>
</div>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
```

```
var createNewPersonActionUrl = '<portlet:renderURL><portlet:param name="formAction"
value="createNewPerson" /></portlet:renderURL>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRView_updateperson.jsp

Example JSP code to demonstrate update user operation.

```
<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<%String contextPath = request.getContextPath(); %>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
   }
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
   }
</script>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRCreatePersonView.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet View here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
```

```
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/>-->
</c:if>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>
<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var viewPersonUrl = '<portlet:renderURL></portlet:renderURL>';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>
```

MyJSRController.java

The steps for developing java portlet controllers typically include:

Step 1 Write handler code for the three portlet modes—View, Edit, and Help.

Step 2 Write handler code for the three portlet views—Normal, Minimized, and Maximized.

Step 3 For JSR portlets that process/display Service Catalog entities, the nsAPI client can be used to invoke the related REST APIs in the portlet controller.

- **a.** Get reference to nsAPI client API.
- **b.** Call nsAPI client to get a list of the instances for the required Service Catalog entity.
- **c.** Optionally get the details for the currently logged-in user; for example, Person ID, First Name, Last Name.

Step 4 Render the instances in a grid or other format (this example also demonstrates how to do paging in nsAPI for a Ext JS Grid).

```
package com.myjsr;

importjava.io.IOException;
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.List;
import java.util.Map;
importjava.util.Properties;

importjavax.portlet.RenderRequest;
importjavax.portlet.RenderResponse;
importjavax.portlet.ResourceRequest;
importjavax.portlet.ResourceResponse;

importnet.sf.json.JSON;
importnet.sf.json.JSONSerializer;
importjavax.portlet.ActionRequest;
importjavax.servlet.http.HttpSession;
importjavax.portlet.PortletURL;
importorg.apache.commons.collections.map.MultiValueMap;
importorg.apache.commons.lang.StringEscapeUtils;
importorg.apache.commons.lang.StringUtils;
importorg.springframework.ui.Model;
importorg.springframework.web.portlet.ModelAndView;
```

```
importorg.springframework.web.bind.annotation.ModelAttribute;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.portlet.bind.annotation.ResourceMapping;
importorg.springframework.web.bind.annotation.RequestParam;

importcom.newscale.comps.conf.domain.AppParamUtil;
importcom.newscale.nsapi.directory.person.Person;
importcom.newscale.nsapi.directory.person.PersonList;
importcom.newscale.nsapiclient.NSApiClient;
importcom.newscale.nsapiclient.NSApiClientConstants;
importcom.newscale.nsapiclient.NSApiClientFactory;
importcom.newscale.portlets.GenericNewScaleSpringPortletBase;


/**
*MyJSRController
*/
publicclass MyJSRController extends GenericNewScaleSpringPortletBase  {
privatestaticfinal String configPropsFile = "jsrportlet.properties";
privatestaticfinal String viewPageList = "MyJSRView_listperson";
privatestaticfinal String viewPageUpdate = "MyJSRView_updateperson";
privatestaticfinal String editPage = "MyJSREdit";
privatestaticfinal String helpPage = "MyJSRHelp";
private NSApiClient nsApiClient = getNSApiClient();

public NSApiClient getNsApiClient() {
return nsApiClient;
  }

public void setNsApiClient(NSApiClient nsApiClient) {
this.nsApiClient = nsApiClient;
  }

public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewNormal(request, response, model);
    getLoginUsername(request , model);
    } catch(Exception e){
e.printStackTrace();
    }

return doView(request, response, model);
  }

public String viewMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewMinimized(request, response, model);
   } catch(Exception e){
e.printStackTrace();
   }
return viewPageList;
  }

public String viewMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.viewMaximized(request, response, model);
    getLoginUsername(request , model);
   } catch(Exception e){
e.printStackTrace();
   }

return doView(request, response, model);
  }
```

```
    private void getLoginUsername(RenderRequest request , Model model){
        Properties properties = getConfigProperties(configPropsFile);
    nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());
             // Get Currently Logged-in user from nsAPI client
         Person persons  = nsApiClient.getDirectory().getCurrentUser();

// Set user info into model so that JSP can access it
model.addAttribute("PersonID",  persons.getPersonId());
    model.addAttribute("HomeOUId",  persons.getHomeOrganizationalUnitId());
    model.addAttribute("firstName",  persons.getFirstName());
    model.addAttribute("lastName",  persons.getLastName());
    model.addAttribute("userName",  persons.getLogin());
  }

private String doView(RenderRequest request, RenderResponse response, Model model) {
try {
    Properties properties = getConfigProperties(configPropsFile);
nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());

int defaultRecordSize = AppParamUtil.getInstance().getMaxMaxPagingSizeInNSApi();
model.addAttribute("defaultRecordSize", "" + defaultRecordSize);
int connectionTimeOut = 0;
if (AppParamUtil.getInstance().isParamExists((AppParamUtil.SESSION_TIMEOUT))) {
connectionTimeOut =
AppParamUtil.getInstance().getIntegerParam(AppParamUtil.SESSION_TIMEOUT);
    }
if (connectionTimeOut < 1) {
connectionTimeOut = 20;
    }
model.addAttribute("connectionTimeOut", "" + connectionTimeOut * 1000 * 60);

    String formAction = request.getParameter("formAction");
    String personIdStr = request.getParameter("personId");

if (null != formAction && formAction.equals("createNewPerson") ) {
    showAddPersonPage(request, response, model);
    return viewPageUpdate;
    } elseif(null != personIdStr){
    editPerson(request, response, model, new Integer(personIdStr).intValue());
    return viewPageUpdate;
    } else {
    return viewPageList;
    }
     } catch (Exception e){
        e.printStackTrace();
     }
    returnnull;
  }

@RequestMapping("VIEW")
  @ResourceMapping
public ModelAndView doPagingOrSavePerson(ResourceRequest request, ResourceResponse
response, Person person)
throws Exception {
    String instanceName= getInstanceName(request.getWindowID());
 Save Button is clicked while adding person
if ("createNewPerson".equals(request.getParameter("formAction")) || null !=
request.getParameter("personId")) {
addPersonData(person ,request, response );
    } elseif (null != request.getParameter("start") &&null !=
request.getParameter("limit")) {  Paging
```

```
int startInt = Integer.parseInt(request.getParameter("start")) + 1; extjs sends 1 less
than what nsAPI wants
int limit = Integer.parseInt(request.getParameter("limit")) + 1; extjs sends 1 less than
what nsAPI wants
try {
if (request.getWindowState().equals(request.getWindowState().NORMAL)) {
doPagingInternal(request, response, instanceName, 1, startInt, limit);
        }
if (request.getWindowState().equals(request.getWindowState().MAXIMIZED)) {
doPagingInternal(request, response, instanceName, 2, startInt, limit);
        }
      } catch (Exception e) {
e.printStackTrace();
      }
    }
returnnull;
  }

private void doPagingInternal(ResourceRequest request, ResourceResponse response, String
portletInst,
int windowStateInt, int start, int limit) throws Exception {
    Map<String, Object> jsonMap = new HashMap<String, Object>();
    List recordList = newArrayList();
int totalCount = 1;
    String editPersonUrl = request.getParameter("editPersonUrl");
 MultiValueMap paramsmap = new MultiValueMap();
    paramsmap.put(NSApiClientConstants.QUERYPARAM_START_ROW, "" + start);
    paramsmap.put(NSApiClientConstants.QUERYPARAM_RECORD_SIZE, "" + limit);
    PersonList personList = nsApiClient.getDirectory().getPeople(paramsmap);
if(personList.getPeople() != null) {
for(Iterator iterator = personList.getPeople().iterator(); iterator.hasNext();) {
        Person portalPerson = (Person) iterator.next();
portalPerson.setPersonURL(StringEscapeUtils.escapeXml(portalPerson.getPersonURL()));
        PortletURL editPersonURL = response.createRenderURL();
editPersonURL.setParameter("personId", "" + portalPerson.getPersonId());
        String firstNameUrl = "<a href='" + editPersonURL.toString() + "'>" +
portalPerson.getFirstName() + "</a>";
        String lastNameUrl = "<a href='" + editPersonURL.toString() + "'>" +
portalPerson.getLastName()  + "</a>";
portalPerson.setFirstName(firstNameUrl);
portalPerson.setLastName(lastNameUrl);
recordList.add(portalPerson);
      }
jsonMap.put("success", "true");
jsonMap.put("results", personList.getTotalCount());
jsonMap.put("rows", recordList);
      JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    }
  }


private ModelAndView addPersonData(@ModelAttribute("personData") Person person,
      ResourceRequest request, ResourceResponse response) throws Exception {
 Add Person from Form Data in Request
    Map jsonMap = newHashMap();
try {
Person  Updateperson = nsApiClient.getDirectory().updatePerson(person);
jsonMap.put("success", "true");
jsonMap.put("successMsg", "Person Added/Updated Successfully");
```

```
jsonMap.put("rows", Updateperson);
      JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    } catch (Exception e) {
e.printStackTrace();
jsonMap.put("success", "false");
jsonMap.put("errorMsg", "Person Add/Update Failed : " + e.getMessage());

      JSON json2 = (JSON) JSONSerializer.toJSON(jsonMap);
      String jsonStr2 = json2.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr2.getBytes());
response.getPortletOutputStream().flush();
    }

returnnull;
  }

private void editPerson(RenderRequest request, RenderResponse response, Model model, int
personId) {
int person = personId;
    Map<String, Object> jsonMap = new HashMap<String, Object>();
try {
Person  persons = nsApiClient.getDirectory().getPersonById(person);
persons.setPersonURL(StringEscapeUtils.escapeXml(persons.getPersonURL()));
     JSON json = (JSON) JSONSerializer.toJSON(persons);
     String jsonStr = json.toString();
model.addAttribute("PersonData", jsonStr);
    } catch (Exception e) {
e.printStackTrace();
    }
  }

private String showAddPersonPage(RenderRequest request, RenderResponse response, Model
model) {
    Person DummyPerson = newPerson();
DummyPerson.setPersonURL(StringEscapeUtils.escapeXml(DummyPerson.getPersonURL()));
    JSON json1 = (JSON) JSONSerializer.toJSON(DummyPerson);
    String jsonStr1 = json1.toString();
model.addAttribute("PersonData", jsonStr1);

return viewPageUpdate;
  }

public String editNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.editNormal(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return editPage;
  }

public String editMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.editMinimized(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
```

```
    return editPage;
      }

public String editMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.editMaximized(request, response, model);
   } catch(Exception e){
e.printStackTrace();
   }
return editPage;
  }

public String helpNormal(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpNormal(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

public String helpMinimized(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpMinimized(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

public String helpMaximized(RenderRequest request, RenderResponse response, Model model) {
try {
super.helpMaximized(request, response, model);
    } catch(Exception e){
e.printStackTrace();
    }
return helpPage;
  }

private NSApiClient getNSApiClient() {
return NSApiClientFactory.getInstance();
  }
}
```

MyJSRApplicationContext.xml

Spring application context XML for the portlet.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beansxmlns="http:www.springframework.org/schema/beans"
xmlns:xsi="http:www.w3.org/2001/XMLSchema-instance"xmlns:p="http:www.springframework.org/s
chema/p"
xmlns:context="http:www.springframework.org/schema/context"
xsi:schemaLocation="
    http:www.springframework.org/schema/beans
http:www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http:www.springframework.org/schema/context
http:www.springframework.org/schema/context/spring-context-2.5.xsd">

<beanid="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
```

```
<propertyname="cache"value="true"/>
<propertyname="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
<propertyname="prefix"value="/WEB-INF/jsp/"/>
<propertyname="suffix"value=".jsp"/>
</bean>
<context:annotation-config/>
<bean
class="org.springframework.web.portlet.mvc.annotation.DefaultAnnotationHandlerMapping">
<propertyname="interceptors">
<bean
class="org.springframework.web.portlet.handler.ParameterMappingInterceptor"/>
</property>
</bean>
<beanid="MyJSRController"class="com.myjsr.MyJSRController">
</bean>
</beans>
```

jsrportlet.properties

URL of the MyJSRerver for the use by nsAPI.

Note In a clustered environment, if the portlet references the Service Catalog application URL, then specify the URL as “http://localhost:<port>/RequestCenter” where <port> is the port number used by each node in the cluster. In other words, do not specify the URL as “http:<host_name>/RequestCenter” where <host_name> is the computer name of the web server or one of the hosts within the cluster.

```
#(Port number and host has to changed as per the application server).
BASE_URL=http:localhost:8088/RequestCenter
```

Log4j.properties

```
log4j.rootCategory=INFO,CONSOLE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ABSOLUTE}%-5p[%c{1}:%L]%m%n
```

jboss-deployment-structure.xml

```
<jboss-deployment-structure>
  <deployment>
     <dependencies>
          <module name="javax.portlet" slot="main" export="true"/>
          <module name="org.apache.pluto.container.om" export="true"/>
          <module name="org.apache.pluto.container.driver" export="true"/>
          <module name="org.apache.pluto.tags" export="true"/>
      </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Compiling JSR Portlet Controller

Include the dependent libraries in the classpath when compiling the portlet controller. For a complete list of libraries, see Portlet Structure and Packaging.

Deploying Portlets

The deployment procedures vary with the application server used. As a general note, a JSR portlet can be deployed like any typical web application through the application server administrator console.

For detailed instructions on how to deploy the JSR portlet, and how to use the portlet on a portal page after deployment, see the *Cisco Prime Service Catalog Designer Guide*.

CHAPTER 8

Integrating with External Directories

Overview

Service Catalog Directory Integration simplifies security administration and enhances user convenience and productivity by implementing centralized user authentication and synchronization with an enterprise directory.

Service Catalog enables customers to integrate with an external directory (typically using the LDAP protocol) for user information synchronization. This synchronization is invoked whenever a user is selected for Order-on-behalf (OOB) or during Person Lookup.

Single Sign-On (SSO) integration enables centralized user authentication, eliminating the need for a separate login mechanism. When the SSO event is enabled, users who are already logged in to an enterprise portal with which Service Catalog has been integrated do not have to login again. Service Catalog relies on the SSO tool to protect all Service Catalog URLs and to perform authentication. Service Catalog requires that the SSO tool provide person identification information for each successful authentication to a Service Catalog URL via the HTTP header or cgi header. Once a person has been authenticated, their information can be synchronized to the application database.

If SSO is not enabled, then the Service Catalog login screen is presented to all users so they can provide a valid username and password combination. By default, these credentials are authenticated against the internal database. Alternatively, Directory Integration could be configured to authenticate to an external system (generally an LDAP directory). Users who wish to access Service Catalog must be present in this source for successful authentication.

The Directory Integration Framework provides the above capabilities for many frequently deployed SSO and directory server products through configuration options available in the Administration module. The framework also includes an application programming interface (API) which can supplement predefined configuration capabilities. The API allows programmers to access additional SSO portals and directory servers, as well as to alter or supplement default behavior for synchronizing user information between Service Catalog and the external directory.

This chapter describes how to configure directory integration for Service Catalog using the Administration module. It also describes the set of public APIs and interfaces available for customizing the integration options available, best practices for compiling and deploying custom code, and steps to configure the custom code using the Administration module.

Prerequisites

Configuring directory integration requires the following:

- A working Service Catalog installation.
- Directory server installed and directories populated with corporate data. Directory entries for all potential users must contain non-null values for all attributes that are mapped to fields required for integration operation, as explained in the Defining Mappings.
- An SSO system that is responsible for the authenticating and authorizing access to Service Catalog, if Single Sign-On (SSO) is to be used.
- A user login with a role that includes the capability to "Manage Global Settings". This capability is automatically included in the "Site Administrator" role and assigned to the "admin" user, but may be assigned to other roles or users as appropriate, using the Roles option in the Administration module.

Note Access to an LDAP browser is strongly recommended.

Prerequisites for Configuring Directory Integration

To configure directory integration, you need to have handy information about the current implementation of SSO (if used) and directory servers at your company, and to document the requirements for integrating these systems with Service Catalog. This section provides a set of worksheets for collecting this information.

These worksheets should help you collect the information required to configure directory/SSO integration, and to identify issues which need to be resolved before the integration can be implemented. This, in turn, can help in estimating the amount of development and testing time required for the directory integration.

Defining Datasources

Service Catalog defines a "datasource" for each directory which stores personnel and organization data to be accessed. The datasource definition includes all information required to connect to the external directory and extracting information from that directory.

You will need to define one datasource for each external directory. For example, different development and production directories may be used. In addition, Service Catalog supports LDAP directory referrals—a datasource needs to be defined for each directory in the referral chain.

Table 8-1 Datasource Definition Table

Setting	Value	Description
Datasource Name		The name of the datasource. Do not use spaces or special characters.
Datasource Description		Optional description of the datasource.
Protocol	• LDAP	LDAP is the only supported protocol at this time. If directory information is stored using another protocol, you need to create custom code to access this information.

Table 8-1 Datasource Definition Table

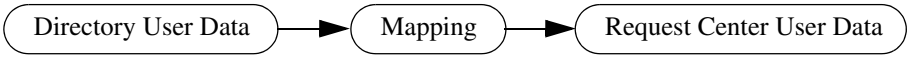
Setting	Value	Description
Server Product	• Sun™ ONE Directory • Microsoft® Active Directory® • IBM® Tivoli®	Choose the directory server product you are using. If the server is not currently supported, you will need to create custom code to access the server and extract directory information.
Authentication Method	• Simple • Anonymous • SASL	**Simple** means plain text user/password. **SASL** (Simple Authentication and Security Layer) is also available, but SASL only works with Sun ONE Directory Server.
Connection Mechanism	• SSL • Non SSL	Only needed if you choose **Simple** or **SASL** as the authentication method. Choose **SSL** to send encrypted information.
BindDN		Bind distinguished name field. BindDN is used to connect to the LDAP server when Service Catalog performs a directory operation. You may want to create a service account for this purpose. When this datasource is used in an External Authentication step, you will provide an EUA Bind DN in the Options area to override this value. For more details, see the External User Authentication (EUA) Operation.
Password		Required if you choose **Simple** or **SASL** authentication; the password for the user specified as the Bind DN. If the account uses password aging, you will need to update this password periodically.
Host		Fully qualified domain name or IP address of the LDAP directory server.
Port Number		Port number to connect to the directory server. Port Number 389 is typically used for non-SSL access.
User BaseDN		The directory from which to start searching for persons in the directory; since corporate directories may include many branches, specifying a base DN for the user data will optimize directory searches.
AuthzID		Required if you choose **SASL** authentication.
Optional Filter		This filter are added to other search filters you use, and it can be used to effectively change the search results. The filter expression must be enclosed in parentheses; for example, the filter: (&(!(msExchHide=true)(ISC-GID=*))) will return only those entries for which the msExchHide attribute is true and for which an ISC-GID attribute is defined.

Table 8-1 Datasource Definition Table

Setting	Value	Description
Security Certificate Name		Required if you choose SSL as the connection mechanism. Do not use spaces or special characters in the certificate alias name. Ensure that you have the certificate data ready to enter.
Referral Datasource		You can add one or more datasources as a referral. When a datasource search does not return results, the system searches the referral datasources as well. Referrals are supported for searches only, not binding. *You cannot set up cyclic referrals. Cyclic referrals are those where one datasource has another datasource as a referral, while that datasource has the original as a referral. For example, datasource A has datasource B as a referral, while datasource B has datasource A as a referral.*

Defining Mappings

A "mapping" is a set of rules that gives instructions for how data is to be transferred from the external directory to Service Catalog. It maps between source attributes in the directory and target fields in the Service Catalog database. The rules are used to transfer data from the directory to the designated target field when the Service Catalog database is synchronized with the directory.



The same mapping can be applied to multiple directories (datasources).

A mapping includes the user/person's profile along with all related entities: addresses, contacts, locations, one or more group associations, one or more organizational unit (OU) associations, and one or more RBAC (role-based access control) role associations.

A person profile includes seven mandatory fields, listed in the "Mandatory" section of the Mapping Worksheet below. Directory records which do not provide a value for any of these fields cannot be imported. Other fields which are part of the person profile can also be mapped. For more information, see People section, in Organization Designer chapter of Cisco Prime Service Catalog Administration and Operations Guide.

Most of the fields on the person profile are used to drive the Service Catalog functionality, and the mapping should ensure that mapped attributes provide a source value appropriate for the field; that is, do not try to overload these fields with more information than would be suggested by the field name, or with information that does not match the field name.

Service Catalog also includes fields which provide an extension to the standard personnel data. These fields are denoted as "Extension" on the following table and appear on the Extensions page of the Person information in Organization Designer. Some of the most frequently required extended fields have been assigned meaningful names (such as Company Code and Division), but others have the names Custom 1 through Custom 10, and are intended to be freely used, with no preconceived semantics. If you have additional personnel information in the LDAP directory that needs to be exposed in Service Catalog, map the attributes containing that information to one of the personnel extended fields.

The “Directory Attribute” column in the worksheet below should be filled in for all Person profile fields for which the directory must supply data. The Attribute should be one of the following:

- The directory attribute name or names, if two or more attributes can be concatenated (with optional literals) to form the value for the field.
- “Custom mapping”, following by a number or description. All custom mappings should be explained in detail in the Custom Mappings or noted briefly in the “Comments” column. Custom mappings may assign the result of a regular expression to the attribute, or may be implemented via a module of custom Java code. Details for implementing these mappings are given in the Configuring Mappings.

Mandatory Mappings

Table 8-2 Mandatory Mapping Field Description Table

Field	Comments
First Name	
Last Name	
Login ID	Unique identifier to be used as the person's login name for Service Catalog.
Person Identification	The Person Identification should map to an attribute that provides a unique value for each person. For example, specify an attribute that contains the employee id or social security number. Ideally, the same attribute should map to both the Login ID and the Person Identification; at a minimum, the two should be tightly coupled.
Email Address	
Home Organizational Unit	The Home OU is always a business unit, not a service team.
Password	Directory servers will typically not return a password. However you can use this field to create, for example, default passwords for new users.

Optional Mappings

Table 8-3 Optional Mapping Field Description Table

Field	Comments
Title	
Social Security Number	
Birthdate	The return type of the LDAP attribute being mapped must return a long. Service Catalog does not support other formats.
Hire Date	The return type of the LDAP attribute being mapped must return a long. Service Catalog does not support other formats.

Table 8-3 Optional Mapping Field Description Table

Field	Comments
Timezone ID	The mapped attribute must return a value in one of the following formats: • GMT+- Offset • Country/Language As of the March 2008, the familiar three-letter time zone designations (for example, "EST" for Eastern Standard Time) should not be used. For a list of supported values for the above formats, see the Supported Time Zones. If the return value does not match one of the valid formats, Service Catalog uses PST as the default time zone.
Locale ID	The mapped attribute must return a value in the form: language_COUNTRY where language is a two-letter language code and the country is a two-letter country code. Directory integration supports the following locales: • en_US (United State English) • de_DE (German) • es_ES (Spanish) • fr_FR (French) • ja_JP (Japanese) • zh_CN (Mainland Chinese) • zh_TW (Taiwanese Chinese) • Korean
Employee Code	
Supervisor	This field represents the identification of manager. For more details, see the Import Manager Operation.
Notes	
Company Street 1	
Company Street 2	
Company City	
Company State	
Company Postal Code	
Company Country	
Building	
Level	
Office	
Cubicle	
Personal Street 1	

Table 8-3 Optional Mapping Field Description Table

Field	Comments
Personal Street 2	
Personal City	
Personal State	
Personal Postal Code	
Personal Country	
Work Phone	
Home phone	
Fax	
Mobile Phone	
Pager	
Other	
Main Phone	
Primary Phone	
Primary Fax	
Sales Phone	
Support Phone	
Billing Phone	
Other Contact Information	
Company Code	Extension
Division	Extension
Business Unit	Extension
Department Number	Extension
Cost Center	Extension
Management Level	This should return a number. When used with the Import Manager event, Management Level is expected to be in increasing order according to the hierarchy. For example, if there are two designations, Junior Engineer and Senior Engineer, Management Level returned for Junior Engineer should be less than the Management Level of Senior Engineer.
Region	Extension
Employee Type	Extension
Location Code	Extension
Custom 1	Extension
Custom 2	Extension
Custom 3	Extension
Custom 4	Extension

Table 8-3 Optional Mapping Field Description Table

Field	Comments
Custom 5	Extension
Custom 6	Extension
Custom 7	Extension
Custom 8	Extension
Custom 9	Extension
Custom 10	Extension
Organizational Unit List	Use this mapping to associate the person with one or more Organizational Units. The mapping may return multiple values. For this field Service Catalog uses all values returned by multivalued LDAP attributes. Input for this field should be in one of the following formats: • Name of the Java class that returns the multiple values as defined in Directory Integration API documentation. • One or more simple mappings separated by "::". For example, ou::departmentNumber. • One or more expression mappings separated by "::", as in: expr:#memberOf#=(cn=(.*),cn=Users,dc=celosis,dc=com)?($1):Default:: expr:#memberOf#=(cn=(.*),ou=Users,dc=celosis,dc=com)?($1):Default
Group List	Similar to Organizational Unit List.
Role List	Similar to Organizational Unit List. The returned roles may be either system- or user-defined. For system-defined roles, the names must be exactly as they appear in a browser with language US English. Other languages are not supported. For example, "My Services Executive" should be returned to associate a user with this role. For user-defined roles, the name must exactly match the user input in user language while creating the role.

Custom Mappings

You may use the worksheet below to document requirements for custom mappings.

Table 8-4 Custom Mapping Field Description Table

Field	Type	Requirements
	Expression Java	
	Expression Java	

Defining Integration Events, Operations and Steps

Integration events are the interfaces between Service Catalog and an external directory or SSO program—the only times in the use of Service Catalog that the external program or directory is accessed. These events consist of a series of operations which are executed in sequence.

Events

Service Catalog supports four directory integration events:

- The "**Login**" event occurs when a user's credentials are validated and the user connects to Service Catalog. This event occurs when a user initially starts a Service Catalog session. It also occurs if a session times out (the administrator-specified time-out period expires) and the user must reconnect.
- A "**Person Lookup**" event occurs every time user information must be retrieved. There are actually three types of Person Lookup events:
  - **Person Lookup for Order on Behalf**: A user requests a service on behalf of another person, and must choose the person who is the customer for the service.
  - **Person Lookup for Service Form**: A service form includes a Person field, which allows the user to designate another person as part of the service data.
  - **Person Lookup for Authorization Delegate**: A user responsible for reviewing or authorizing service requests modifies his/her profile to designate another person as a temporary authorization delegate.

Operations

You can configure events to perform various types of operations. The operations are specified for each event in a series of steps, which determines the sequence in which each operation in invoked.

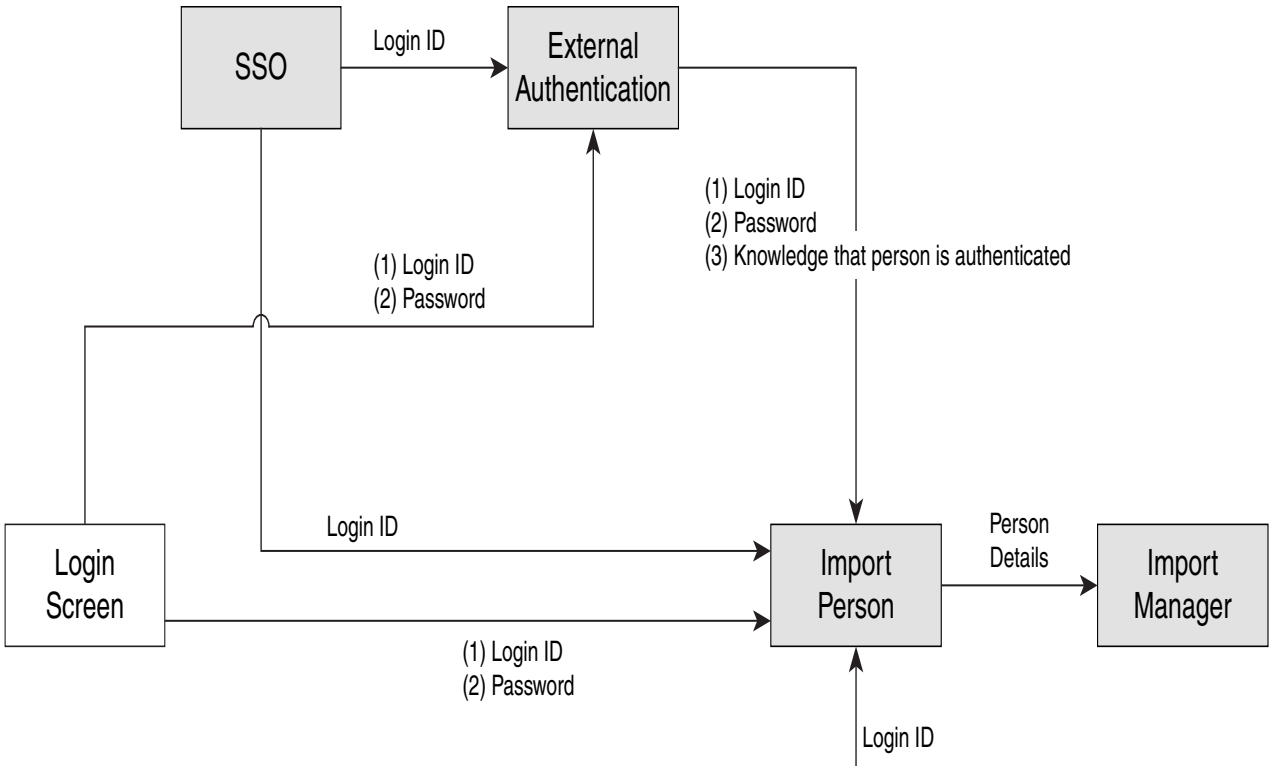
The directory framework includes the following operations:

- **Single Sign-On** (SSO) is always the first step in the Login event. The SSO operation identifies the login name of the user.
- **External Authentication** can occur after the SSO operation or, if an SSO operation is not used, after the default Login screen. External Authentication uses the login name and password of the user and authenticates them against an external datasource.
- **Person Search** is triggered when the user invokes a search on a datasource. Person Search uses the First Name and Last Name of the user to provide a list of matched items.
- **Import Person** can occur after External Authentication or after SSO, or after a person is chosen in the Person Search dialog box. Import Person uses the login name of the person searched or logging in to query a datasource and import the person into the database.
- **Import Manager** can only occur after Import Person. The Import Manager operation will use the imported person information to import the managers of this person.

Each operation can be customized via implementation of custom code interfaces.

Trigger Order below shows the sequence in which operations are triggered.

Figure 8-1 Trigger Order



Login Event

If a directory integration login event is not configured, the default behavior is to present the login screen and validate the credentials entered (user name and password) against the contents of the application database.

If the directory integration event is enabled, the Login event may be configured with either one of the following operations as its first step:

- Single Sign-On: In a corporate environment where all users are preauthenticated using SSO vendors, automatically extract the login id of the user from request headers or CGI headers and allow transparent login, bypassing the application login screen.
- External User Authentication: Present the application login screen and validate the credentials entered against the specified external directory. External User Authentication may also follow an SSO operation.
- Mixed Mode Authentication: Avoids NTLM authentication via application server port. When DB Credentials(both Username and Password) are passed via the Application server Port and NTLM Authentication is enabled on the web Server, Database Authentication triggers and database user owns the session.

  If EUA is enabled and both Username and Password is passed , an LDAP authentication is triggered. The Password provided must be the LDAP Password and LDAP user owns the session..

Once the user credentials have been validated, the Login event may include additional operations to synchronize user data between the external datasource and Service Catalog:

- The “Import Person” operation may be the next step. This operation imports the profile of the authenticated person selected to Service Catalog, synchronizing the data.

- The “Import Manager” operation may follow the “Import Person” step. This operation retrieves information on the managers of the selected person from the external directory and updates the Service Catalog database with that information.

Single Sign-On Operation

Integration with Single Sign-On (SSO) solutions can use one of the following two mechanisms/protocols:

1. Active Directory Services (ADS)/NT LAN Manager (NTLM)-based authenticated user
   - The third-party IM/AM/SSO product is not needed to log into Service Catalog.
   - The logged in user credentials from any POSIX-compliant OS are returned by the browser to Service Catalog.
   - This is also called integration through CGI Headers for SSO.
2. HTTP Request Headers
   - This is for non-ADS/NTLM integration.
   - It requires the third-party IM/AM/SSO product to log into Service Catalog using RequestHeaders in the http protocol.

For customers who plan to use SSO for both Portlet and Directory Integration, only HTTP Header SSO is supported. Custom SSO plug-ins within the Directory Integration framework are not supported.

Setting	Value	Description
Single Sign-On Type	HTTP Header Remote User	Specify the type corresponding to your SSO solution. Be sure to verify that login ID information is accessible. Check HTTP Header to use http Request Header protocol. Check Remote User to use ADS/NTLM protocol.
Login ID Mapping		Login ID mapping for HTTP Sign-Ons should be the exact name of the Http Request Header that contains the login name of user signing in. Login ID mapping for ADS/NTLM Sign-Ons should be of the following format: #AnyDomain#\#LoginId# For example, celosis\#LoginId# limits users to the “celosis” domain, while #AnyDomain#\#LoginId# allows logins across multiple domains. If multiple domains are in use, the LoginId must be unique across domains.
Redirect URL		The URL of the corporate portal from which users typically access Service Catalog products. Users are redirected to this URL if authentication fails, or when the application user session times out.

Administrative Bypass of SSO

It is sometimes necessary to allow some users to bypass the Single Sign-On and login directly to Service Catalog. This capability is typically required for:

- System administrators who need to investigate problems with Single Sign-On

- Testers who need to emulate the performance of multiple users in order to validate a service design and task plan

Service Catalog provides a mechanism for allowing users to access the login screen and enter a user name and password. The newscale.properties file (located within the RequestCenter.war) specifies a value for the "BackDoorURLParam"; for example:

```
BackDoorURLParam=AdminAccess
```

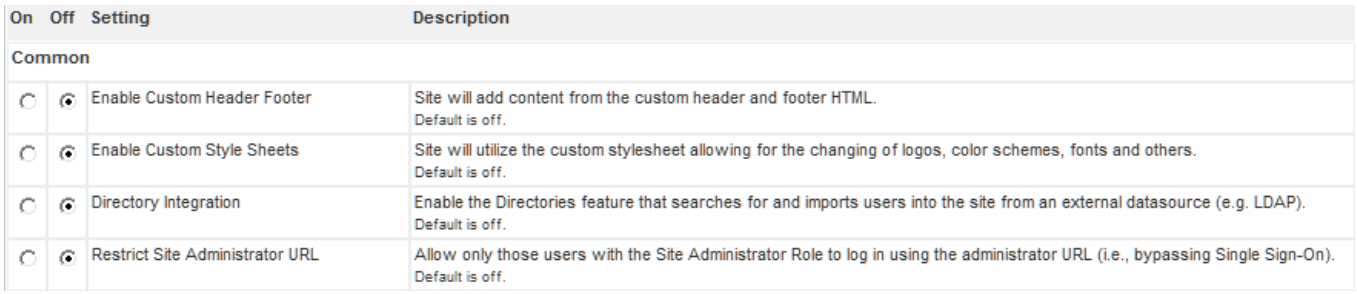
The URL used to access Service Catalog via the login screen must include a parameter. For the above value of the backDoorURLParam; for example, a sample URL might be:

```
http://prod.RequestCenter.com:<app_server_port>/RequestCenter?AdminAccess=true
```

<app_server_port> is the port number of application server.

It is the responsibility of the administrator to establish policies for aging out the value of the BackDoorURLParam according to corporate guidelines and for controlling administrative access to Service Catalog. Access via the administrative URL can be restricted to only those users who have the "Site Administrator" role via the corresponding Administration Setting:

Figure 8-2 Administration Setting



On	Off	Setting	Description
Common			
○	◉	Enable Custom Header Footer	Site will add content from the custom header and footer HTML. Default is off.
○	◉	Enable Custom Style Sheets	Site will utilize the custom stylesheet allowing for the changing of logos, color schemes, fonts and others. Default is off.
○	◉	Directory Integration	Enable the Directories feature that searches for and imports users into the site from an external datasource (e.g. LDAP). Default is off.
○	◉	Restrict Site Administrator URL	Allow only those users with the Site Administrator Role to log in using the administrator URL (i.e., bypassing Single Sign-On). Default is off.

The administrator must also ensure that the URL is directly accessible to users—access to the Service Catalog application may have previously been restricted to the SSO software via web server or network configuration parameters.

The Service Catalog service must be restarted for a change to this parameter to take effect.

External User Authentication (EUA) Operation

Use External Authentication to authenticate all Service Catalog users with a corporate directory. This way you do not have to worry about synchronizing user passwords.

External User Authentication must follow a login attempt—either via a configured Single Sign-On operation or through the application login screen. The LoginId retrieved from the previous operation is available to the EUA operation. However, validating this user in the external directory requires additional information, so that the BindDN can be located.

The EUABindDN setting allows the application to automatically extrapolate the bind DN of the user trying to sign on.

Table 8-5 EUABindDN setting

Setting	Description

External Authentication EUABindDN	EUABindDN is of the format:
	`Prefix#LoginId#Suffix.`
	Service Catalog will replace #LoginId# with the loginId of the user signing in from EUABindDN and use it as BindDN for authentication.
	For example, you can provide the EUABindDN like this:
	`uid=#LoginId#,OU=People,dc=example,dc=com`
	In such case if the user provides scarter as the login id in the logic screen during sign up, Service Catalog will use
	`uid=scarter,OU=People,dc=example,dc=com`
	to bind the user with external datasource.

Person Lookup Events

All Person Lookup events (Order on Behalf, Service Form, and Authorization Delegate) share the same behavior and configuration options.

If the directory integration event is not enabled, the Person Search window searches personnel information in the Service Catalog database. If a person is selected, their information is used. Personnel information is not updated.

If the directory integration event is enabled, the Person Lookup event may be configured with the following operations:

- The “Person Search” operation must be the first step. This operation retrieves personnel information from the external directory and displays it in the Person Search window. If the user selects a person, additional information on that person is retrieved, according to the mapping specified for the event, and supplied to the calling context.
- The “Import Person” operation may be the next step. This operation imports the profile of the person selected from the external directory to Service Catalog, synchronizing the data.
- The “Import Manager” operation may follow the “Import Person” step. This operation retrieves information on the managers of the selected person from the external directory and updates the Service Catalog database with that information.

Person Search Operation

Settings for the Person Search operation determine the appearance and behavior of the window that displays people meeting the search criteria.

In order for a person to be imported into Service Catalog, all mandatory fields must have a valid attribute mapping, which returns in a nonblank value. If any required values are missing, the default behavior is to exclude that person from the Search Results. The alternative is to include such people in the Search Results, and flag them as having incomplete information.

People with incomplete information cannot be chosen.

When configuring a Person Search operation

Table 8-6 Person Search Operation

Setting	Value	Comments
Search Selectivity	• Show People with Incomplete Information	Default is to exclude people with incomplete information from the Search Results window.
Sort By	• First Name • Last Name First Name • First Name Last Name • Last Name • No Sort	Default is to sort by Last Name.
Max Results		Default for the number of rows to display in the Search Results is 1000.

The * (Asterisk) Wildcard Character and Person Search

When configuring and testing a Person Search, you need to be aware of the use of the asterisk (*) as a wildcard character.

Transparent to the user, the system always appends an * to the end of the search string. Therefore, if a user enters john in the Last Name field, and clicks **Search**, the system returns all persons in the directory whose last name begins with the word john, such as “John”, “Johnson”, and “Johnston”.

A user may also explicitly enter the * character in the search string of the Search Person dialog box. Some examples of the usage for wildcard search are:

- Enter * in the **Last Name** field, and click **Search**. The system returns all persons in the directory.
- Enter **john*** in the **Last Name** field, and click **Search**. This is essentially the same as typing just **john** in the **Last Name** field. The system returns all persons in the directory whose last name begins with the word “john”.
- Enter ***john** in the **Last Name** field, and click **Search**. The system returns all persons whose last name contains the word “john,” including “John”, “McJohn”, and “Johnson”.
- Enter ***john*son** in the **Last Name** field, and click **Search**. The system returns all persons whose last name contains the word “john,” followed (not necessarily immediately) by the word “son.” These include “Johnson”, “Mcjohnson”, and “Upjohningson”.

Note The * is always treated as a wildcard character in the search string. Therefore, the user is NOT able to search for a value in the directory that contains the character *. Any other special characters may be used in the search string.

Configuring the Search Results Window

By default, the Search Results window in the Select Person Popup displays the person's first name followed by the last name. Additional fields can be added to the display by changing the Setting for the Person Popup available in the Administration module.

Import Person Operation

Import Person settings govern whether person information in the application is refreshed from current information about the selected person (when Import Person is used in a Person Search event) or the person who has logged in (when Import Person is used in a Login event).

Table 8-7 Import Person Operation

Setting	Value	Comments
Refresh	• Refresh Person Profile • Refresh Period (Hours)	Leave the refresh period blank or zero to refresh on every import—this will ensure that the Service Catalog database always reflects recent changes in the external directories. Alternatively, you can designate that a user's profile should be refreshed only after the designated period has passed since this last refresh.
Create Associations	• Do Not Create Organizational Unit • Do Not Create Group	Default is to create organizational units and groups if they do not exist. Roles cannot be created via directory integration and must exist before the person is imported.
Remove Existing Associations	• Organizational Unit • Group • Role	Default is not to remove existing organizational unit, group, or role associations.

Import Manager Operation

Service Catalog allows authorizations and reviews to be dynamically assigned. For example, a request with a dollar value greater than a specified threshold might need approval by the director of a particular department. Another request might need to be reviewed by the requestor's immediate superior.

To implement business rules like these, the managers of an employee who can request a service must also be present in the Service Catalog database. The Import Manager operation supports this requirement, importing manager (supervisor) data in conjunction with the employee's data.

To govern the behavior of the Import Manager operation:

- Identify the attribute in the employee's directory entry that is to designate his/her manager.
- For all employees, ensure that the designated attribute is populated with a value that uniquely identifies their manager. This is typically the login id or email address.
- In the mapping for the Supervisor field (listed in the Optional Person Data Mappings) specify the attribute in the employee data that holds the manager information. In the sample below, the managerEmail attribute is used.

Figure 8-3 Person Data

Person Data	Mapped Attributes
* First Name	givenName
* Last Name	sn
* Login ID	sAMAccountName
* Person Identification	sAMAccountName
* Email Address	mail
* Home Organizational Unit	department
* Password	sn
⊟ Optional Person Data Mappings	
Person Data	**Mapped Attributes**
Title	
Social Security Number	
Birthdate	
Hire Date	
Timezone ID	
Locale ID	
Employee Code	
Supervisor	managerEmail

362272

- In the Import Manager settings, specify as the "Key Field for Manager ID" the field in the manager's directory record whose value corresponds to the Supervisor attribute specified for the original person.

In one possible scenario, a single attribute exists in each person's directory record which uniquely identifies the person's supervisor. Assume, for example, that the person's directory record contains the manager's email ID within the attribute **manager_email**. No other manager information is present.

Table 8-8 Key Field for Manager ID

Solution	Supervisor	manager_email
	Key Field for Manager ID	email (the email attribute in the manager's directory record)

An alternative scenario may be that the directory record contains an attribute that is exactly the DN of the person's supervisor. Assume the name of this attribute is **manager**.

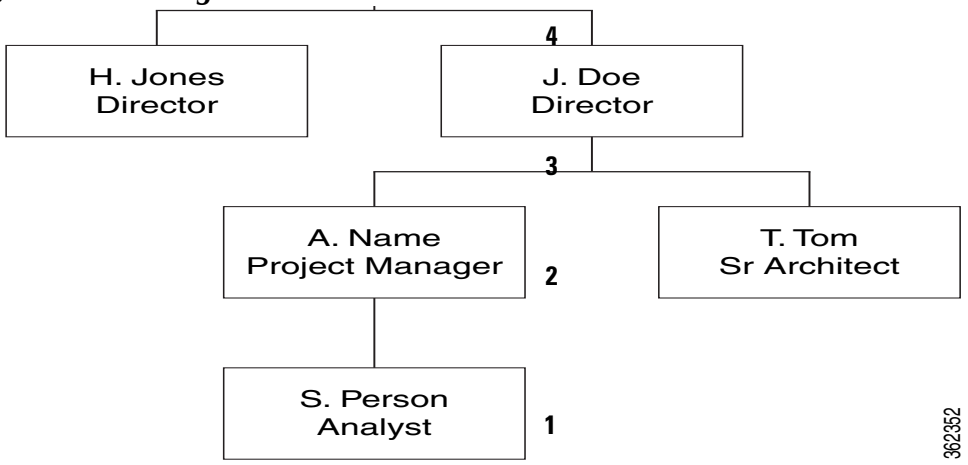
Table 8-9 Key Field for Manager ID

Solution	Supervisor	manager
	Key Field for Manager ID	dn (DN is a special attribute and is not prefixed before the search string)

Supervisory hierarchies may also need to be accommodated.

For example, consider this organizational chart:

Figure 8-4 ***Organizational Chart***



1	Level 1	**3**	Level 3
2	Level 2	**4**	Level 4

If requests were subject to an immediate supervisor's approval, a "relative" search is needed, going up the tree one level.

Alternatively, if certain requests were subject to, for example, a Director's approval, an "absolute" search is needed. People (managers) would be imported until the position of the current person was "Director". In the example above, in the case of S. Person, two additional people would be needed—her immediate manager, A. Name, and his manager, J. Doe, who is their Director. For T. Tom, only one import would be required.

If you are using an absolute search (import all managers with successively higher levels of authority until you find one with the specified position), you must assign numeric equivalents to the positions:

- Analyze the corporate hierarchy, assigning numeric equivalents to all management positions.
- Identify the attribute in the employee's directory entry that is to designate his/her management level. For example, perhaps an attribute named "paygrade" could be used.
- For all employees, ensure that the designated attribute is populated.
- In the mapping for the Management Level field (listed in the Optional Person Data Mappings) specify the attribute that holds this information.
- Enter the highest level of manager to be imported as the "Maximum Level" in the Import Manager settings.

You may configure a search terminator if you do not want to synchronize supervisors beyond a known value. You can specify multiple values in the format: #value1#, #value2# and so on.

For example, you may not want to import any supervisors who rank above a person with uid as "scarter." His Supervisor attribute is mapped to his email (scarter@email.com). In this case set the Search Terminator to #scarter@email.com#. The directory integration will stop supervisor synchronization as soon as a record is found with scarter@email.com as the supervisor.

Supervisor synchronization stops as soon as either limiting condition is met—Maximum Level or Search Terminator.

Table 8-10 Import Manager Operation

Setting	Value	Comments
Key Field for Manager ID		The directory attribute that uniquely identifies the employee's manager (supervisor).
Maximum Level		For absolute search this indicates the highest management level for a manager to be imported. For relative search this indicates the number of managers above the current employee that need to be imported.
Search Mode	• Absolute • Relative	
Search Terminator		The value or values that match the key field for managers that stop the search.
Refresh options	• Refresh Person Profile • Refresh Period (Hours)	Check the **Refresh Person Profile** check box to indicate that the manager's profile within Service Catalog is to be refreshed. If the Refresh Period is left blank, the profile is refreshed every time the Import Manager event takes place for the same person. If a number is provided, the manager's profile is refreshed only once within the specified period.
Associations	• Do Not Create Organizational Unit • Do Not Create Group	Identical to settings for Import Person.
Remove Existing Association	• Organizational Unit • Group • Role	Identical to settings for Import Person.

Custom Code Operations

Use a custom code operation to invoke routines not supported by the application. A custom code operation may replace or supplement Service Catalog operations.

Table 8-11 Custom Code Operations

Setting	Value	Comments
Custom Code Operation Type	• Single Sign-On • External Authentication • Import Person • Import Manager • Custom Code • Person Search	Use a Java class to provide the name of your mapping. For more details about the Java class see the Javadocs.

Configuring SSO With ADS

Create a file **jboss-web.xml** with the following contents in the directory **ServiceCatalogServer/RequestCenter.war/'WEB-INF'**.

Contents

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
<security-domain>mySSO</security-domain>
</jboss-web>
```

Modify web.xml

Note You must add this after </context-param>, and before the filters.

```
<login-config>
<auth-method>EXTERNAL</auth-method>
</login-config>
```

Modify standalone-full.xml, add a security domain:

Note You must search for *security–domains* and add the following contents below it.

```
<security-domain name="mySSO" cache-type="default">
      <authentication>
                    <login-module code="Client" flag="optional">
                    </login-module>
      </authentication>
</security-domain>
```

Note For Cluster, you should create **jboss-web.xml** and should modify the **web.xml** in the directory **C:\Install_dir\dist\RequestCenter.war**, for more information see Cisco Prime Service Catalog 11.1.1, section *Applying Patch or Customizations to the WildFly Cluster Setup.*

- You must search with keyword *security-domains* in the **domain.xml** (wildfly-8.2.0.Final\domain\configuration) and add the following contents below it.

```
<security-domain name="mySSO" cache-type="default">
      <authentication>
                    <login-module code="Client" flag="optional">
                    </login-module>
      </authentication>
</security-domain>
```

Configuring Directory LDAP Integration

Configuring directory integration involves using the Directories options of the Administration module. The basic process is to:

- **Enable directory integration**. Click the **Directory Integration** on the Administration module's Settings tab to enable directory integration.
- **Configure datasource information**. Use the Datasources area of the Administration module's Directories tab to configure datasources that connect to directory servers. Information such as the datasource name, description, protocol, server product, and authentication method is required.
- **Configure mapping**. Use the Mappings area of the Administration module's Directories tab to map application data to the directory server data. Mappings update the entire user/person's profile along with all related entities: addresses, contacts, locations, one or more group associations, one or more organizational unit (OU) associations, and one or more role associations.
- **Configure events**. Use the Events area of the Administration module's Directories tab to configure directory integration behavior. The Login and Person Lookup events can be configured to include operations such as Single Sign-On (SSO), End User Authentication (EUA), Import Person, Import Manager, and Person Search.
- If required, **configure custom code interfaces** for client customizations, including directory java class attribute mapping, directory server API, and Import Person, with its related entities.

Enabling Directory Integration

To enable directory integration:

Step 1 Log in using an account with administrative privileges and choose the **Administration** module.

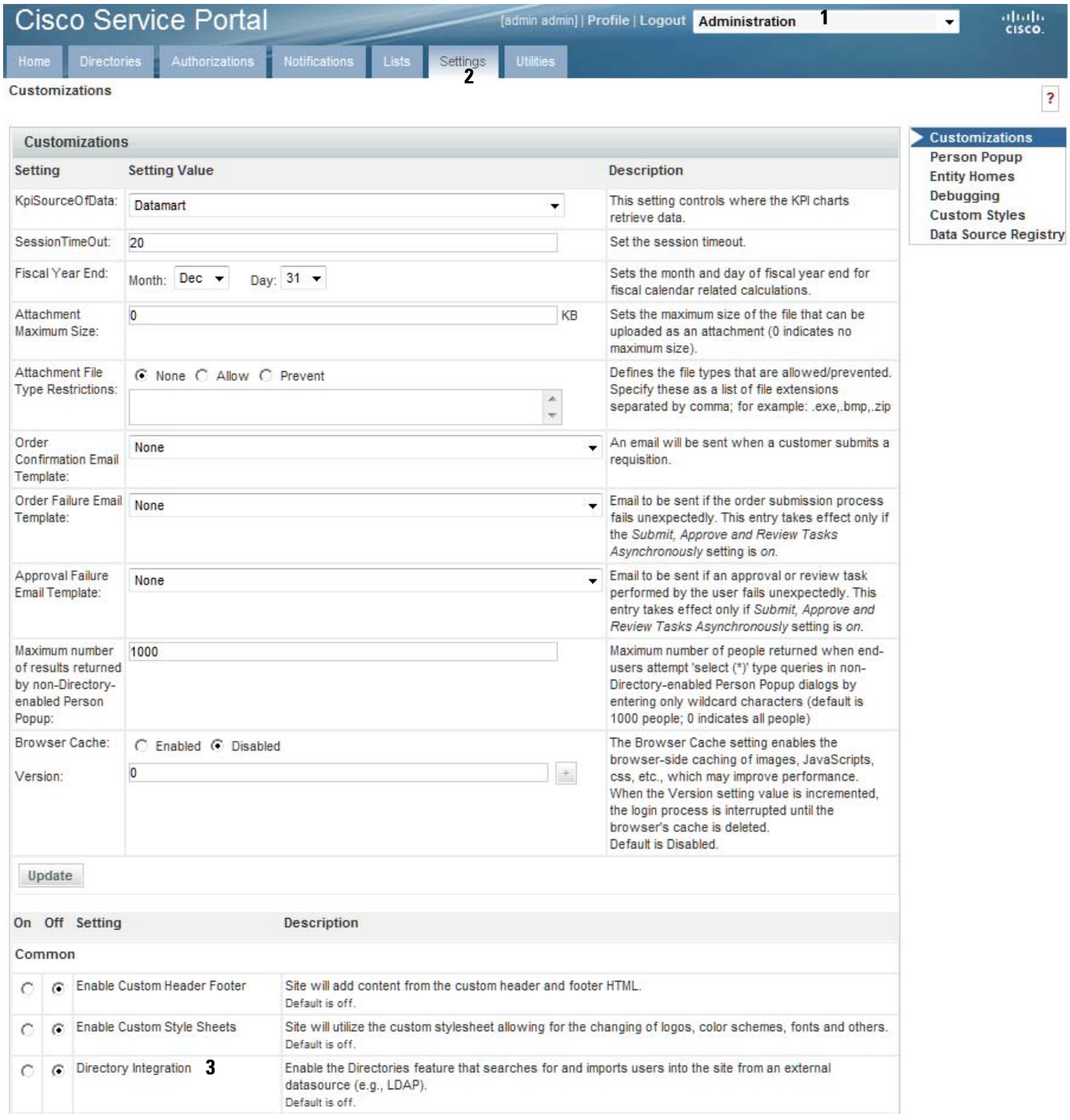
Step 2 Click the **Settings** tab.

Step 3 Next to Directory Integration, click **On**.

Step 4 On the bottom of the Customizations screen, click **Update**.

You have now enabled directory integration. (See Enabling Directory Integration.)

Figure 8-5 Enabling Directory Integration



1	Administration module	3	Directory Integration setting
2	Settings tab		

Configuring Directory Integration Settings

You use the Directories tab of the Administration module to configure many of the directory integration settings.

Figure 8-6 The Directory Integration Area

1	Administration module
2	Directories tab

To configure directory integration settings:

Step 1 Log in using an account with administrative privileges.

Step 2 From the drop-down menu, choose **Administration**.

Step 3 Click the **Directories** tab.

The Directory Integration page appears. These settings will be in effect once directory integration has been enabled.

Configuring Datasource Information

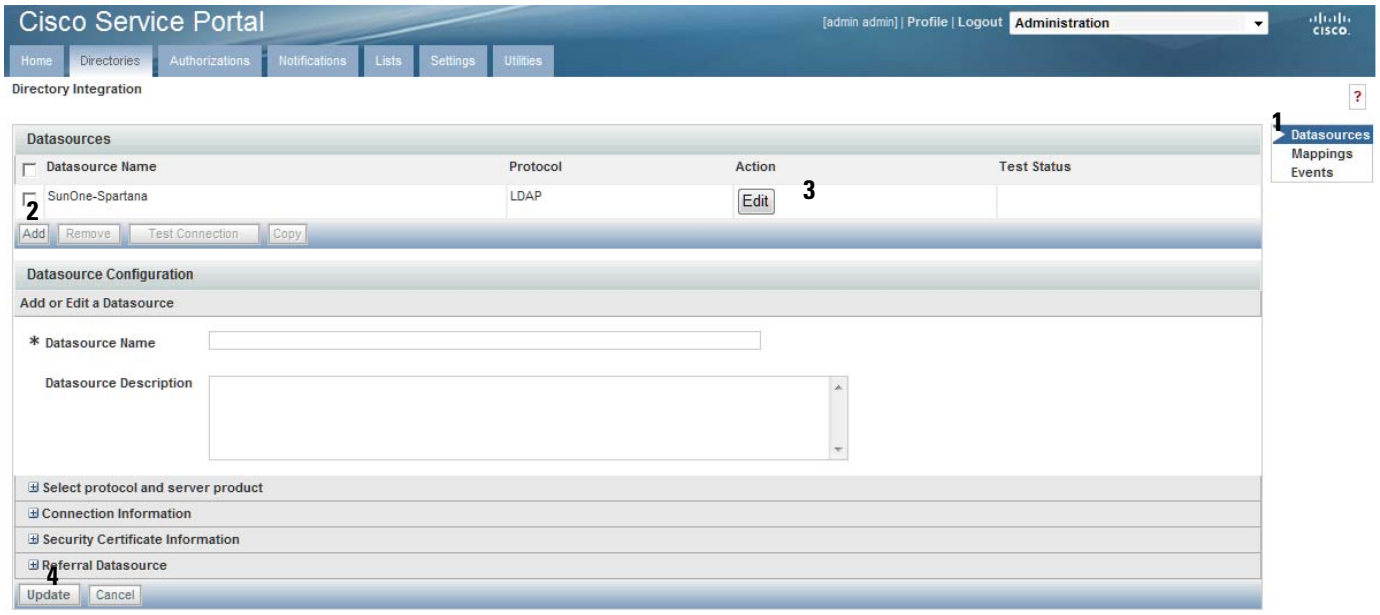
The following sections guide you through configuring datasource specific information. The tasks include:

- **Adding or editing a datasource** – You need to add a datasource to a new installation that does not yet have any datasources. If datasources exist, you may edit them.
- **Adding a server certificate for SSL connections** – You only need to do this if you choose SSL as the connection mechanism.
- **Adding referral datasources** – Only if desired.
- **Testing the connection** – You should always test the connection to prove connectivity.

Adding or Editing a Datasource

At least one datasource must be defined. To add a new datasource:

Figure 8-7 Adding or Editing a Datasource



1	Datasources option	**3**	Edit Datasource button
2	Add Datasource button	**4**	Update button

Step 1 Navigate to the Directory Integration page by choosing the **Administration** module and then clicking the **Directories** tab.

Step 2 In the page navigator, click the **Datasources** option, if not already selected.

Step 3 Click **Add**. To edit an existing datasource instead of adding a new datasource, click **Edit** next to the desired datasource in the list.

The Datasource Configuration area expands.

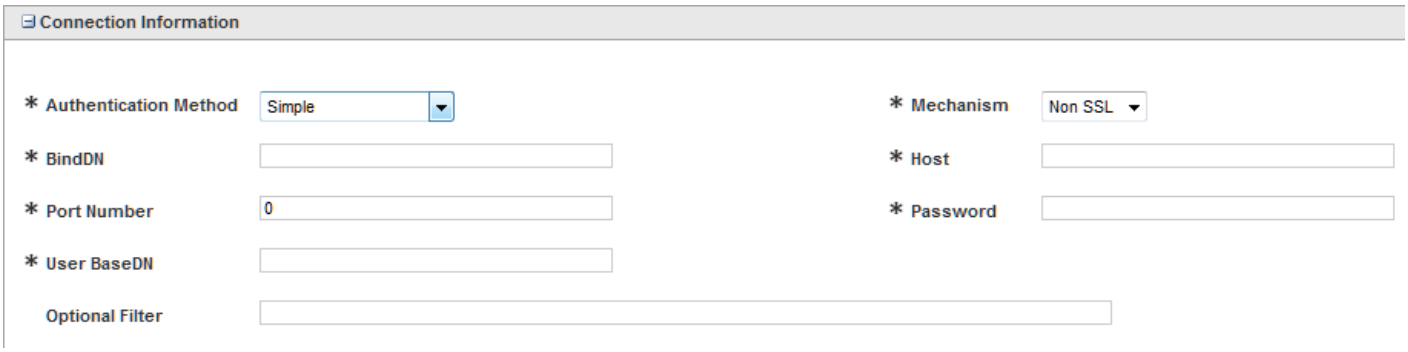
Step 4 Enter the Datasource Name, Datasource Description, and the desired settings. Click ⊞ to access all of the settings in the adjacent area. See the Datasource Worksheet for more information about these settings, or see the following sections.

Step 5 Click **Update**.

Configuring Connection Information

Specify the connection protocol and user credentials used to connect to the datasource.

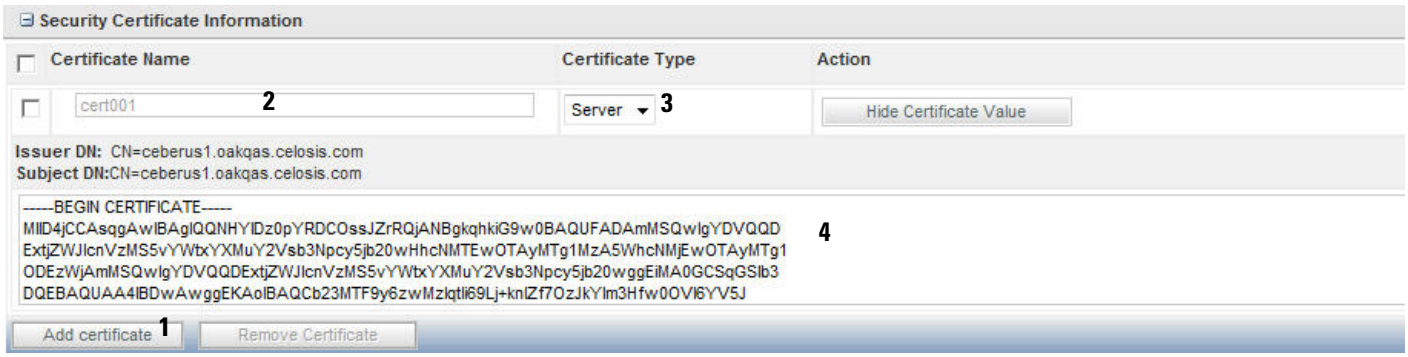
Figure 8-8 Configuring Connection Information



Configuring Certificates

If you chose SSL as the connection mechanism, you need to specify the certificates for the directory integration system.

Figure 8-9 Configuring Security Certificates



1	Add certificate button	**3**	Certificate Type drop-down menu
2	Certificate Name field	**4**	Certificate Value field

To configure certificates:

Step 1 Navigate to the Directory Integration page by choosing the **Administration** module and then clicking the **Directories** tab.

Step 2 In the page navigator, click the **Datasources** option, if not already selected.

Step 3 Next to the datasource to which you wish to add a certificate, click **Edit**.

Step 4 Click **Add Certificate**.

Step 5 Name the certificate. Do not use spaces or special characters in the certificate alias name.

Step 6 From the Certificate Type drop-down menu, choose the certificate type.

Step 7 Paste the certificate value (obtained from a vendor like VeriSign) into the certificate field.

Step 8 Click **Update**.

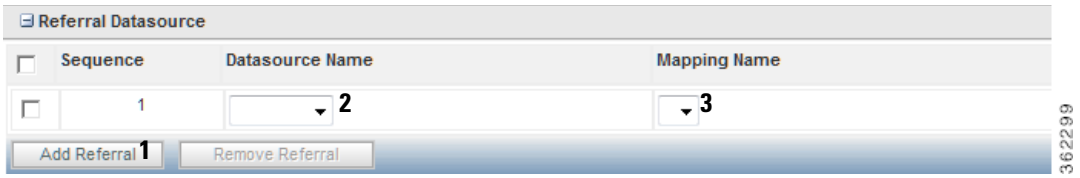
Configuring Referral Datasources

If you have multiple datasources configured, you can designate datasources as referral systems to a selected datasource. This way, whenever the system performs a search against the selected datasources, it will also search all referral datasources.

The referral datasources are searched in the order in which they are specified until a match is found. A match is said to be found when the search criteria returns one or more records.

Referral datasources are typically used when directory information is divided among multiple directories. For example, different company divisions may each maintain their own directory.

Figure 8-10 Configuring Referral Datasources



1	Add Referral button	**3**	Mapping Name drop-down menu
2	Datasource Name drop-down menu		

To configure referral datasources:

Step 1 Navigate to the Directory Integration page of the Administration module.

Step 2 In the page navigator, click the **Datasources** option, if not already selected.

Step 3 Next to the datasource for which to configure a referral datasource, click **Edit**.

Step 4 Click **Add Referral**.

Step 5 The Referral Datasource area appears. From the Datasource Name drop-down menu choose a datasource name, and then from the Mapping Name drop-down menu choose a mapping name.

Step 6 Click **Update**.

Testing the Connection

If you have completed all the necessary configuration steps, then you are ready to test the directory integration connection.

Figure 8-11 Testing the Connection

1	Test Connection button	**2**	Test Status column

To test the connection:

Step 1 Navigate to the Directory Integration page in the Administration module.

Step 2 In the page navigator, click the **Datasources** option, if not already selected.

Step 3 Choose the datasource to test by checking the check box to the left of the datasource name.

Step 4 Click **Test Connection**.

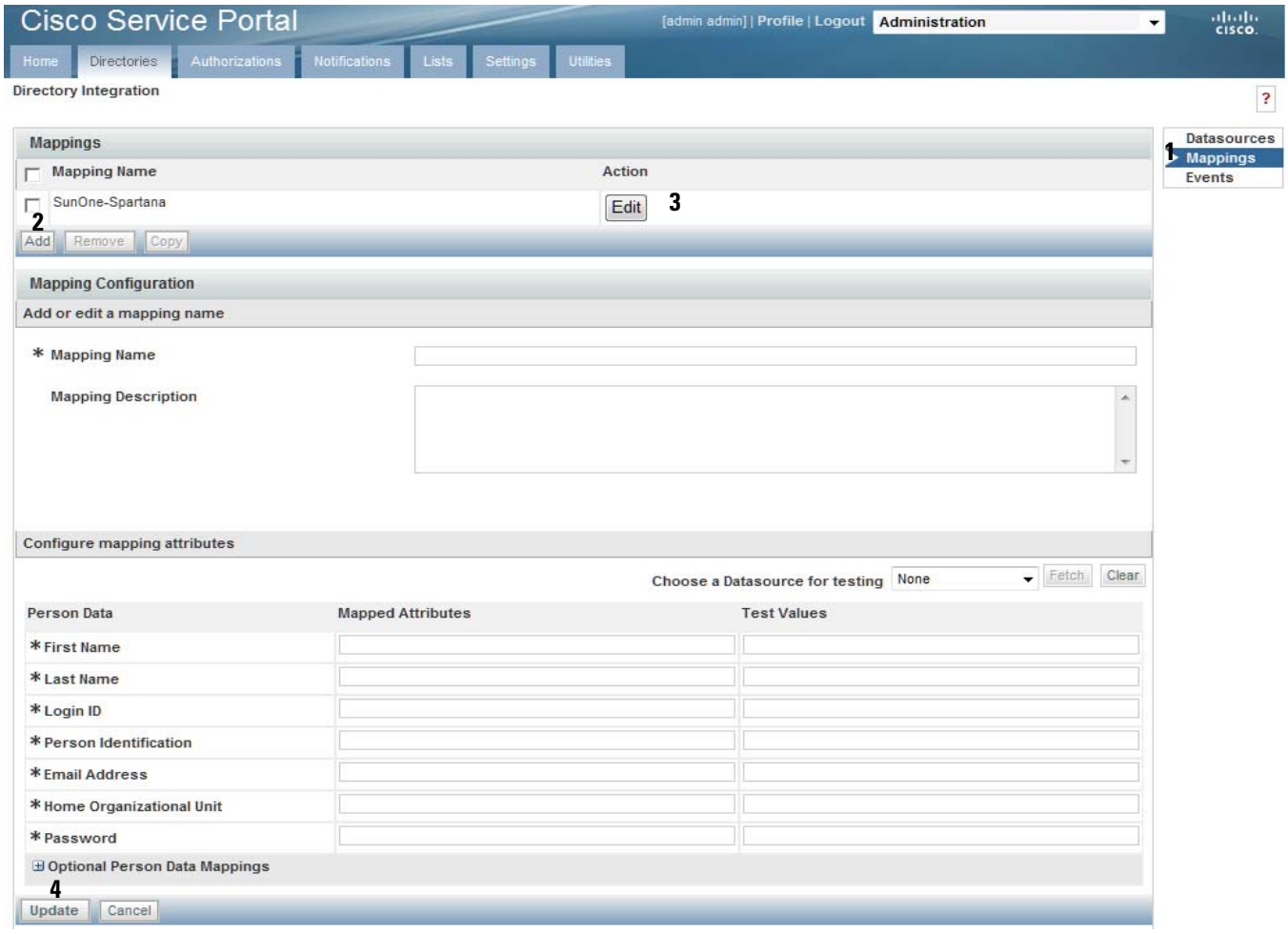
The Test Status column displays OK if the connection is successful, and if it is unsuccessful.

Configuring Mappings

You use the Mappings area of the Administration module's Directories tab to map Service Catalog data to directory server data.

To configure mapping, see Configuring Mapping and follow the procedure below.

Figure 8-12 Configuring Mapping



1	Mappings options	**3**	Edit Mapping button
2	Add Mapping button	**4**	Update button

Step 1 Navigate to the Directory Integration page of the Administration module.

Step 2 In the page navigator, click the **Mappings** option.

Step 3 Click **Add** to add a new mapping, or click **Edit** next to the desired mapping in the list to edit an existing mapping.

The Mapping Configuration area expands.

Step 4 Configure the mapping name, description, and attributes, based on the requirements documented in the Mapping Worksheet. The mappings prefixed with an asterisk (*), shown in the Person Data section, are mandatory. You may also configure optional mappings by clicking the ⊞ button, to expand the Optional Person Data Mappings section.

Step 5 Click **Update**.

The mapping fields accept simple, composite, expression, and Java mapping types, as described below.

Mapping Types

This section describes accepted mapping types, illustrates a valid sample mapping, and explains with examples expression mapping. The following table describes the supported mapping types.

Table 8-12 Mapping Types

Mapping Type	Description
Simple	One directory attribute maps to the field. This is simple one-to-one mapping. For example: Person Field: First Name Directory Attribute: givenName
Composite	A combination of attributes maps to the field. # delimits each attribute name. The mapping may include literals. For example: Person Field: Email Directory Attributes: #givenName#_#sn#@#domain#.com
Expression	An expression uses regular expressions and pattern matching to derive the mapping. For more details see the Expression Mapping.
Java Class	Use Java mapping when simple, composite, or expression mapping does not offer the desired functionality. This involves writing a Java class and placing the compiled class file on the appropriate directory on the application server. For more details see the Java Class Mapping.

Simple and Composite Mappings

The following table illustrates sample simple and composite mappings for the mandatory fields.

Table 8-13 Sample Mapping

Person Data	Directory Value
First Name	givenName
Last Name	sn
Login ID	uid
Person Identification	uid
Email Address	#givenName#_#sn#@#company#.com
Home Organizational Unit	Ou
Password	Uid

Expression Mapping

Expression mapping allows you to conditionally assign a value to an attribute, based on which pattern (regular expression) the expression matches. The system expression mapping uses the Perl5 Regular Expression Language, to specify patterns to be matched, combined with syntax similar to that of the Java conditional operator. Syntax:

```
expr:<expression>=(<patternlist>)?(<valuelist>):<default>
```

where

expr	is a prefix to indicate that expression mapping is used.
<expression>	is the expression to match against.
<patternlist>	is a set of patterns, separated by a pipe (|).
<valuelist>	is a set of values, separated by a pipe (|), corresponding to the set of patterns. Each value designates the return value if the expression matches the corresponding pattern.
<default>	is the return value to use if no pattern in the <patternlist> matches the <expression>.

For example:

```
expr:<expression>=
(<pattern1>|<pattern2>…<patternn>)?(<value1> | <value2>  <valuen>):<default>
```

If <expression> matches <pattern1>, then return <value1>.

If <expression> matches <pattern2>, then return <value2>.

If <expression> does not match any pattern, then return <default>.

Each element (expression, pattern, or value) can contain a directory attribute name, delimited by the # symbol. For example, a pattern can be specified as "#givenName#_#sn#", where both #givenName# and #sn# are attribute names:

In addition, parentheses can be used to group a series of pattern elements to a single element. When you match a pattern within parentheses, you can use back-references, in the form of $1, $2, and so on, to refer to the previously matched pattern.

Examples of Expression Data Mapping

A simple use of an expression applied to directory integration may be to translate one or more coded values in the directory to more user friendly descriptions or broader categories. For example, some services may need to differentiate between employees and contractors. The costCenter attribute is known to be "000000" for contractors. Therefore, the following expression could be applied to the "Employee Type" field:

```
expr:#costCenter#=(000000)?(Contractor):Employee
```

Another suse of an expression is to supply a default value for a field when the source attribute is blank. This may frequently be a "stop gap" measure, until directory data can be standardized. Or it could be standard; for example, if outside contractors are not assigned a department. The following expression could be applied to the "Home OU" field (a mandatory field for the mapping):

```
expr:#DeptLevel2#=(.+)?(#DeptLevel2#):Contractors
```

This expression uses the DeptLevel2 attribute if available, or defaults to the "Unknown" Business Unit for the user's Home OU.

Similarly, the expression can be used to translate from a set of input values to a different set of return values. This is the equivalent of a case statement, or nested if/then construct. For example, the following expression could be applied to the "Locale ID" field, to assign a language for the user, based on his/her location:

```
expr:#country#=(United States | Germany)?(en_US | de_DE):en_US
```

If the user's country is the United States, set the language to American English; if it is Germany, set the language to German. For any other country, set the language to American English.

Regular expressions can check the length of a source attribute and whether it is composed of alphabetic or numeric characters. For example, sometimes zip codes are stored as numeric data types, truncating leading zeroes. To restore a leading zero, an expression such as the following could be applied to the "Company Postal Code" field:

```
expr:#postalCode#=(^[1-9][0-9][0-9][0-9]$)?(0#postalCode#):#postalCode#
```

If the postalCode attribute consists of precisely four digits, add a leading zero to the value of the attribute. This converts zip code **1701** to **01701**, and leaves any source values which do not match the specified pattern unchanged.

A similar use of regular expressions might check that the format of an attribute value matches an expected pattern. Consider a use case in which a valid manager's user ID needs to consist of two letters followed by a series of numbers. Valid IDs would be, for example, fd1024 and ID3839. The following expression could be used:

```
expr:#manager#=(cn=([a-zA-Z][a-zA-Z][0-9]+),.*)?($1):None
```

Attributes can be used in the expression, pattern, or return value:

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(All Smiths|Only John):Others
```

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(#givenname#|Only John):Others
```

The last name and first name from directory records are combined into a string such as "Doe, Jane" before any attempt is made to match the patterns.

Embedded parentheses and back-references are useful for extracting a portion of the pattern. For example, the organization to which a person belongs is frequently embedded within a distinguished name (dn) attribute:

```
dn: cn=plee,ou=Corporate,dc=InfoSys,dc=com
```

The expression mapped to the "Home Organizational Unit" field might have the format:

```
expr:#dn#=((cn=[^,]+,ou=([a-zA-Z]+),dc=InfoSys,dc=com)?($1):Default
```

The returned value, "Corporate" is a back-reference value $1, which equals the pattern matched by the expression within the first set of parentheses, ([a-zA-Z]+).

Usage of back-referenced variables may be required to parse overloaded attributes which include the values for more than one field. For example, an attribute can include the business address of a person, including the building name, floor (level), and office.

```
location=Corporate Headquarters-Fifth Floor-Office #5F
```

The same pattern could be used to match the three elements in the expression, by using different back-referenced variables as the value:

Office Building	expr:#Location#=(([^-])+-([^-])+-(.*))?($1): Unknown
Building Level	expr:#Location#=(([^-])+-([^-])+-(.*))?($2): Unknown
Cubic Location	expr:#Location#=(([^-])+-([^-])+-(.*))?($3): Unknown

Java Class Mapping

You will need to be familiar with Java programming and have a Java development environment set up in order to implement a custom Java class to map directory data to fields.

Any custom mapping class must follow the guidelines given in "Using Custom Code in Directory Integration" section on page 8-36. The mapping class must implement an IEUIAttributeMapping interface.

The developer must follow the guidelines below to test and install the custom code module.

1. Install a Java IDE of choice, and set up a project for developing custom mapping code.
2. Edit the custom code file to fulfill your requirements.
3. Compile.
4. The custom Java class must be installed on the Service Catalog web archive (war), to be accessible to the Service Catalog service. Create a directory in RequestCenter.war/WEB-INF/classes to correspond to the package. Such directories are typically named:

   com/newscale/client/<clientname>, for example, com/newscale/client/aib.
5. Copy the CustomMapping.class file to the directory created in the previous step.
6. Restart the Service Catalog service.
7. Specify the fully qualified name of the class file as the Mapped Attribute for the field to be populated.
8. Test the custom code by using the Directories Test feature.
9. Save your source in an appropriate repository.

Testing Mappings

You can use the Mapping Test feature to test that your data mapping settings are configured correctly and pulling the correct values from the directory server.

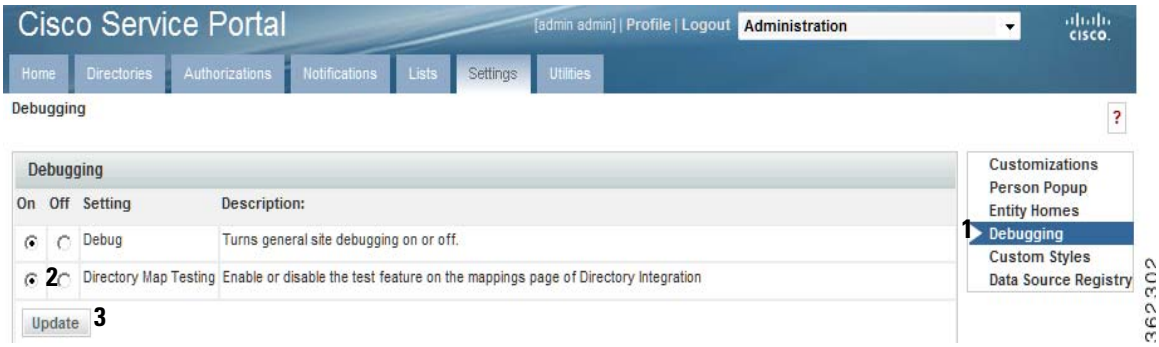
Using the Data Mapping Test feature involves:

- Enabling the Data Mapping Test Feature
- Using the Data Mapping Test Controls

Enabling the Directory Map Testing Feature

To enable the directory map testing feature, see Enabling Mapping Testing and follow the procedure below.

Figure 8-13 Enabling Mapping Testing



1	Debugging option	**3**	Update button
2	Directory Map Testing setting		

Step 1 Click the **Settings** tab of the Administration module to display the Settings page.

Step 2 In the page navigator, click the **Debugging** option.

The Debug Settings page appears.

Step 3 Next to the Directory Map Testing setting, click **On**.

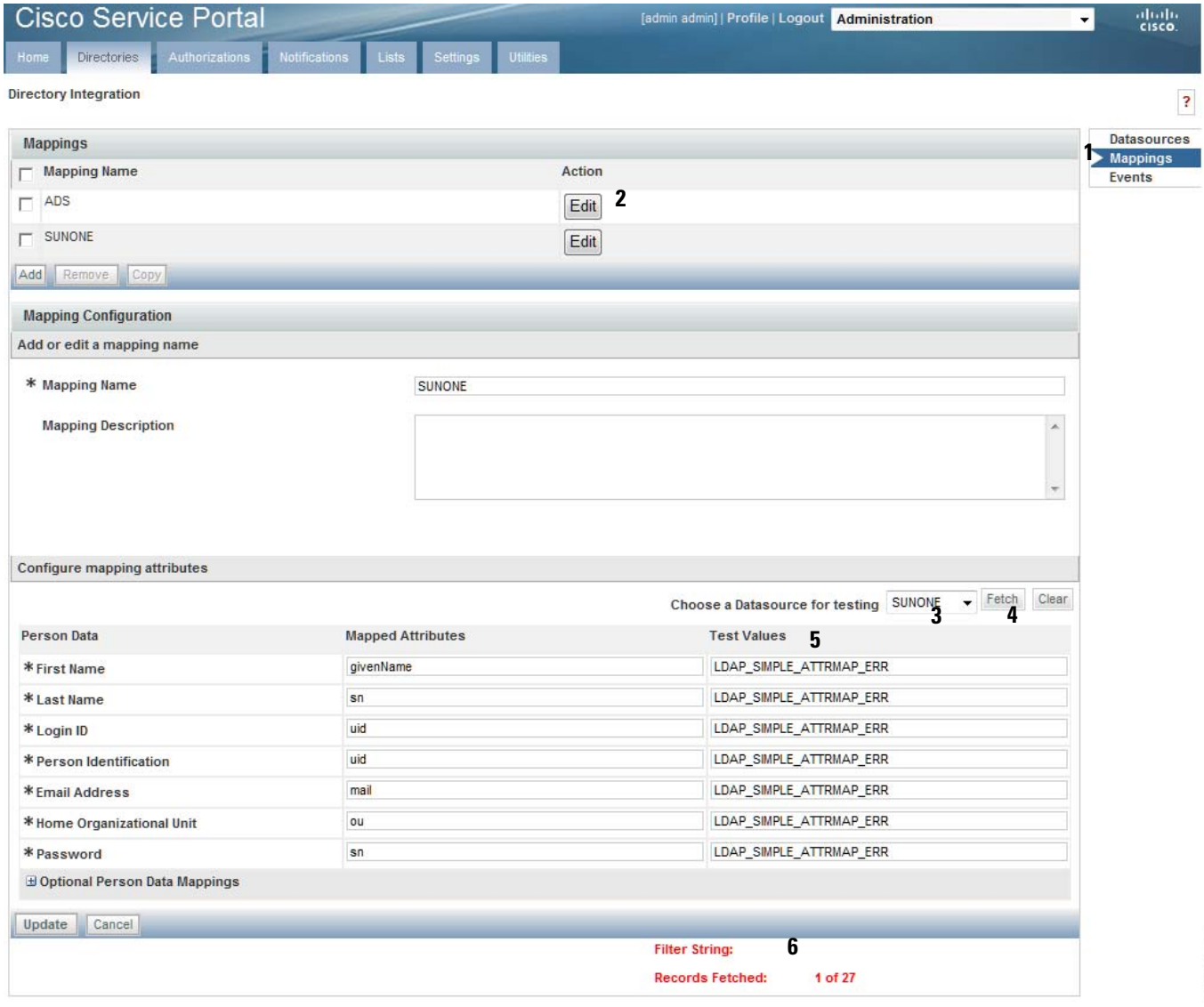
Step 4 Click **Update**.

The system enables the Data Mapping Test feature. Now when you access the Data Mapping tab, the following additional features appear as shown in Mapping Test Controls:

- The Choose a Datasource for Testing drop-down menu
- Fetch
- Clear
- Test Values

Using the Data Mapping Test Controls

Figure 8-14 Mapping Test Controls



1	Mappings option	**4**	Fetch button
2	Edit button	**5**	Test Values column
3	Choose a Datasource for testing drop-down menu	**6**	Test summary area

To use the Data Mapping test controls:

Step 1 Click **Mappings**, if you are not already on the Mapping page.

Step 2 Next to the mapping you wish to test, click **Edit**.

Step 3 From the "Choose a Datasource for testing" drop-down menu, choose the desired datasource.

Step 4 In the Test Values column, enter test values. You can use simple, composite, Java, or expression mapping.

Step 5 Click **Fetch**.

Step 6 The test values appear in the Test Values column and a summary of the results appears at the bottom of the page.

Note Fetch returns values from only one datasource and does not search referrals. This is for convenience because it becomes difficult to debug with referrals search integrated.

Step 7 To the right of the Fetch button, click **Clear** and retry new values until you have configured the desired mappings.

Configuring Directory Integration Events

You use the Events area of the Administration module's Directories tab to configure directory integration behavior for the following events:

- Login
- Person Lookup for Order on Behalf
- Person Lookup for Service Form
- Person Lookup for Authorization Delegate

To configure events, see Configuring Events and follow the procedure below.

Step 1 Navigate to the Directory Integration page of the Administration module.

Step 2 In the Page Navigator, click **Events** to display the Events page.

Step 3 Next to the type of event to configure, click **Edit**.

The Event Configuration area appears.

Step 4 From the Event Status drop-down menu, choose **Enabled** to enable the event.

Step 5 Click **Add step** to add a step for the system to initiate when the selected event occurs.

Step 6 Choose an operation associated with the step you just added.

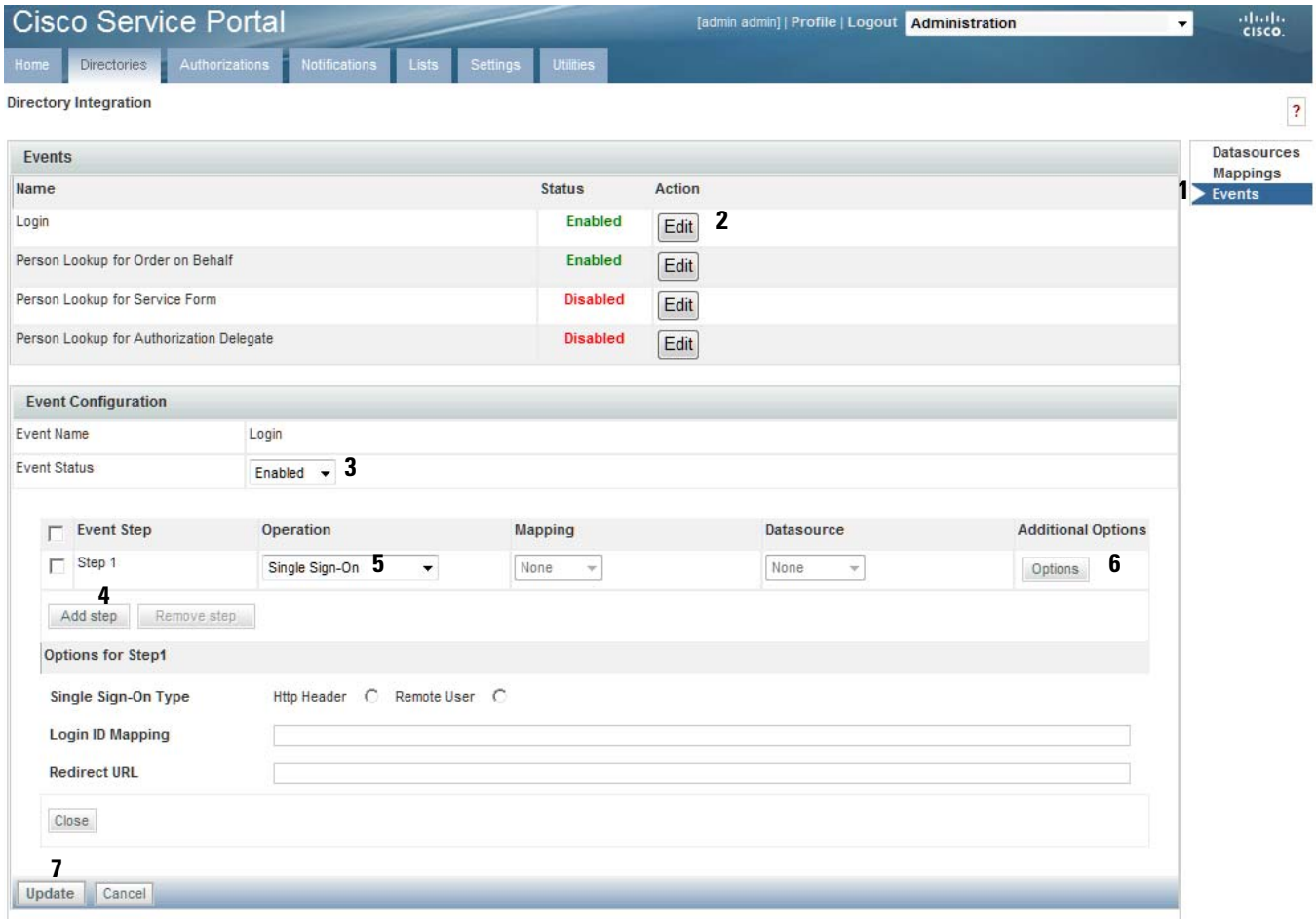
- All operations are available in this menu even though some operations, such as SSO and EUA, are not applicable for all event types.

Step 7 Click **Options** to configure the options associated with the operation you just chose. The Options area appears. The Options area will differ according to which operation is chosen. Details on the available operations and their options are given in the next section.

Step 8 Configure the associated options. See the relevant sections in this chapter on directory Events for a description of the operations available and options for configuring them.

Step 9 Click **Update** and repeat these steps for each step and operation you wish to add.

Figure 8-15 Configuring Events



1	Events option	**5**	Operation drop-down menu
2	Edit	**6**	Options
3	Event Status drop-down menu	**7**	Update
4	Add step		

Using Custom Code in Directory Integration

The directory integration framework is designed for flexibility and customization of the "Login" and "Person Lookup" events.

Standard operations for all events are available on the Administration module's Directories tab. These include: SSO, External User Authentication, Import Person, Import Manager, and Person Search.

In cases where these standard operations do not fully satisfy a business scenario, the Directories tab also provides interfaces to execute custom Java code. This custom code should adhere to the interfaces described in this chapter, and you should develop any customized solutions using Service Catalog exposed APIs.

The following are valid use cases for scenarios in which you may wish to customize an event operation:

Table 8-14 UseCases

If…	Then…
The format of SSO headers input through the HttpServletRequest cannot be parsed …	Provide a custom code SSO operation to retrieve user credentials, in order to support the SSO integration with your vendor.
You wish to authenticate a user via a web service or database other than Service Catalog…	Provide a custom code External Authentication operation.
The main user repository in your company is a database other than an LDAP directory…	Provide custom code External Authentication and custom code Import Person operations.

The directory integration custom code framework also defines interfaces that can be implemented to provide complex retrieval logic for a specific field in the person/user profile from a record in an external datasource.

Public APIs and interfaces for directory integration include the:

- **Custom Code Operation Interfaces**, which are used to customize directory integration operations.
- **Custom Java Class Mapping Interface**, which is used to provide customized retrieval of a specific attribute in an external datasource from its record.
- **Directory Server API**, used to query/authenticate against an external datasource and retrieve records.
- **Import/Refresh Person API**, used to update person attributes in the Service Catalog database.

A typical custom code project will involve following types of activities:

- Identify the need for custom code.
- Configure the Directories tab in the Administration module to include the Datasource to be used by your custom code and, if relevant, the Mappings which your custom code will use.
- Develop the custom code. You will need to understand the public APIs and interfaces provided by Cisco for directory integration tasks.
- Build and deploy the custom code.
- Configure the Directories tab in the Administration module to use your custom code.

Directory Integration Operations below summarizes the directory integration operations in more detail.

Table 8-15 Directory Integration Operations

Operation	Purpose	Input	Output
Single Sign-On	Identifies the login name of the user	HttpServletRequest	Login Name
External Authentication	Authenticates a user against an external datasource	Login Name Password	Authenticity of User
Person Search	Retrieves the list of persons matching first name or last name	First Name and Last Name	List of Persons
Import Person	Imports a person into the Service Catalog database from an external datasource.	Login Name	Imported person information, including the managerID
Import Manager	Imports a manager or chain of managers into the Service Catalog database from an external datasource	Imported person information including manager information	Managers are imported into the system

Mixing and matching, or replacing, standard operations with custom code operations is also supported by the directory integration framework. Service Catalog supports various combinations of operations per event, as described in the table below, using your own customized code and Service Catalog public APIs, designed to help implement these interfaces.

It is important that custom code design and development engineers understand the directory integration framework, public APIs, and custom code interfaces, which are discussed in detail in this chapter.

Table 8-16 below portrays the relationship between methods, events, and operation types for custom code operations. Combinations not listed in Table 8-16 below are not supported.

Table 8-16 Custom Code Operations

Event	Operation Type	Interface	Method
Login	SSO	ISignOn	getCredentials
	EUA	ISignOn	authenticate
	Import Person	ISignOn	importPerson
	Import Manager	ISignOn	importManager
	Custom Code	ISignOn	performCustom
Person Search for: • Order On Behalf • Authorization Delegate • Service Form	Person Search	IPersonSearch	getCredentials
	Import Person	IPersonSearch	importPerson
	Import Manager	IPersonSearch	importManager
	Custom Code	IPersonSearch	performCustom

Custom Code Operation Interfaces

If you are providing a custom implementation of an operation configured within an event, you will need to implement a “custom code operation interface”.

Custom code operation interfaces define callback methods that are invoked when a particular operation is triggered. Exactly which method is invoked depends on the operation type chosen in the operation. For more details see the Method, Event, and Operation Type for Custom code Operations table. All methods defined in the custom code operation interfaces follow the same pattern:

Parameters

In the following list, “**” must be replaced by the operation type, which is one of:

- IEUISignon
- IEUIPersonSearch

1. **OperationDTO: This object contains the information on how you have set the operation on the Directories tab of the Administration module. It includes mapping and datasource information.
2. **OperationContext: The Context object is used to share information across method invocations. The directory Integration framework makes information stored in one context object available to other context objects during the same HttpServletRequest invocation.
   a. Use setLocalContextObject and getLocalContextObject to set any custom information that does not fall as a part of results.
   b. Use get**Result to get a result object. Result objects contain all the information about what happened throughout the event request. Results contain information that is supported in a productized import. The LocalContext object is used to store objects that were unforeseen during the implementation of productized operations.
3. Request: This is the HttpServletRequest.
4. **ImportAPI: This object is used to import a person. More details can be found in the Javadocs.
5. **LDAPAPI: This API is used to make LDAP queries. More details can be found in the Javadocs.

Return

**Result. After performing the custom task the API must return a valid return type with results populated. Return the same result object retrieved from OperationContext after updating relevant properties. There may be unexpected behavior if a new instance of the result object is returned.

Table 8-17 below maps the expected input/return to the objects in the parameters of each of these callback methods:

Table 8-17 Input for Custom Code Callback Methods

Information	Object/Property
HttpServletRequest	Request
Login Name	• IEUISignOnOperationContext .IEUISignOnOperationResult.ssoLoginId • IEUIPersonSearchOperationContext .IEUIPersonSearchOperationResult.ssoLoginId
First Name and LastName	• First Name: IEUIPersonSearchOperationContext. firstNameSearchString • Last Name: IEUIPersonSearchOperationContext. lastNameSearchString

Table 8-17 Input for Custom Code Callback Methods

Information	Object/Property
List of Persons	IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.SearchPersonList. SearchPersonList is a collection with all elements of type IExtPersonDTO
Imported Person Information	• IEUISignOnOperationContext .IEUISignOnOperationResult.ImportedPersonExtDTO • IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO
Manager Id	IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO.PersonDTO.managerID

You must implement all methods to compile your implementation class. If you customize only limited operation types, you must provide an empty implementation of methods not relevant to the operation types.

For example, if you are only interested in a customized SSO, then provide a complete implementation of the getCredentials method. For all other methods, return null.

The system may pool an instance of an interface and may be concurrently accessed from multiple threads. Thus, it is recommended to keep the instance stateless.

There are two types of custom code operation interfaces:

- **ISignOn** is used for customizing the login.
- **IPersonSearch** is used for customizing the “Person Lookup” dialog box.

Custom Code Interface for Login Event – ISignOn

This is the interface that custom code should implement in order to customize login events: SSO, EUA, Import Person, Import Manager and custom code operations.

Customizing the SSO Operation

The primary purpose of an SSO custom code operation is to retrieve and return the Login Name from HttpHeader based Sign-On or from CGI Header (CGI variable REMOTE_USER) in the case of Remote NTML/IWA type of Sign-On.

As outlined in Table 8-16, you must provide a Java class that implements the ISignOn interface. Please provide a complete implementation of the getCredentials method in this interface, and read the documentation for the ISignOn interface for detailed specifications.

The following are some guidelines for implementing the getCredentials method. It is not required that all of these guidelines are implemented; There may be additional requirements, dependent on the customization, which are not covered by what is outlined below.

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- Use the parameter request and process it to derive the login name of the person.
- Return LoginId back by calling IEUISignOnOperationResult.setSsoLoginId(<login id>), if using the in-product directory lookup functionality.

- Call IEUISignOnOperationResult.setSsoRedirectUrl("<any url or error page>"), which are used for redirecting the user on SSO failure.

SSO Operation Options received through IEUIEventSSOOperationDTO.getEventSsoDTO() may be null as SSO options are not accepted in the Administration module for custom code operations.

Customizing the EUA Operation for Login Event

The primary purpose of an EUA custom code operation is to authenticate a user against an external system.

As outlined in Table 8-16, you must provide a Java class that implements the ISignOn interface. Please provide a complete implementation of the authenticate method in this interface, and read the documentation for the ISignOn interface for detailed specifications.

The following are some guidelines for implementing an EUA operation. It is not required that all of these guidelines are implemented; There may be additional requirements, depending on the customization, that are not covered below.

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- The EUIDatasourceDTO object from the IEUIEventEUAOperationDTO object contains the interface to the Datasource configured in the Administration module for this operation.
- Populate the LDAPConfigInfo object from the EUIUtil and pass EUIDatasourceDTO. This is needed to call LDAP API with the connection information to LDAP Server.
- Get the Login Name by calling IEUISignOnOperationResult.getSsoLoginId().
- Form a BindDN and set it into LDAPConfigInfo by calling setBindDN().
- Get the Password entered by the user in the Login page by calling IEUISignOnOperationResult.getEuaPassword().This function returns BASE64 encoded password. To convert this string into a plain password, perform the following steps programmatically:
  - Remove the prefix “!@^_” and suffix “_^@!” from the string.
  - Use Java’s Base64 decode method on the above resultant string.
- Set it into LDAPConfigInfo by calling setBindPassword().
- Authenticate the user against the Directory Server by passing the LDAPConfigInfo object ILDAPApi.authenticate() API.
- If the user has been authenticated, then call IEUISignOnOperationResult.setEuaAuthenticated(true).
- If the user authentication failed or any exception occurred, then call IEUISignOnOperationResult.setEuaAuthenticated(false).

EUA Operation Options received through IEUIEventEUAOperationDTO.getEventEuaDTO() will be empty as EUA options are not accepted in the Administration module for custom code operations.

Customizing the Import Person Operation for the Login Event

The primary purpose of the Import Person operation is to import/refresh a user from an external system, like a directory server or an external database, into the Service Catalog application.

As outlined in Table 8-16, you must provide a Java class that implements the ISignOn interface. Please provide a complete implementation of the importPerson method in this interface, and read the documentation for the ISignOn interface for detailed specifications.

The following are some guidelines for implementing an Import Person operation. It is not required that all of these guidelines are implemented; There may be additional requirements, dependent on the customization, which are not covered below.

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- The EUIDatasourceDTO object from the IEUIEventImportPersonOperationDTO object contains the interface to the datasource configured in the Administration module for this operation.
- The EUIDataMappingDTO object from the IEUIEventImportPersonOperationDTO object contains the interface to the mapping configured in the Administration module for this operation.
- Using the Login Name retrieved from the IEUISignOnOperationResult.getSsoLoginId() method, query for the user on the external system either from an LDAP server or an external database and collect all information related to the Person profile, including organizational units, groups, and roles.
- Check to see if the user already exists in the Service Catalog database by calling ISignOnImportPersonAPI.getPersonByLoginName(<Login Id>). If the person already exists, this method returns the IPersonDTO object. If the person does not exist, the method throws a signOnImportPersonAPIException.
- If the person is not found, create an IPersonDTO object through the PersonFactory.createPersonDTO() method in preparation for importing the person.
- From the data fetched from the external system, create these DTOs using PersonFactory and populate them as well: IPersonDTO, ILoginInfo, IContactDTO, IAddressDTO, and IPersonExtensionDTO.
- Begin the database transaction by calling ISignOnImportPersonAPI.beginTransaction().
- Check to see if an organizational unit (OU) exists by calling ISignOnImportPersonAPI.getOrgUnitByName(<OU Name>). If it does, this method returns an IOrganizationalUnitDTO object. If the organizational unit does not exist, the method throws a signOnImportPersonAPIException.
- If an OU does not exist, it may be created by calling ISignOnImportPersonAPI.createOrgUnit(<IOrganizationalUnitDTO>).
- If the user already exists, call ISignOnImportPersonAPI.updatePerson(<IPersonDTO>). This updates a person's basic profile, login information, preferences, Home OU and extensions.
- If the user already exists, link/update addresses/location and contacts by calling ISignOnImportPersonAPI.linkAddresses(<IAddressDTO collection>) and ISignOnImportPersonAPI.linkContact(<IContactDTO>.
- If the person is associated with one or more groups in the external system, first try getting all the existing groups by calling ISignOnImportPersonAPI.getGroupByName (<ou name>). If not, create all the new groups by calling ISignOnImportPersonAPI.createGroup(<IOrganizationalUnitDTO>).
- If the person is new, link all the lists of OUs and groups to the user by calling ISignOnImportPersonAPI.linkPersonToOrgUnit() and ISignOnImportPersonAPI.linkPersonToGroup().
- If the person already exists, any OUs and groups may be unlinked from the user by calling ISignOnImportPersonAPI.unlinkPersonToOrgUnit() and ISignOnImportPersonAPI.unlinkPersonToGroup().
- To find out the existing associations of OUs, including the home OU, and groups for a person, call the ISignOnImportPersonAPI.getOrgUnitsForPerson() and ISignOnImportPersonAPI.getGroupsForPerson() methods.

- To find out the existing associations to roles, call the ISignOnImportPersonAPI.getRolesForPerson() method.
- If the imported person needs to be associated with a role, first get the role using ISignOnImportPersonAPI.getRBACRoleByLogicName(<roleLogicName>).
- Link/unlink roles to a person by calling ISignOnImportPersonAPI.linkPersonToRole() or ISignOnImportPersonAPI.unlinkPersonToRole().
- If the person was imported/refreshed successfully, set the flag ImportPersonDone = true into IEUISignOnOperationResult.
- After successful import/refresh, also create an object of IExtUserDTO through PersonFactory.createExtUserDTO() and set IPersonDTO and HomeOUDTO (IOrganizationalUnitDTO) into IExtUserDTO, then return the IExtUserDTO of the imported person by calling IEUISignOnOperationResult.setImportedPersonExtDTO(<IExtUserDTO>).
- If the import/refresh operation failed, set the flag ImportPersonDone = false into IEUISignOnOperationResult.
- End/commit the database transaction by calling ISignOnImportPersonAPI.commitTransaction().
- If the transaction failed, roll back the transaction in the `exception` block by calling ISignOnImportPersonAPI.rollbackTransaction() and releasing the transaction in the `finally` block by calling ISignOnImportPersonAPI.releaseTransaction().

Import Person operation options through the IEUIEventImportPersonOperationDTO.getImportPersonDTO() method will be empty as Import Person options are not accepted in the Administration module for custom code operations.

Customizing the Import Manager Operation for the Login Event

The primary purpose of the Import Manager operation is to import/refresh the Supervisor chain of the person from an external system, like a directory server, into Service Catalog.

As outlined in Table 8-16, you must provide a java class that implements the ISignOn interface. Please provide a complete implementation of the importPerson method in this interface, and read the documentation for the ISignOn interface for detailed specifications.

The following are some guidelines for the Import Manager operation:

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- Get the user imported/refreshed user ImportedPersonExtDTO from IEUISignOnOperationResult.
- Get the Person who was imported through the IEUISignOnOperationResult.getImportedPersonExtDTO() method. This will return a IExtUserDTO object, from this get IPersonDTO object.
- Import all managers as needed from the external system, create/update each manager in the same way as explained in Import Person example above.
- Link a manager to a person, assuming personDTO is a reference to IPersonDTO for the imported manager and managerDTO is a reference to the IPersonDTO returned after the manager is imported.
- Use personDTO.setManagerId(managerDTO.getId() to set the manager association for personDTO.
- Save the association by saving personDTO using one of the mechanisms explained in the Import Person Operation.

It is recommended that when importing the manager chain, you import the top level managers before persons. This avoids unnecessary updates for personDTO to update the link with the person's manager.

Import Manager Operation Options received through IEUIEventImportManagerOperationDTO. getImportManagerDTO() will be empty as Import Manager Options are not accepted in the Administration module for custom code operations.

Customizing Custom Operations for the Login Event

The primary purpose of the custom code operation is to perform any custom operation that is needed and not represented elsewhere in the application.

The following are some guidelines for the Custom Code operation:

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- Get EUIDatasourceDTO from IEUIEventCustomOperationDTO. This object contains the datasource configured in the Administration module for this operation.
- Get EUIDataMappingDTO from IEUIEventCustomOperationDTO. This object contains the mapping configured in the Administration module for this operation.
- Perform any custom operation as needed.
- IEUISignOnOperationResult should be populated appropriately based on previous examples.

Custom Code Interface for Person Lookup – IPersonSearch

This is the interface that a custom code should implement in order to customize Person Search events: Person Search, Import Person, Import Manager and custom code operations.

The implementation class is configured in the **Administration module > Directories tab > Events**, and can be configured for searching a for person in the following places within the Service Catalog application:

- Person Search for Order On Behalf
- Person Search for Authorization Delegate
- Person Search for Service Form

Customizing the Person Search Operation

The primary purpose of the Person Search operation is to search for users from an external system, like a directory server.

As outlined in Custom Code Operations table, you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the search method in this interface, and read the documentation for the ISignOn interface for detailed specifications.

The following are some guidelines for the Person Search operation:

- Get IEUISignOnOperationResult from IEUISignOnOperationContext. This is the object that must be returned from this interface.
- Since a custom Person Search operation can be configured using Person Search, we can add to, or manipulate, the search results from the previous operation in the Search Event by getting the list of persons already in the Search result by calling IEUISignOnOperationResult.getSearchPersonList().
- Search the users on an external system, either a directory server using the API methods in the interface ILDAPApi, or in an external database using the API in ISignOnImportPersonAPI for connecting to SQL datasources.
- For every person found on the external system, create IExtUserDTO.

- Populate IExtUserDTO with IPersonDTO, IOrganizationalUnitDTO (for Home OU) and ILoginInfoDTO.
- Optional – based on the person popup global setting, also populate collection IContactDTO, collection of IAddressDTO, IPersonExtensionDTO.
- Get the flag “All Users For Order On Behalf” using ISignOnImportPersonAPI getCustomParam(“ShowAllUsersForOrderOnBehalf”).
- To make the custom code consistent with the standard platform behavior, if the flag is Off, and any mandatory attributes are missing for the person, remove the entry. This will prevent any incomplete persons from being shown in the popup.
- To make the custom code consistent with the standard platform behavior, if the flag is On and the Person is missing any mandatory attributes, call IExtUserDTO.setResultHasError(true). This includes the incomplete person in the popup, but displays a red asterisk “*” instead of the radio button. The starred user cannot be chosen by the end user or imported.
- Return the list of all persons searched by calling IEUISignOnOperationResult. setSearchPersonList(<List of all IExtUserDTO>).

Person Search Operation Options received through the IEUIEventPersonSearchOperationDTO.getPersonSearchOperationDTO() method will be empty as Person Search Options are not accepted in the Administration module for custom code operations.

Customizing the Import Person Operation for Person Search Event

As outlined in Custom Code Operations table, you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the importPerson method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the Customizing the Import Person Operation for the Login Event.

Customizing the Import Manager Operation for Person Search Event

As outlined in Custom Code Operations, you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the search method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the Customizing the Import Manager Operation for the Login Event.

Customizing the Custom Operation for Person Search Event

As outlined in Custom Code Operations, you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the performCustom method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the Customizing Custom Operations for the Login Event.

Custom Java Class Mapping Interface

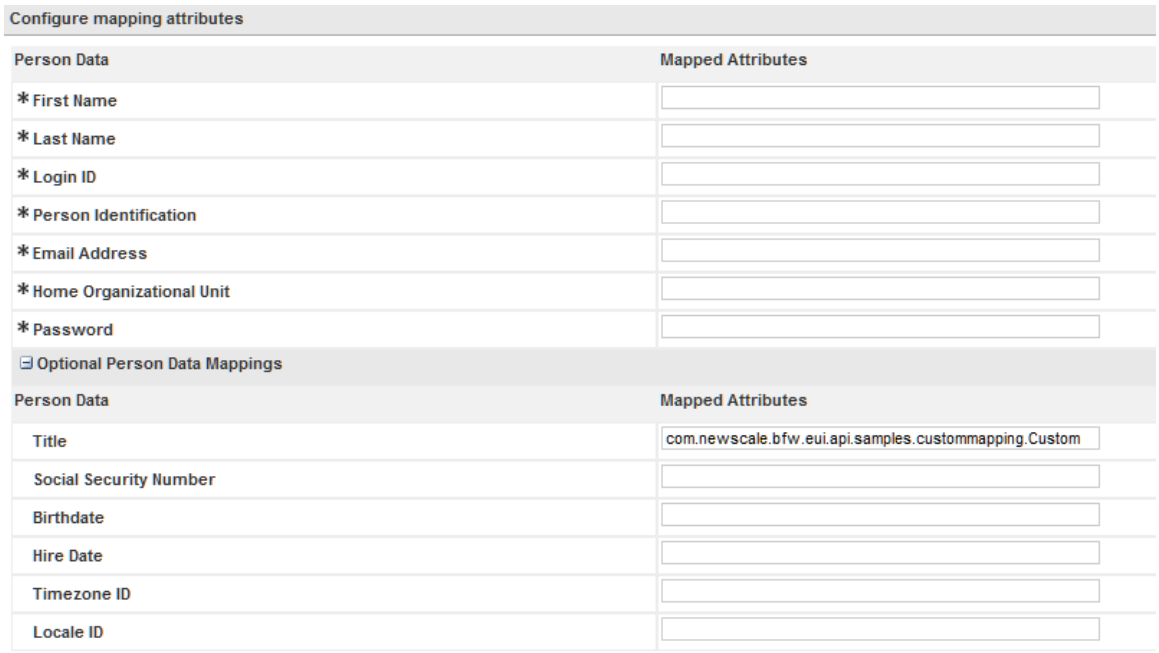
When simple, composite, or regular expression attribute mappings do not suffice, a custom Java class can be used in a directory integration attribute mapping.

Custom Java Class for Attribute Mapping – IEUIAttributeMapping

This is the interface that a custom code should implement in order to customize directory attribute mappings. The primary purpose of custom mapping class is to customize the attribute value fetched from the directory server.

The implementation class has to be configured in the **Administration module > Directories tab > Mappings**, and can be configured for any attribute in the mapping.

Figure 8-16 Custom Java Class for Attribute Mapping



The following are some guidelines for using a custom Java class mapping class:

- The mapping class should only be used for simple logic to be applied to the value retrieved from the directory.
- For performance reasons, the mapping class should not be used to perform a call to a directory server using the Directory Server API or to execute any database operations. The Person Search or Login interfaces should be used for these use cases.
- Implement IEUIAttributeMapping.getAttributeValue() for returning a single value for the mapped attribute. This method should not be implemented for the OU List, Group List, or Role List mapping fields.
- Implement IEUIAttributeMapping.getAttributeValueArray() for returning multiple values for the mapped attribute. This method should only be implemented for the OU List, Group List, and Role List mapping fields.

Directory Server API

This is an API wrapper that Cisco provides for integrating with the directory server (LDAP) connection facility built into the product.

Authentication to, and querying, the directory server are the only features this API provides. This API supports all directory servers supported by Service Catalog.

Typically, the Directory Server API works from the directory integration datasource and mapping configurations, and eliminates the need for hand-coding connection information, filters, and the attributes for querying.

Generally, to use the LDAP API, you also need the LDAPConfigInfo object. Use EUIUtil.get LDAPConfigInfo() from any datasource and mapping for this purpose.

The javadoc for LDAP API can be located in the javadocs folder of the product package.

Getting an Instance of ILDAPApi – API Implementation

An instance of ILDAPApi does not need to be created. It is available in all method arguments of both custom code API interfaces (ISignOn and IPersonSearch).

Directory Integration Utility (EUIUtil) Class

The directory integration utility class (EUIUtil) converts the datasource and mapping configured in the Administration module into a format that the Directory Server API can use as input for authentication, search, and query functions.

LDAP Configuration Info (LDAPConfigInfo) Class

An object of LDAPConfigInfo class encapsulates all the following configuration options that must be passed to the directory server API:

- Authentication information
- Connection information
- Query attributes
- Search filter

For more advanced users, if there is a need to override any configuration, LDAPConfigInfo provides getters and setters for all configurations. For further details on these methods, see the Javadoc for this class.

Main interface of the API – ILDAPApi

The ILDAPApi is the main interface that provides two basic operations on the directory server:

- Authenticate
- Search/Query

The ILDAPApi interface provides methods to interact with LDAP consistently throughout Service Catalog.

LDAPEntryBean

After querying/searching the directory server using the ILDAPApi.query(…) method, the results are returned as a collection of LDAPEntryBean.

Import/Refresh Person API

This API can be used to import/refresh Person profile, create OUs or groups and also to link or unlink a person to an OU, group, or role. This API also supports transaction management for importing a person, and connectivity to SQL datasources. This API includes a method to read from the CnfParams table.

Import/Refresh Person API Interface – ISignOnImportPersonAPI

The Import/Refresh Person API interface provides methods for the following:

- Get a Person object by PersonID or LoginName. This returns the Person with login information, preferences, home OU, address, contact, location, and extensions.
- Create a Person with login information, preferences, home OU, address, contact, location, and extensions.
- Update a Person with login, preferences, home OU, and extensions.
- Get OU by OrganizationalUnitID, Name. This does not return the members of the OU.
- Get all the OUs for a given Person. This does not return the members of the OU.
- Create an OU.
- Link/unlink a Person with an OU.
- Get Group by GroupID, Name. This does not return all the members of the Group.
- Get all the groups for a given person.
- Create a group.
- Link/unlink a person with a group.
- Get a user-defined role by name.
- Get LogicName object for a system-defined role.
- Get system-defined role by LogicName object.
- Get all the roles for a given person.
- Link/unlink a person with a role.
- Link/update address or location for a person.
- Add/update/delete a contact for a person.
- Begin transaction, commit transaction and release transaction resources for Import Person.
- Get a connection to a SQL datasource.
- Rollback the transaction on the SQL datasource connection.
- Return the connection to the SQL datasource back to the connection pool.
- Get parameter values from the CnfParams table.

For further details see the Java documentation.

Customizing Java Class to Connect to a SQL Datasource

To customize the Java class to connect to a SQL datasource:

Step 1 Get a connection to a SQL datasource database from ISignOnImportPersonAPI by passing the DatasourceName. The DatasourceName should be prefixed with the JNDI prefix, as defined by the "DatasourceJNDIPrefix" property in newscale.properties file.

Step 2 Use the above connection to execute any query using a JDBC statement.

Step 3 Commit the connection object directly at the end of the **try** block.

Step 4 Call ISignOnImportPersonAPI to roll back the connection when there are any failures/exceptions.

Step 5 In the final block, close the statement directly and call ISignOnImportPersonAPI to release the connection and return it to the connection pool.

Best Practices

Compiling Custom Code Java Files

The following are steps to compile and deploy custom code:

Step 1 Copy the build.xml file given in the Sample build.xml File and paste it to any folder; for example, C:\CustomCode.

Step 2 Edit the build.xml file to change the property "rcwar.dir" to point to the full path where the RequestCenter.war is available.

Step 3 Edit the build.xml to change the property "javax.servlet.dir" to point to the full path where the servlet-api.jar is available. This is specific to the application server.

Step 4 Create a subfolder for the custom code java files; for example, C:\CustomCode\src.

Step 5 Create a custom code with a package name like "com.newscale.SignOnCustomCode" and place the SignOnCustomCode.java file in the following directory: C:\CustomCode\src\com\newscale\SignOnCustomCode.java

Step 6 Run "ant" from a command line in the C:\CusomCode folder.

Step 7 The ant build file will compile all the java files under the "src" subfolder and place the class files in the "out" subfolder.

Step 8 The ant build file will also deploy the class files to the "RequestCenter.war\WEB-INF\classes" folder.

Step 9 Restart the application server.

Coding Guidelines

Package Names

- We recommend that the package name should be: com.newscale.[yourcompanyname].*.
- Use the key name “com.yourcompanyname.*” to store any ContextLocalAttributes. This eliminates clashes with the internal namespaces.

Logging

- Use the Logger to log messages to the server logs instead of using System.out.println.
- For debug logs, always begin by checking whether debugging is enabled. This is essential for performance.
- Always log the error in the exception block before propagating the exception back to the caller.

Exception Handling

- When EUIException is caught, throw it back as is.
- Wrap all other exceptions as EUIException and throw it back.

Configuring Custom Code in the Administration Module

After you have developed, compiled, and deployed the custom code, the Administration module must be configured to use the code. Configuration involves specifying when (in which event), in which operation and in what sequence (step) to invoke the custom code.

Step 1: Configure Global Settings

Ensure that the Directory Integration has been enabled by turning on this setting in the Administration module’s Settings tab. Instructions for turning on Directory Integration are given in the Enabling Directory Integration.

Step 2: Configure Datasources

Most operations, customized or not, require a datasource and mapping, so these two areas of the Directory Administration must be configured first.

Datasources are the external servers, such as LDAP, where your data is currently stored, and which Service Catalog must access. The only custom operation which does not require a datasource is SSO.

See the Defining Datasources and the Configuring Datasource Information for more information on configuring datasources.

Step 3: Configure Attribute Mappings

Once you have set up the external datasource, you must map the person-related data available in Service Catalog to the data in the LDAP directory (or other external datasource). These mappings tell Service Catalog where to look and what to get during an event and sequence of operations.

To configure a mapping follow the guidelines and instructions in the Defining Mappings and the Configuring Mappings.

Step 4: Configure Events/Customized Events

Customizing the Single Sign-On (SSO) and authentication operations for any event other than Login is considered an illegal action. There is no other time when these operations are necessary. Once a user is signed into and authenticated in the application from the external LDAP server, the process does not need to be replicated.

All events requiring connection to external datasources are configured here. When invoking the Custom Code APIs described in this guide, it is important to think through the sequence of operations for each event so that the custom operation does not occur out of order and fail.

Step 1 From the Navigation Pane, click **Events**.

Step 2 For the event you wish to customize, click **Edit**.

Step 3 If the event is disabled, use the drop-down menu to choose **Enabled**.

Step 4 Click **Add step** to add an operation. You can add as many steps are as necessary now, or complete the details of each step before adding and configuring the next.

Step 5 Choose the **Operation** from the drop-down menu.

- To simply invoke the code for SSO; for example, you can choose SSO from the menu. To *customize* the code for SSO, choose Custom Code, and then, in the next step, choose which operation you want to customize.
- To configure a customized operation, choose **Custom Code**.

Step 6 Choose your **mapping** and **datasource** from the drop-down menus.

Step 7 Under the "Additional Options" heading, click **Options**.

Step 8 Configure the options for that step:

- For Custom Code Operation Type, use the drop-down menu to choose the operation you wish to customize.
- For Java Class, enter the entire package name for that operation, followed by the class name; for example, com.newscale.bfw.eui.api.samples.operations.*CustomCodeTester*.
- In the above example, the Java class name is in *italics*. Both of these may be found in and copied from the code itself.

Step 9 Click **Close** to close the additional options for the step.

Step 10 Continue adding and configuring steps, as necessary.

Step 11 Click **Update** to save all steps for that event.

Using Custom Code as an Operation Type

In the steps above, if you choose Custom Code as the operation and Custom Code again for the operation type, you are then calling an undefined Custom Code, which you must design.

In the Custom Code test example provided by Cisco, you can use the Java Class "performCustom" to define your own custom code.

Deploying Custom Code

All custom code must be packaged as a customization to the Service Catalog installer. This allows the customizations to be reapplied if the installation needs to be upgraded or to install a new site.

Instructions for packaging and deploying custom code are dependent on the application server which hosts Service Catalog. See the *Cisco Prime Service Catalog 12.1 Integration Guide* and *Cisco Prime Service Catalog Administration and Operations Guide* for further information about configuring custom code.

Sample View/Usage of the API

The solution here satisfies these use cases:

- Create an event class that searches for a person using data collected from a container-managed SQL datasource.
- Create an event class that imports a person using data collected from a container-managed SQL datasource.
- Create an event class that modifies a person using data collected from a container-managed SQL datasource.
- Create an event class that can receive configuration parameters from the UI. The mappings interface is used in this example to pass the configuration parameters to the class.

It also creates the home OU for the person as a business unit, if it doesn't already exist in Service Catalog.

Note The solution requires a datasource to be configured on the application server. The following sections illustrate configuration and usage of the EUIPersonSearchSQL class.

SQL Datasource

Any SQL table or tables that contain data for the mandatory fields in a Person profile (or from which values for those fields can be derived) could be used as a datasource. Here is the table definition used in this example:

```
CREATE TABLE [psgextusers] (
  [login]     [nvarchar] (100) COLLATE Latin1_General_CI_AI NOT NULL,
  [firstname] [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [lastname]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [password]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [email]     [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [homeOU]    [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  CONSTRAINT [PK_extuser] PRIMARY KEY CLUSTERED
```

```
  (
     [login]
  )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

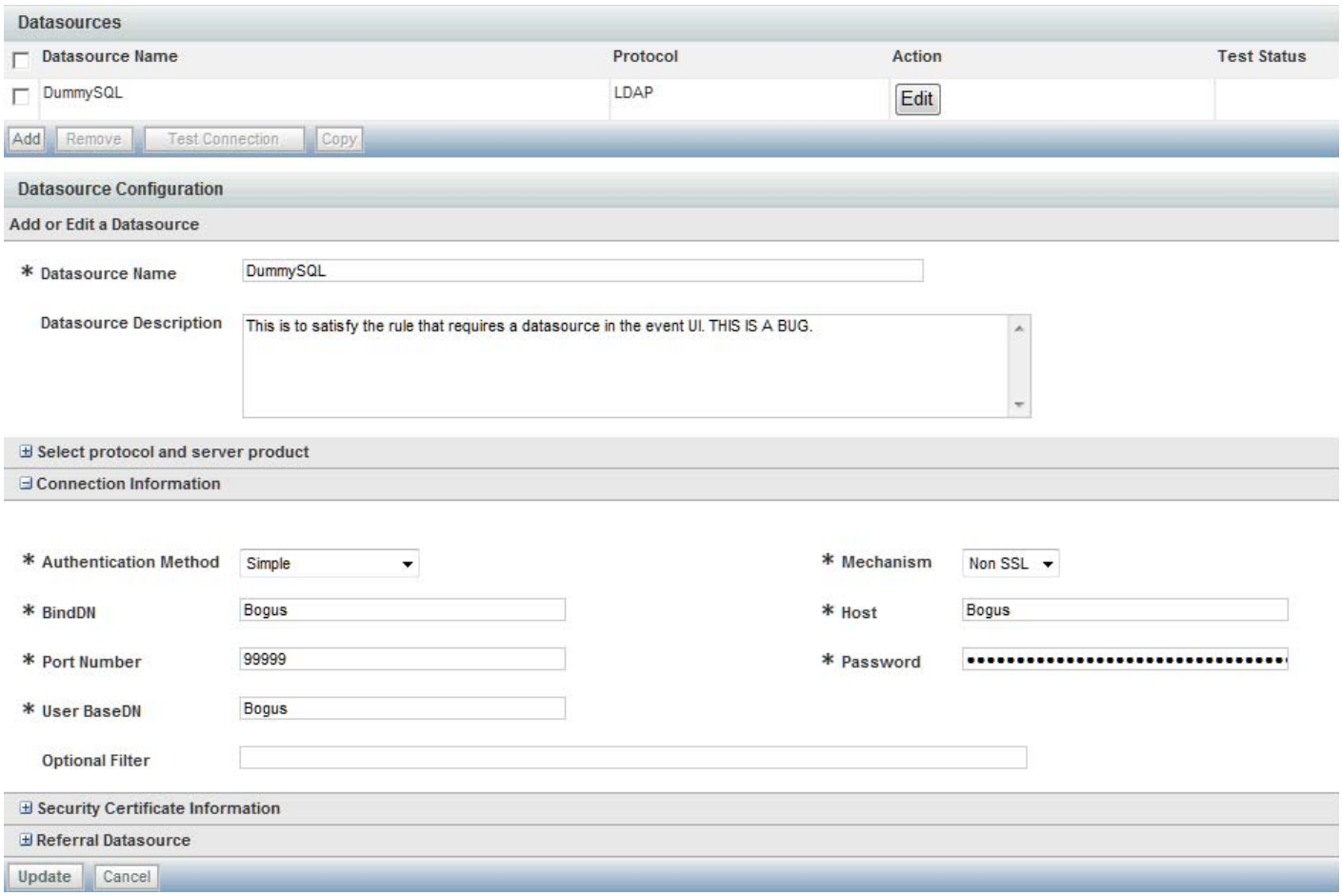
The following is some sample data to go with the above table definition:

```
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Moe', 'Moe', 'Howard', 'Moe', 'moe@stooge.com',
'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Larry', 'Larry', 'Fine', 'Larry',
'larry@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Curly', 'Curly', 'Howard', 'Curly',
'curly@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Shemp', 'Shemp', 'Howard', 'Shemp',
'shemp@stooge.com', 'Nyuk Nyuk Nyuk')
```

Datasource Definition

To use the Directory Integration interface you have to have an LDAP datasource configured. LDAP is the only UI supported datasource. You can create maps without a datasource, but you cannot test them without an LDAP datasource.

Figure 8-17 Sample Datasource Configuration

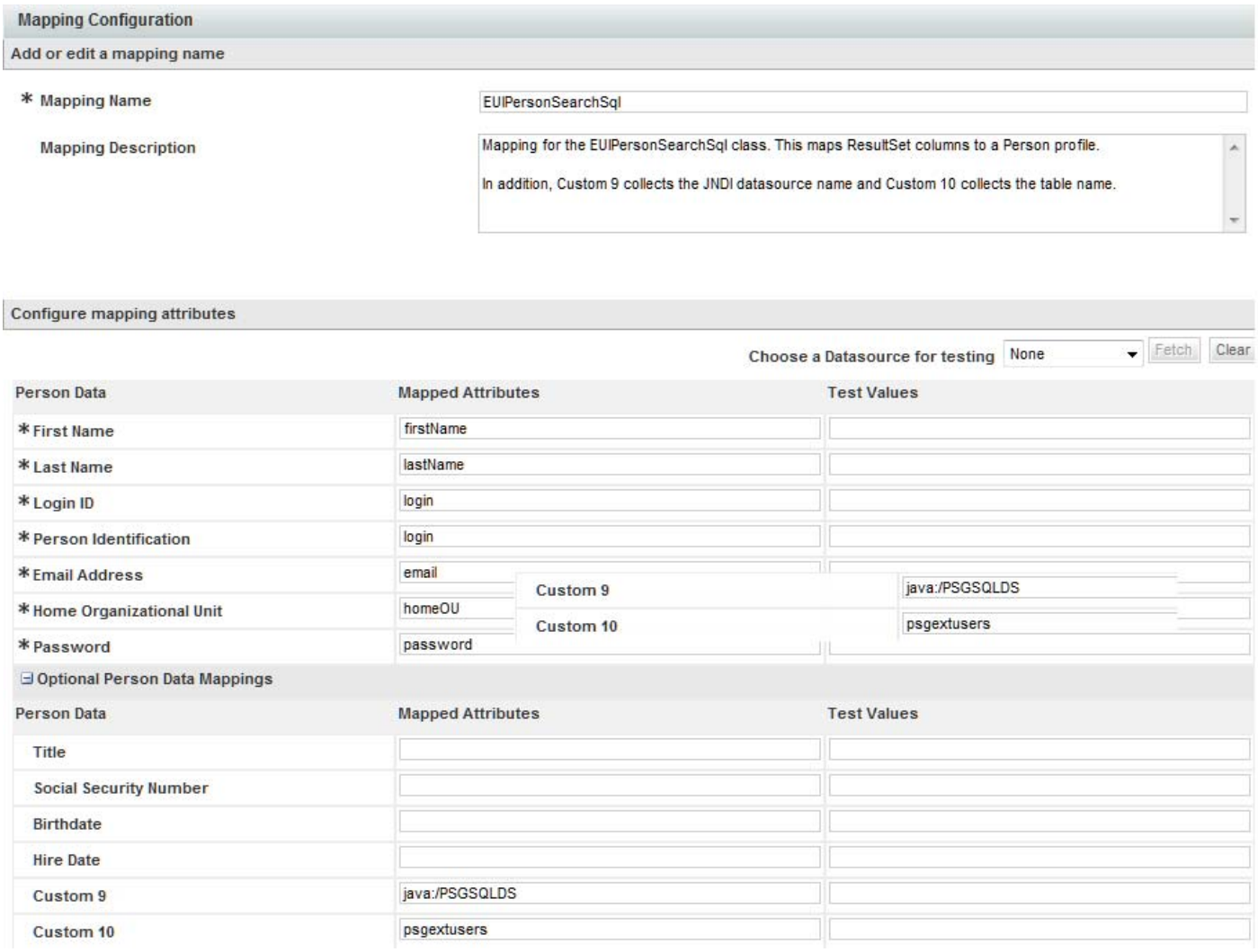


Configuring a container-managed datasource depends on the container. Detailed instructions on configuring datasources are given in the *Cisco Prime Service Catalog Installation and Upgrade Guide*.

Sample Mapping

A mapping must be created for the EUIPersonSearchSQL class.

Figure 8-18 Sample Mapping Configuration

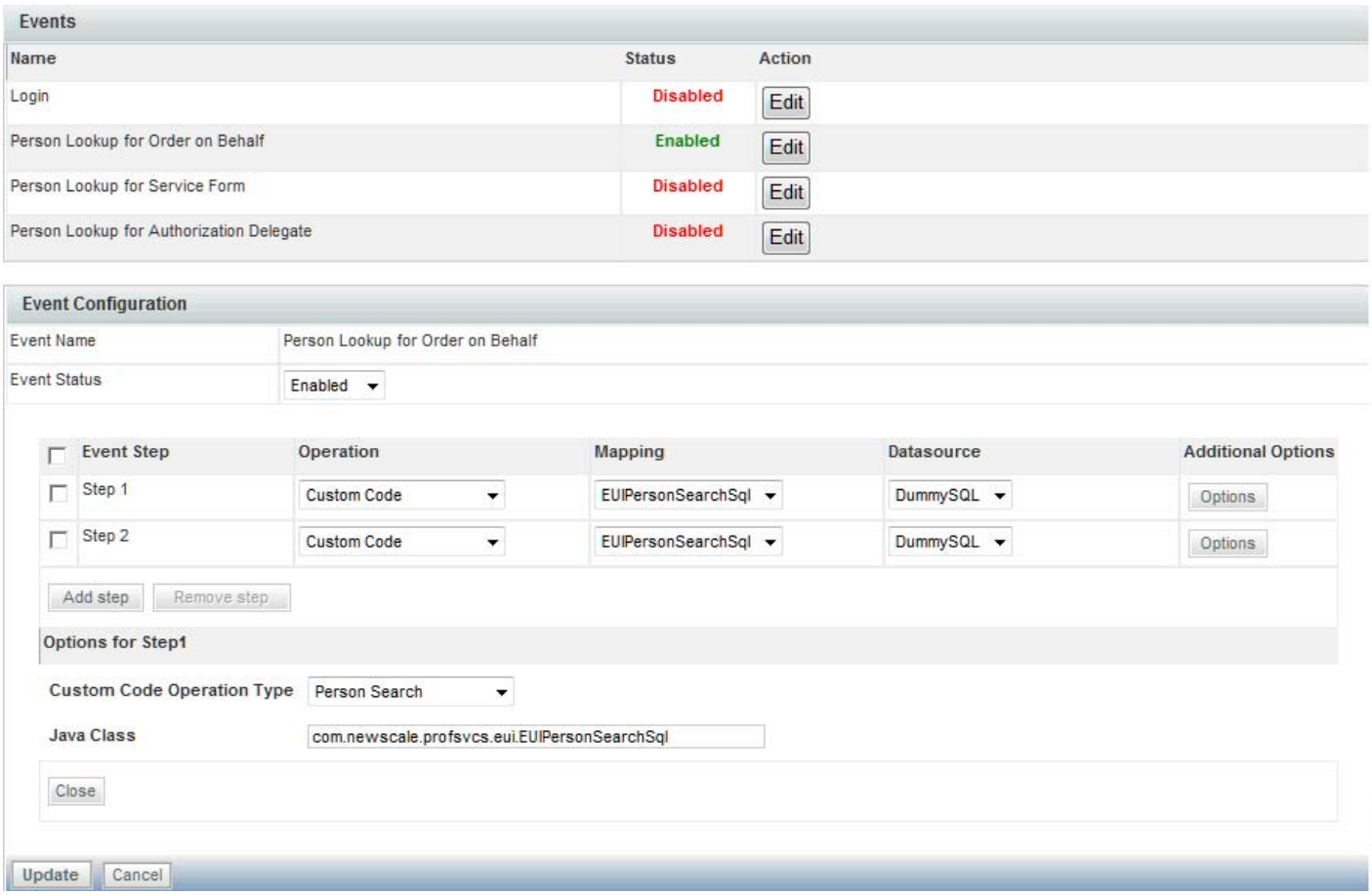


This mapping includes references to the JNDI as Custom 9 and the table name for Custom 10. Using a mapping like this, it is possible to do a simple query such as “select * from tablename” and use the metadata functionality in JDBC to select the column based on the mapping.

Sample Event Configuration

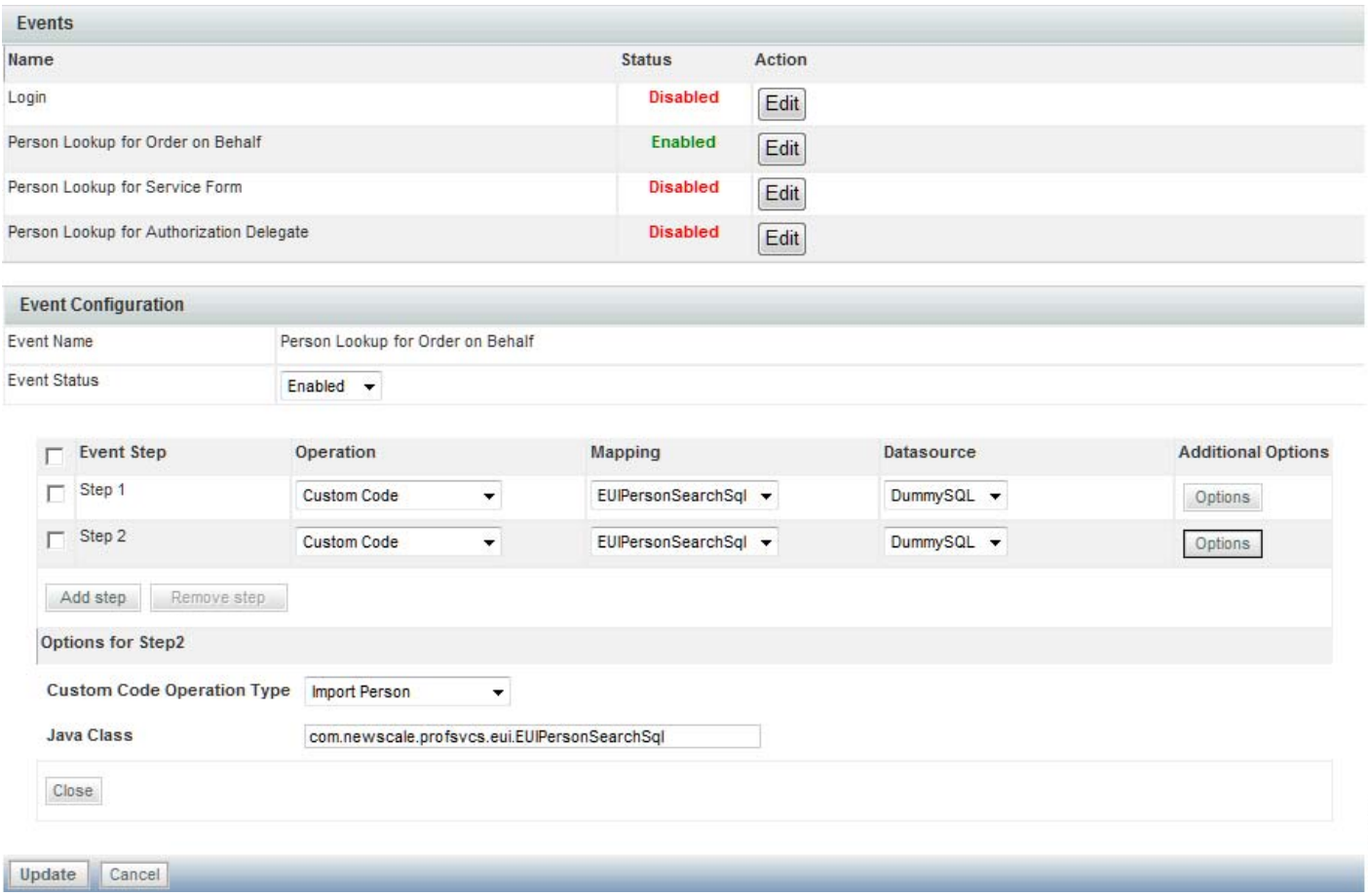
The “Person Lookup for Order on Behalf” event has two steps: The first must perform a “Person Search” operation. The name of the class is given as the mapping. The complete package specification is given as the Java class.

Figure 8-19 Custom Person Search Operation



The second step in the "Person Lookup for Order on Behalf" event is to import the selected person ("Import Person"). This configuration uses the same Java class, but a different Custom Code Operation Type. The Custom Code Operation Types in the drop-down menu correspond to the methods that are called in the interface class.

Figure 8-20 Event Step 2 – Custom Import Person Operation



Sample Code for SQL-Based Person Lookup

The following is the source for the custom class:

```
package com.newscale.profsvcs.eui;

import com.newscale.api.person.*;
import com.newscale.bfw.eui.EUIException;
import com.newscale.bfw.eui.api.*;
import com.newscale.bfw.ldap.ILDAPApi;
import com.newscale.bfw.logging.ILogUtil;
import com.newscale.bfw.logging.LogUtilFactory;
import com.newscale.comps.extuserintegration.session.*;

import javax.servlet.http.HttpServletRequest;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.*;

/**
 * Person Search to an external SQL datasource
```

```
 *
 * @author Lee Weisz
 * @version $Revision$
 */
public class EUIPersonSearchSql implements IPersonSearch {
  /**
   * Logger instance
   */
  private ILogUtil log = LogUtilFactory.getLogUtil(EUIPersonSearchSql.class);

  /**
   * Implement Person Search Operation and fetch users from an external system
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult search(IEUIEventPersonSearchOperationDTO
euiOperationDTO,
                                                IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                HttpServletRequest request,
                                                ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {

    log.debug("search: Entering search method...");
    IEUIPersonSearchOperationResult euiOperationResult = euiPersonSearchOperationContext
        .getEUIPersonSearchOperationResult();

    // Check if there is any SearchPerson List already available, if so we
    // can append to the existing List

    // Typically if there is a productized Person Search Operation is
    // configured before the custom code, this list would be populated

    // TODO Why is this an ArrayList? Can't it be a List?
    ArrayList personList = euiOperationResult.getSearchPersonList();

    if (null == personList) {
      personList = new ArrayList();
    }

    // Get the search criteria from the dialog box
    String searchFirstName = euiPersonSearchOperationContext.getFirstNameSearchString();
    String searchLastName = euiPersonSearchOperationContext.getLastNameSearchString();

    log.debug("search: Looking for " + searchFirstName + " " + searchLastName);

    EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
    Map attributeMap = dataMappingDTO.getAllAttributeMap();

    // What's in this map?
    if (log.isDebugEnabled()) {
      Set ks = attributeMap.keySet();
      for (Iterator it = ks.iterator(); it.hasNext();) {
        Object key = it.next();
        log.debug("search: " + key + " is " + attributeMap.get(key));
      }
```

```
    }

    // Use the map to map the columns to Person fields
    String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
    String lastNameColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
    String loginColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);

    // Use the custom9 mapping to hold the datasource value and custom10 to
    // hold the tablename. Since we control the import as well, it won't show up
    // in the imported Person's profile
    String ds = (String) attributeMap.get("custom9");
    String sourceTable = (String) attributeMap.get("custom10");


    StringBuffer searchSQL;
    searchSQL = new StringBuffer().append("select ")
        .append(firstNameColumn).append(", ")
        .append(lastNameColumn).append(", ")
        .append(loginColumn).append(" from ")
        .append(sourceTable);

    if (searchFirstName != null && searchFirstName.trim().length() > 0 ||
        searchLastName != null && searchLastName.trim().length() > 0) {
      searchSQL.append(" where ");

      if (searchFirstName != null && searchFirstName.trim().length() > 0) {
        searchSQL.append(firstNameColumn).append(" like
'").append(searchFirstName.trim()).append("%'");
      }

      if (searchFirstName != null && searchFirstName.trim().length() > 0 &&
          searchLastName != null && searchLastName.trim().length() > 0) {
        searchSQL.append(" and ");
      }

      if (searchLastName != null && searchLastName.trim().length() > 0) {
        searchSQL.append(lastNameColumn).append(" like
'").append(searchLastName.trim()).append("%'");
      }
    }

    log.debug("search: " + searchSQL.toString());

    Connection conn = null;
    Statement s = null;

    // get a connection to the external db
    try {
      conn = signOnImportPersonAPI.getExternalDBConnection(ds);

      s = conn.createStatement();
      ResultSet rs = s.executeQuery(searchSQL.toString());

      while (rs.next()) {
        String fname = rs.getString(firstNameColumn);
        String lname = rs.getString(lastNameColumn);
        String login = rs.getString(loginColumn);

        IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
        IPersonDTO personDTO = PersonFactory.createPersonDTO();
```

```
            personDTO.setFirstName(fname);
            personDTO.setLastName(lname);
            personDTO.setPersonIdentification(login);

            // Make the IPersonDTO into an IExtPersonDTO
            extUserDTO.setPersonDTO(personDTO);
            // Add IExtUserDTO to the collection of searched persons
            personList.add(extUserDTO);
          }
        } catch (SQLException e) {
          log.error("search: " + searchSQL.toString(), e);
        } catch (SignOnImportPersonAPIException e) {
          log.error("search: Cannot get a connection to " + ds, e);
        } finally {
          try {
            s.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
          try {
            conn.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }

        // Set the list of Persons Searched into the Result to be returned
        euiOperationResult.setSearchPersonList(personList);

        log.debug("search: Leaving search method...");
        return euiOperationResult;
      }

      /**
       * Implement the Import Person Operation to Import a user from External
       * system
       *
       * @param euiOperationDTO       .
       * @param euiPersonSearchOperationContext
       *                              .
       * @param request               .
       * @param signOnImportPersonAPI
       * @param ldapApi
       * @return .
       * @throws EUIException .
       */
      public IEUIPersonSearchOperationResult importPerson(IEUIEventImportPersonOperationDTO
euiOperationDTO,
                                                          IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                          HttpServletRequest request,
                                                          ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
          throws EUIException {

        log.debug("importPerson: Entering importPerson method...");

        /* Potentially useful stuff on the request...
          Name : isOOB          Value : true/false
          Name : customerid     Value : personDTO.setPersonIdentification() from search
          Name : customerId     Value : personDTO.setPersonIdentification() from search
          Name : lDAPCustomerId Value : personDTO.setPersonIdentification() from search
        */
```

```
    // What's on this request?
    if (log.isDebugEnabled()) {
      log.debug("importPerson: Parameters collected from the search window...");
      Enumeration paramNames = request.getParameterNames();

      if (paramNames.hasMoreElements()) {
        while (paramNames.hasMoreElements()) {
          String paramName = (String) paramNames.nextElement();
          String paramValues[] = request.getParameterValues(paramName);
          if (paramValues != null) {
            log.debug("importPerson: Name : " + paramName);
            for (int i = 0; i < paramValues.length; i++) {
              log.debug("importPerson: Value : " + paramValues[i]);
            }
          }
        }
      }
    }

    boolean refreshPerson = true;
    String login = request.getParameter("customerId");

    // Defaults
    String homeOU    = "";
    String firstName = "";
    String lastName  = "";
    String email     = "";
    String password  = "password";

    // Get the UI mapping
    EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
    Map attributeMap = dataMappingDTO.getAllAttributeMap();

    // Use the map to map the columns to Person fields
    String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
    String lastNameColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
    String loginColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);
    String passwordColumn  = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_PASSWORD);
    String emailColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_EMAILADDRESS);
    String homeOUColumn    = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_HOMEORGANIZATIONALUNIT);

    // Use the custom9 mapping to hold the datasource value and custom10 to
    // hold the tablename. Since we control the import as well, it won't show up
    // in the imported Person's profile unless we screw up somehow and put it there...
    String ds = (String) attributeMap.get("custom9");
    String sourceTable = (String) attributeMap.get("custom10");

    StringBuffer importSQL;
    importSQL = new StringBuffer().append("select ")
        .append(firstNameColumn).append(", ")
        .append(lastNameColumn).append(", ")
        .append(loginColumn).append(", ")
        .append(passwordColumn).append(", ")
        .append(emailColumn).append(", ")
        .append(homeOUColumn)
        .append(" from ").append(sourceTable).append(" where ")
        .append(loginColumn).append("='").append(login).append("'");
```

```
    log.debug("import: " + importSQL.toString());

    Connection conn = null;
    Statement s = null;

    try {
      // get a connection to the external db
      conn = signOnImportPersonAPI.getExternalDBConnection(ds);
      s = conn.createStatement();
      ResultSet rs = s.executeQuery(importSQL.toString());

      while (rs.next()) {
        homeOU = rs.getString(homeOUColumn);
        firstName = rs.getString(firstNameColumn);
        lastName = rs.getString(lastNameColumn);
        email = rs.getString(emailColumn);
        password = rs.getString(passwordColumn);
      }
    } catch (SQLException e) {
      log.error("import: " + importSQL.toString(), e);
    } catch (SignOnImportPersonAPIException e) {
      log.error("import: Cannot get a connection to " + ds, e);
    } finally {
      try {
        s.close();
      } catch (SQLException e) {
        log.error("import: ", e);
      }
      try {
        conn.close();
      } catch (SQLException e) {
        log.error("import: ", e);
      }
    }

    log.debug("import : Got " + login + "," + firstName + "," + lastName + "," + email +
"," + password + "," + homeOU);

    IPersonDTO personDTO = PersonFactory.createPersonDTO();
    try {
      // Get or Create the Person
      // This API throws an exception if the Person is not found in Service Catalog
      try {
        personDTO = signOnImportPersonAPI.getPersonByLoginName(login);
        log.info("importPerson: " + login + " exists in Request Center");
      } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new Person for " + login);
        refreshPerson = false;
        personDTO.setLogin(login);
      }

      // Get or Create the Home OU that the Person should be associated with
      // This API throws an exception if the OU is not found in Service Catalog
      IOrganizationalUnitDTO homeOUDTO;
      try {
        homeOUDTO = signOnImportPersonAPI.getOrgUnitByName(homeOU);
        log.info("importPerson: " + homeOU + " exists in Request Center");
      } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new OU " + homeOU + " for " + login);
        homeOUDTO = PersonFactory.createOrganizationalUnitDTO();
        homeOUDTO.setName(homeOU);
        homeOUDTO.setBillable(false);
        homeOUDTO.setOrganizationalUnitTypeId(2); // business unit.
        homeOUDTO.setRecordStateId(1); // active
```

```
        homeOUDTO.setLocaleId(EUIAPIConstants.LOCALEID.USEN);
        try {
          homeOUDTO = signOnImportPersonAPI.createOrgUnit(homeOUDTO);
        } catch (SignOnImportPersonAPIException crEx) {
          log.error("importPerson: Can't create " + homeOU + " for " + login);
          throw crEx;
        }
      }
      personDTO.setHomeOrganizationalUnitId(homeOUDTO.getId());

      // Populate the Login Object...
      // Modify the login information only if this is a new Person
      if (!refreshPerson) {
        ILoginInfoDTO loginInfoDTO = PersonFactory.createLoginInfoDTO();

        loginInfoDTO.setLoginname(personDTO.getLogin());
        loginInfoDTO.setPrivateKey(personDTO.getLogin());
        // Set the un-encrypted password
        loginInfoDTO.setPassword(password);

        // Set ILoginInfoDTO to IPersonDTO
        personDTO.setILoginInfoDTO(loginInfoDTO);
      }

      // Populate the rest of the essential fields
// Presumably, any expression on the mapping will have already been executed
// and the result is what's returned in the personDTO
      personDTO.setFirstName(firstName);
      personDTO.setLastName(lastName);
      personDTO.setEmail(email);

      // Set the active status
      // TODO These methods are bogus...
//    personDTO.setIsInactive(false);
//    personDTO.setIsActive(true);
      // TODO What do these numbers mean? Is there a constants library to convert these
codes into something meaningful?
      personDTO.setRecordStateId(1);

      // Upsert the Person
      signOnImportPersonAPI.beginTransaction();
      if (refreshPerson) {
        // Update the existing Person
        // This method updates only Basic Info, LoginInfo, Preferences, Home OU and Person
Extension
        signOnImportPersonAPI.updatePerson(personDTO);
      } else {
        // Create the Person
        // This creates a Person with Basic Info, LoginInfo, Preferences, Home OU and
Person Extension
        personDTO = signOnImportPersonAPI.createPerson(personDTO);
        // From here on out it's a refresh
        refreshPerson = true;
      }
      signOnImportPersonAPI.commitTransaction();
    } catch (Exception e) {
      log.error("importPerson: Exception during Import Person", e);
      try {
        // Rollback Transaction
        signOnImportPersonAPI.rollbackTransaction();
      } catch (SignOnImportPersonAPIException se) {
        log.error("importPerson: Error while Rolling back transaction", se);
      }
    } finally {
```

```
      // Release Transaction
      signOnImportPersonAPI.releaseTransaction();
    }

    IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
    extUserDTO.setPersonDTO(personDTO);

    IEUIPersonSearchOperationResult psor =
euiPersonSearchOperationContext.getEUIPersonSearchOperationResult();
    psor.setImportedPersonExtDTO(extUserDTO);

    log.debug("importPerson: Leaving importPerson method...");

    return psor;
  }

  /**
   * Implement Import Manager Operation and Import all the Supervisors chain
   * of the Person being imported
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *                              .
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult importManager(IEUIEventImportManagerOperationDTO
euiOperationDTO,
                                                       IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                       HttpServletRequest request,
                                                       ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {
    return null;
  }

  /**
   * Implement any Custom Operation
   *
   * @param euiOperationDTO       .
   * @param euiPersonSearchOperationContext
   *
   * @param request               .
   * @param signOnImportPersonAPI
   * @param ldapApi
   * @return .
   * @throws EUIException .
   */
  public IEUIPersonSearchOperationResult performCustom(IEUIEventCustomOperationDTO
euiOperationDTO,
                                                       IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                       HttpServletRequest request,
                                                       ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
      throws EUIException {
    return null;
  }
}
```

Supported Time Zones

The supported time zone values when mapping time zones are listed below.

Table 8-18 Supported Time Zones

Time Zone Name	GMT Equivalent
Etc/GMT+12	(GMT-12:00) International Date Line West
Pacific/Apia	(GMT-11:00) Samoa
US/Hawaii	(GMT-10:00) Hawaii
US/Aleutian	(GMT-10:00) Hawaii Aleutian Daylight Time
US/Alaska	(GMT-09:00) Alaska
America/Tijuana	(GMT-08:00) Pacific Time (US and Canada); Tijuana
America/Chihuahua	(GMT-07:00) Chihuahua, La Paz, Mazatlan
US/Arizona	(GMT-07:00) Arizona
Canada/Mountain	(GMT-07:00) Mountain Time (US and Canada)
Canada/Saskatchewan	(GMT-06:00) Saskatchewan
US/Central	(GMT-06:00) Central America
Canada/Central	(GMT-06:00) Central Time (US and Canada)
America/Mexico_City	(GMT-06:00) Guadalajara, Mexico City, Monterrey
America/Bogota	(GMT-05:00) Bogota, Lima, Quito
Canada/Eastern	(GMT-05:00) Eastern Daylight Time (US and Canada)
America/Jamaica	(GMT-05:00) Eastern Time (US and Canada)
US/East-Indiana	(GMT-05:00) Indiana (East)
America/Antigua	(GMT-04:00) Atlantic Time (Canada)
Canada/Atlantic	(GMT-04:00) Atlantic Daylight Time (Canada)
America/Manaus	(GMT-04:00) Manaus
America/Santiago	(GMT-04:00) Santiago
America/Caracas	(GMT-04:30) Caracas
America/La_Paz	(GMT-04:00) La Paz (Bolivia)
America/Sao_Paulo	(GMT-03:00) Brasilia
America/Godthab	(GMT-03:00) Greenland
America/Argentina/Buenos_Aires	(GMT-03:00) Buenos Aires
America/Guyana	(GMT-04:00) Georgetown
America/St_Johns	(GMT-03:30) Newfoundland and Labrador
Atlantic/South_Georgia	(GMT-02:00) Mid-Atlantic
Atlantic/Azores	(GMT-01:00) Azores
Atlantic/Cape_Verde	(GMT-01:00) Cape Verde Islands
Etc/Greenwich	(GMT) Greenwich Mean Time: Dublin, Edinburgh,
Africa/Casablanca	(GMT) Casablanca, Monrovia
Europe/Sarajevo	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
Europe/Brussels	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
Africa/Brazzaville	(GMT+01:00) West Central Africa
Europe/Amsterdam	(GMT+01:00) Amsterdam, Berlin, Bern, Rome,

Table 8-18 Supported Time Zones

Time Zone Name	GMT Equivalent
Europe/Belgrade	(GMT+01:00) Belgrade, Bratislava, Budapest,
Africa/Cairo	(GMT+02:00) Cairo
Europe/Helsinki	(GMT+02:00) Helsinki, Kiev, Riga, Sofia, Tallinn,
Europe/Minsk	(GMT+02:00) Minsk
Europe/Athens	(GMT+02:00) Athens, Bucharest, Istanbul
Asia/Jerusalem	(GMT+02:00) Jerusalem
Africa/Windhoek	(GMT+02:00) Windhoek
Africa/Harare	(GMT+02:00) Harare, Pretoria
Asia/Baghdad	(GMT+03:00) Baghdad
Africa/Nairob	(GMT+03:00) Nairobi
Europe/Moscow	(GMT+03:00) Moscow, St. Petersburg, Volgograd
Asia/Kuwait	(GMT+03:00) Kuwait, Riyadh
Asia/Tehran	(GMT+03:30) Tehran
Asia/Baku	(GMT+04:00) Baku
Asia/Muscat	(GMT+04:00) Abu Dhabi, Muscat
Asia/Yerevan	(GMT+04:00) Yerevan
Asia/Tbilisi	(GMT+04:00) Tbilisi
Asia/Kabul	(GMT+04:30) Kabul
Asia/Karachi	(GMT+05:00) Islamabad, Karachi, Tashkent
Asia/Yekaterinburg	(GMT+05:00) Ekaterinburg
Asia/Kolkata	(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
Asia/Kathmandu	(GMT+05:45) Kathmandu
Asia/Dhaka	(GMT+06:00) Astana, Dhaka
Asia/Novosibirsk	(GMT+07:00) Novosibirsk
Asia/Colombo	(GMT+05:30) Sri Jayawardenepura
Asia/Rangoon	(GMT+06:30) Yangon (Rangoon)
Asia/Bangkok	(GMT+07:00) Bangkok, Hanoi, Jakarta
Asia/Krasnoyarsk	(GMT+08:00) Krasnoyarsk
Asia/Irkutsk	(GMT+09:00) Irkutsk
Asia/Kuala_Lumpur	(GMT+08:00) Kuala Lumpur, Singapore
Asia/Taipei	(GMT+08:00) Taipei
Australia/Perth	(GMT+08:00) Perth
Asia/Chongqing	(GMT+08:00) Beijing, Chongqing, Hong Kong SAR,
Asia/Seoul	(GMT+09:00) Seoul
Asia/Tokyo	(GMT+09:00) Osaka, Sapporo, Tokyo
Asia/Yakutsk	(GMT+09:00) Yakutsk
Australia/Darwin	(GMT+09:30) Darwin
Australia/Adelaide	(GMT+09:30) Adelaide
Australia/Hobart	(GMT+10:00) Hobart
Australia/Canberra	(GMT+10:00) Canberra, Melbourne, Sydney

Table 8-18 Supported Time Zones

Time Zone Name	GMT Equivalent
Australia/Brisbane	(GMT+10:00) Brisbane
Asia/Vladivostok	(GMT+10:00) Vladivostok
Pacific/Guam	(GMT+10:00) Guam, Port Moresby
Pacific/Guadalcanal	(GMT+11:00) Solomon Islands, New Caledonia
Pacific/Auckland	(GMT+12:00) Auckland, Wellington
Pacific/Fiji	(GMT+12:00) Fiji Islands
Pacific/Tongatapu	(GMT+13:00) Nuku alofa

Sample build.xml File

```
<?xml version="1.0" ?>
<project name="Sample Project" default="all" basedir=".">
- <!-- Main target -->
  <target name="all" depends="init,build,deploy" />
  <!- Set the following properties to point to appropriate folders -->
  <property name="rcwar.dir" value="<apps server path where Request Center application WAR
file is deployed>" />
  <property name=" javax.servlet.dir" value="<path where
jboss-servlet-api_3.0_spec-1.0.0.Final.jar is available in the app server>" />
  <property name="rcwar_webinf_classes.dir"value="${rcwar.dir}/WEB-INF/classes" />
  <target name="init">
    <property name="dirs.base" value="${basedir}" />
    <mkdir dir="${dirs.base}/out" />
    <property name="src" value="${dirs.base}/src" />
    <property name="out" value="${dirs.base}/out" />
  </target>
  <path id="classpath">
    <fileset dir="${rcwar.dir}" includes="*.jar" />
    <fileset dir="${javax.servlet.dir}"
includes="jboss-servlet-api_3.0_spec-1.0.0.Final.jar" />
    <pathelement path="${rcwar_webinf_classes.dir}" />
  </path>
- <!-- Compile Java Files -->
  <target name="build" depends="init">
    <javac srcdir="${src}" destdir="${out}" debug="true" includes="**/*.java"
classpathref="classpath" deprecation="true" fork="true" memoryinitialsize="256M"
memorymaximumsize="512M" />
  </target>
  <target name="deploy" depends="init">
    <copy todir="${rcwar_webinf_classes.dir}">
      <fileset dir="${out}">
        <include name="**/*.class" />
      </fileset>
    </copy>
  </target>
</project>
```

Configuring SSO Using SAML

The Security Assertion Markup Language (SAML) is an XML based open standard data format for exchanging authentication and authorization information data between parties. SAML is implemented for Prime Service Catalog so that any other application integrating with Prime Service Catalog can use SAML as a means to provide Authentication and import person profile information from IDP.

There are three key elements in SAML:

- **User**—The client that is attempting to log-in to a service provider (Cisco Prime Service Catalog).
- **Identity Provider (IDP)**—Typically a portal where the user logs in, it has the authority on a user's identity. It knows the user's username, password, and any groups/attributes.

Note The Prime Service Catalog supports only one IDP connection to authenticate a user at login.

- **Service Provider (SP)**—The application the user wishes to use. In this case, Cisco Prime Service Catalog.

Caution You cannot configure both LDAP and SAML configured for SSO login in Prime Service Catalog. If you wish to use SAML SSO, the LDAP Login event must be manually disabled, failing which will lead to incorrect login behavior.
To disable LDAP login, go to **Administration > Directories > Events** and click **Edit** for the Login event. Change the event status to Disabled and click **Update**.

Log In Behavior

Implementing single sign-on via SAML means that the sign in process and user authentication are handled entirely outside of Prime Service Catalog. Prime Service Catalog uses SAML as means of securely authenticating against an IDP; authorization is provided by Prime Service catalog. With SAML configured in a system, the user must first authenticate with the IDP. On successful authentication the user is imported into Prime Service Catalog, if the user does not exist and is redirected to PSC, they will be granted access only if they have a valid permission and the IDP is correctly configured. On the same browser the user sessions are maintained.

Log Out Behavior

Log out behaviors are different based on the **saml.enable.globalLogout** property settings made in *newscale.properties* file, see section Properties for SAML Configuration, page 9-2.

By default global logout is enabled. In this case, when the user logs out of one instance of Prime Service Catalog the user is also logged out of other instance on the same browser.

With global logout disabled, when the user logs out of Prime Service Catalog or other applications integrated with Prime Service Catalog, SAML logs the user out only from that particular application. This is called local logout.

The below table describes the various logout behavior when the global logout is set on two SPs on the same browser. Here SP1 and SP2 are two instances of Prime Service Catalog.

Use Case	Global Logout Setting on SP1	Global Logout Setting on SP2	Logout Behavior
1	True	True	Both SP1 and SP2 would be logged out, if either of the SP is logged out.
2	True	False	• If SP1 logout, SP2 will also be logged out. • If SP2 logout, SP1 will not be logged out.
3	False	True	• If SP1 logout, SP2 will not be logged out. • If SP2 logout, SP1 will also be logged out.
4	False	False	If either SP1 or SP2 logout, the other SP is not logged out.

User Management in SAML

After you have enabled SAML all the user management and authentication is handled outside of Prime Service Catalog. However, changes made outside of Prime Service Catalog are immediately synced back to Prime Service Catalog. User information is imported on first attempt at authentication against an IDP and every time user logs in to Prime Service Catalog, SAML refreshes the user data and syncs from IDP to Prime Service Catalog. If you delete a user in the system, the user will no longer be able to sign in to Prime Service Catalog (though their account will still exist in Prime Service Catalog).

Unlike LDAP, SAML does not support person search. However, if the IDP uses LDAP for user management, any changes to the user will be synced to Prime Service Catalog database. The admin must have the credentials for that LDAP connection so as to configure it for Person lookup OOB, Authorization delegate, Person Lookup Service form, and the Import person event.

Properties for SAML Configuration

Below table describes the configuration settings in *newscale.properties* that allows you to configure SAML for your system.

Property	Description
saml.lb.protocol	Set to 'http' or 'https' for LB.
saml.lb.hostname	Set to the exposed RC endpoint Ensure it is not loop back address (127.0.0.1 or localhost). If LB or Reverse proxy is used this will be the exposed endpoint's IP or domain name.
saml.lb.port	Set to the appropriate port number.
saml.lb.config.includeServerPortInRequestURL	Set to true or false. If set to *true* the port will be used for validating request/response during SAML exchanges between SP and IDP.
saml.matadata.refreshInterval	Set the time interval for the matadata refresh.
saml.provider.trustCheck	Sets the validation of signature trust for all providers.
saml.force.auth	Sets whether the user must authenticate even if the session is valid.
saml.enable.global.logout	Sets whether global logout is enabled or disabled. By default, it is set to true.
saml.certificate.validation.config	Sets the certificate validation configurations. For more information, see SAML Certificate Validation Settings, page 9-3.

SAML Certificate Validation Settings

This section provides information on the validation settings provided in Prime Service Catalog for SAML Certificates while configuring the SAML certificate validation.

Under SAML specifications, when you receive messages, the messages must be digitally signed. Signing is always required for SAML. You can validate the SAML certificate by setting the following properties:

Property	Description
checkFQDNValidity	When set to true, it checks the fully qualified domain name or the common name in the certificate.
allowSelfSignedCertificates	When set to true, allows the Self-Signed certificates.
allowOnlyRootCertificates	When set to true, allows only the root certificates. Default is false. **Note** If you set allowOnlyRootCertificates to true, it allows all the Self-Signed certificates even if allowSelfSignedCerticates is set to false. As all root certificates are self-signed.
checkValidity	When set to true, checks the validity period of the certificate.
checkMaxExpiryDays	When set to true, checks the maximum period of the certificate validity.
checkCertificateRevocation	When set to true, checks the dynamic certificate revocation list in the certificate.
checkTrust	When set to true, it validates the certificate from the trust chain.

Configuring SAML Settings and IDP Mapping

For detailed information on configuring the SAML settings and Mapping the IDP with Prime Service Catalog, see the *SAML Configuration* section in Cisco Prime Service Catalog Administration and Operation Guide.

SAML REST APIs

The SAML nsAPIs can be accessed only by the Site Administrator and users having SAML Configuration capability. The nsAPI authentication for SAML Configurations and IDP Mappings uses RC DB even when SAML is enabled. So the user needs to use their RC DB credentials.

The response messages for a successfully submitted order is 200.

For information on the error response messages, *see* REST/Web Services Error Messages table and Error Messages.

Table 9-1 SAML REST API Table

Area	Examples
DELETE	**Delete an IDP Configuration** URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name> To delete an IDP Configuration, enter the unique name of the IDP.
GET	**Get an IDP Configuration** URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs/<idp configuration name> To get an IDP configuration, enter the unique name of the IDP.
PUT	**Refresh metadata(s) on node** URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/refreshThis

Table 9-1 SAML REST API Table

Area	Examples

POST	**Save an IDP Configuration**
	URL: http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs
	Sample Input:
	{
	"name": "ssocirclef631a5967b044cec94893ac700851de3",
	"metadata": "<?xml version=\"1.0\"
	encoding=\"UTF-8\"?>\r\n<md:EntityDescriptor
	\r\n\txmlns:md=\"urn:oasis:names:tc:SAML:2.0:metadata\"
	\r\n\tentityID=\"https://auth.miniorange.com/moas\">\r\n\r\n\t<md:IDPSSODescr
	iptor \r\n\t\tWantAuthnRequestsSigned=\"true\"
	\r\n\t\tprotocolSupportEnumeration=\"urn:oasis:names:tc:SAML:2.0:protocol\">\
	r\n\r\n\t\t<md:KeyDescriptor \r\n\t\t\tuse=\"signing\">\r\n\t\t\t<ds:KeyInfo
	\r\n\t\t\t\txmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\">\r\n\t\t\t\t<ds:X
	509Data>\r\n\t\t\t\t\t<ds:X509Certificate>MIICnjCCAgegAwIBAgIJAK3CyOFtrUj
	MA0GCSqGSIb3DQEBBQUAMGgxCzAJBgNVBAYTAklOMQswCQYDVQQIDAJNSDENMAsGA1UEBwwEUFVOR
	TETMBEGA1UECgwKbWluaU9yYW5nZTETMBEGA1UECwwKbWluaU9yYW5nZTETMBEGA1UEAwwKbWluaU
	9yYW5nZTAeFw0xNTAyMTEwNDQ1NDdaFw0xODAyMTAwNDQ1NDdaMGgxCzAJBgNVBAYTAklOMQswCQY
	DVQQIDAJNSDENMAsGA1UEBwwEUFVORTETMBEGA1UECgwKbWluaU9yYW5nZTETMBEGA1UECwwKbWlu
	aU9yYW5nZTETMBEGA1UEAwwKbWluaU9yYW5nZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAx
	WMW0HNXVL4VB14PklXDo6rJlK3W4XHsxD7rBsG8e2LgbfjEjC0b
	k2/5ODuP9OvVQyHaZhMPWbS2z5S6cxCIxPfAJC5pCn9EVVoSDbz4C1Biyg9NJAUYp7oF
	8JfKByLeWCOPRb9/G8/Bq5xQRAf CH/hSSsrNEQm5h
	NnhcCAwEAAaNQME4wHQYDVR0OBBYEFFq3KKnNFb1777slDNKfn30gXcvjMB8GA1UdIwQYMBaAFFq3
	KKnNFb1777slDNKfn30gXcvjMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEACP2t4JNkG
	h2ElJltQ3FSdWWHsvhGpGnpAdltdC8vW/Sf3a97IDeixr5GcQVfUfyYE nMQU0g2NJLYG1
	hb13J58eQ9NhU8PgkSsJWaskST1KTNRu
	30K3Dm8TOhZShWEvYBUzSDjcSJFUguXeoK/gx4wBuA8WEaKb9PC6xvac/4=</ds:X509Certifica
	te>\r\n\t\t\t\t</ds:X509Data>\r\n\t\t\t</ds:KeyInfo>\r\n\t\t</md:KeyDescripto
	r>\r\n\r\n\t\t<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emai
	lAddress</md:NameIDFormat>\r\n\t\t<md:NameIDFormat>urn:oasis:names:tc:SAML:1.
	1:nameid-format:unspecified</md:NameIDFormat>\r\n\t\t<md:SingleSignOnService
	\r\n\t\t\tBinding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST\"
	\r\n\t\t\tLocation=\"https://auth.miniorange.com/moas/idp/samlsso\"/>\r\n\t\t
	<md:SingleSignOnService
	\r\n\t\t\tBinding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\"
	\r\n\t\t\tLocation=\"https://auth.miniorange.com/moas/idp/samlsso\"/>\r\n\t</
	md:IDPSSODescriptor>\r\n</md:EntityDescriptor>",
	"attributesMapping": {
	"firstName": "FName",
	"lastName": "LName",
	"businessUnit": "bu",
	"localeCode": "locale",
	"costCenter": "costCenter",
	"organizationUnit": "Department",
	"login": "Email",
	"email": "Email"
	}
	}
	Note OrganizationalUnit, Locale, Business Unit and Cost Center are optional. You can ignore these values if you do not want to map these fields.

Table 9-1 SAML REST API Table

Area	Examples

PUT	**Update an IDP Configuration**
	Method: PUT
	URL:
	http://<ServerURL>/RequestCenter/nsapi/v1/idp/configs
	Sample Input:
	{
	"name": "idp1",
	"metadata": "<?xml version=\"1.0\"?>\n<EntityDescriptor xmlns=\"urn:oasis:names:tc:SAML:2.0:metadata\" entityID=\"https://app.onelogin.com/saml/metadata/655471\">\n <IDPSSODescriptor xmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\" protocolSupportEnumeration=\"urn:oasis:names:tc:SAML:2.0:protocol\">\n <KeyDescriptor use=\"signing\">\n <ds:KeyInfo xmlns:ds=\"http://www.w3.org/2000/09/xmldsig#\">\n <ds:X509Data>\n <ds:X509Certificate>MIIEIzCCAwugAwIBAgIUFZeKpXTlJF3kJ/lzmIGmeMuTUB8wDQYJKoZIhvcNAQEF\nBQAwXDELMAkGA1UEBhMCVVMxFDASBgNVBAoMC0Npc2NvX1ZpdmVrMRUwEwYDVQQL\nDAxPbmVMb2dpbiBJZFAxIDAeBgNVBAMMF09uZUxvZ2luIEFjY291bnQgMTA2MTg3\nMB4XDTE3MDUwNzEwMTUyOFoXDTIyMDUwODEwMTUyOFowXDELMAkGA1UEBhMCVVMx\nFDASBgNVBAoMC0Npc2NvX1ZpdmVrMRUwEwYDVQQLDAxPbmVMb2dpbiBJZFAxIDAe\nBgNVBAMMF09uZUxvZ2luIEFjY291bnQgMTA2MTg3MIIBIjANBgkqhkiG9w0BAQEF\nAAOCAQ8AMIIBCgKCAQEAvxPHxsSgjG3wl+xvgYNgAIdi9dDE6yJTAl63UoW67kgo\n1/BsY56Xd+Dul+LdyipUTGpU21v6zcNBXRGvGi5A7nGg5uUNpMwaGMTA0WJ0cegV\n/aRgYlUWL/01Vf+ep+f9B3ELpoUOMHA01+1OG2LX+WvlfHsPMMJD5v2lxY0uhATf\nDgLzTJox09LJfmadjtAuMgMa2D4Uf6NNWE5+DSXEv8aaZgsm/s8AkKrdUO++nNZD\nWcBbIn3BrkkMz4LIYIduw8kbONfEjaRnXdleYvJrgQkAGFzp2sGAqb+R2ll0Xm5H\neeG5Oe7naXjQHprlaQjLY/OYGk57d5JISuKN66Mw+wIDAQABo4HcMIHZMAwGA1Ud\nEwEB/wQCMAAwHQYDVR0OBBYEFN2xMXzEdCLhADEtFEE8l+9ykU2TMIGZBgNVHSME\ngZEwgY6AFN2xMXzEdCLhADEtFEE8l+9ykU2ToWCkXjBcMQswCQYDVQQGEwJVUzEU\nMBIGA1UECgwLQ2lzY29fVml2ZWsxFTATBgNVBAsMDE9uZUxvZ2luIElkUDEgMB4G\nA1UEAwwXT25lTG9naW4gQWNjb3VudCAxMDYxODeCFBWXiqV05SRd5Cf5c5iBpnjL\nk1AfMA4GA1UdDwEB/wQEAwIHgDANBgkqhkiG9w0BAQUFAAOCAQEAOipv7wvG0PDl\nnyjix4+XtAOHvhsjcnXOow5l9xfTKqOIawyz7Gy6NIoR17gvqBoVjmxMK1nTlNRS4\nhtQPMRzlX/ITt5rExDk4NT1cuhRaVd0rvgv6gvyj0gp/9Yjq+XH09JjR6Swc0Cgh\nd+2lyoGMIwogPI6ZmfikcsbdkCjvTAvKWodihBciavcdgt+QGKb2TAtGaOpQwGL6\nnNFESz6snUtI8LLAWsqQOpns24oZhEEUdZq7ozouTDDilZwRYJBuwzI0FhM86RPgg\njrKJ6FM9vODeCZQ7uLZWHCFCdB7hFX4uCyH9/v8xch3iT+mmzriJQyj1/DEgxqOf\nK7VuhD4SGQ==</ds:X509Certificate>\n </ds:X509Data>\n </ds:KeyInfo>\n </KeyDescriptor>\n <SingleLogoutService Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\" Location=\"https://cisco-vivek.onelogin.com/trust/saml2/http-redirect/slo/655471\"/>\n \n <NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>\n \n <SingleSignOnService Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect\" Location=\"https://cisco-vivek.onelogin.com/trust/saml2/http-redirect/sso/655471\"/>\n <SingleSignOnService Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST\" Location=\"https://cisco-vivek.onelogin.com/trust/saml2/http-post/sso/655471\"/>\n <SingleSignOnService Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:SOAP\" Location=\"https://cisco-vivek.onelogin.com/trust/saml2/soap/sso/655471\"/>\n </IDPSSODescriptor>\n</EntityDescriptor>",
	"attributesMapping": {
	"email": "User.email",
	"firstName": "User.FirstName",
	"lastName": "User.LastName",
	"login": "User.email",
	"organizationUnit": "department",
	"businessUnit": "User.LastName",

Table 9-1 SAML REST API Table

Area	Examples
	"localeCode": "User.LastName", "costCenter": "User.LastName", "title": "User.LastName", "socialsecuritynumber": "User.LastName", "birthdate": "User.LastName", "hiredate": "User.LastName", "timezoneid": "User.LastName", "employeecode": "User.LastName", "notes": "User.LastName", "companycode": "User.LastName", "division": "User.LastName", "departmentnumber": "User.LastName", "managementlevel": "User.LastName", "supervisorid": "User.FirstName", "region": "User.LastName", "employeetype": "User.LastName", "locationcode": "User.LastName", "custom1": "User.LastName", "custom2": "User.LastName", "custom3": "User.LastName", "custom4": "User.LastName", "custom5": "User.LastName", "custom6": "User.LastName", "custom7": "User.LastName", "custom8": "User.LastName", "custom9": "User.LastName", "custom10": "User.LastName", "companystreet1": "User.LastName", "companystreet2": "User.LastName", "companycity": "User.LastName", "companystate": "User.LastName", "companypostalcode": "User.LastName", "companycountry": "User.LastName", "officebuilding": "User.LastName", "buildinglevel": "User.LastName", "officelocation": "User.LastName", "cubiclocation": "User.LastName", "personalstreet1": "User.LastName", "personalstreet2": "User.LastName", "personalcity": "User.LastName", "personalstate": "User.LastName", "personalpostalcode": "User.LastName", "personalcountry": "User.LastName", "workphonenumber": "User.LastName",

Table 9-1 SAML REST API Table

Area	Examples

	<pre>"homephonenumber": "User.LastName",
	"faxnumber": "User.LastName",
	"mobilephonenumber": "User.LastName",
	"pagernumber": "User.LastName",
	"other": "User.LastName",
	"mainphonenumber": "User.LastName",
	"primaryphonenumber": "User.LastName",
	"primaryfaxnumber": "User.LastName",
	"salesphonenumber": "User.LastName",
	"supportphonenumber": "User.LastName",
	"billingphonenumber": "User.LastName",
	"othercontactinfo": "User.LastName",
	"ouList": "User.LastName::User.FirstName::User.email::department",
	"groupList": "User.LastName::User.FirstName::User.email::department",
	"roleList": "User.LastName::User.FirstName::User.email::department"
	}
	}</pre>
	Note The attributes `email`, `firstName`, `lastName`, `login`, and `organizationUnit` are mandatory inputs.

GET	**Get SAML Configuration** GET URL: http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs

Table 9-1 SAML REST API Table

Area	Examples

PUT	**Update SAML Configuration**
	PUT URL:
	http://<ServerURL>/RequestCenter/nsapi/v1/saml/configs
	Sample Input:
	{
	"entityID": "75781d57-a5cd-4db2-a1d5-58407a8c7887",
	"b64Certificate":
	"MIIDsjCCApqgAwIBAgIEIXc9vjANBgkqhkiG9w0BAQsFADB5MUMwQQYDVQQDDDo3YjQwNDMwYS04
	\nODAxLTQ2NDctOTNjNy03YzNjMjVkZTBkYTQtc2Vydmljb2dkZWZhdGFsb2dkZWZhdWx0MQ0wCwYDVQQ
	L\nDAROb25lMRQwEgYDVQQKDAtOb25lIEw9Tm9uZTENMAsGA1UEBhMETm9uZTAeFw0xNjExMDIxMz
	Uw\nNTBaFw0xNzAxMzExMzUwNTBaMHkxQzBBBgNVBAMMOjdiNDA0MzBhLTg4MDEtNDY0Ny05M2M3L
	Tdj\nM2MyNWRlMGRhNC1zZXJ2aWNlY2F0YWxvZ2RlZmF1bHQxDTALBgNVBAsMBE5vbmUxFDASBgNV
	BAoM\nC05vbmUgTD1Ob25lMQ0wCwYDVQQGEwROb25lMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMII
	BCgKC\nAQEAryLcEinIjhnUu9wP8H/AWn/rYA2IkcuacD6VNEzHaNCBR+k//2MNv5jsVGAxpxUkjM
	i8uIjM\nJvTvW7wVEzMGVTai6XDG48jpZSTIkftnpeZuO3iydJoSI5BOiYxn4d6VqZnEDPas1Qxrf
	iKsMqbC\nbfuWCtdOYE2Rqh8s0U6+BA2D/pXxbykfMYGa3hNbTgsvZjkfUropWTxrkNbP6mWOMbcC
	03e9ih9i\n95y3Et1APQ9uLDxcGf3Rr7h/md7k1S7pEunuJw7YSgmSDsg2gFnEnubT9SeWUvj5oT3
	/fHFElOvQ\nf8QlGKAJdRG1sP07mBSztDMlSYbtHWJfi+bYitD81wIDAQABo0IwQDAfBgNVHSMEGD
	AWgBQPOMLi\nmFPO0Ooj9Vs7UKmMdmhg3zAdBgNVHQ4EFgQUDzjC4phTztDqI/VbO1CpjHZoYN8wD
	QYJKoZIhvcN\nAQELBQADggEBAAwyRikaRzL/7ZahIonrsIxRr8QW+JRCAXJS52PRag/dGlpsxCp6
	/xD3QxJ+/EY2\n7gvO0lyBth23oKJVt3zgIH5tC+VHTdmT4Eeluv4iw4ZU0qYD/NCCEBilII68xOr
	ASbE5fiBWpn3Q\nm7le5IXK7KIFUa5VmfOuGgXap9s0AF1TelGPjjlNXmMxWJgxlu8ms7/Uoaju2H
	dFyznAyK0bdzSX\nguR2VsQiwbWTuBDKySc9hoZd4qVFTJmVTVrjbpmrAEY/xk+OCVb0T1JJBtlZQ
	EsYe6KR2xdnE6ny\nqycNHpclxVJ8yIXxeoLnJK2pmCbIcBt8v2fQPhPneBbaZOlerBg=",
	"b64PrivateKey":
	"MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCvItwSKciOGdS73A/wf8Baf+tg
	\nDYiRy5pwPpU0TMdo0IFH6T//Yw2/mOxUYDGnFSSMyLy4iMwm9O9bvBUTMwZVNqLpcMbjyOllJMi
	R\n+2el5m47eLJ0mhIjkE6JjGfh3pWpmcQM9qzVDGt+IqwypsJt+5YK105gTZGqHyzRTr4EDYP+lf
	Fv\nKR8xgZreE1tOCy9mOR9SuilZPGuQ1s/qZY4xtwLTd72KH2L3nLcS3UA9D24sPFwZ/dGvuH+Z3
	uTV\nLukS6e4nDthKCZIOyDaAWcSe5tP1J5ZS+PmhPf98cUSU69B/xCUYoAl1EbWw/TuYFLO0MyVJ
	hu0d\nYl+L5tiK0PzXAgMBAAECggEALqim4N/o4pLXLkVuqbAfWv0BhFGWtOD9gDHsJkbeSXpjNvl
	ZZ3zI\nSOdA7ynBkLX9StSgErm/ShGvQ01UgAzz/vfTZ0X4du8r3xpxpRLJhlVhwM5jHNV/R6JGij
	ax5mca\nkFi69okxeoEYkj5CiiwLWKnSS4kZBGcmC6DKm+jSjtlop+ErzcLmiBqBPlQHL/rZpp0T6
	2ojOMB/\nD8Au0IFecNIyitnTORBaOVRt1ohQXBhsrjSHQcXmP7TsDrm6H5XmE3sDfDT6UrYyvLNM
	uCNBfmrj\noE/kNnFUiQZthJWkFWoHSM1eehuUR6nsubg0q0KGrsI9ta+rof0FY51Ogr5jYQKBgQD
	44/5LT1u1\n6NLfM24dd2f6gD8cSV4VVfRLRktLogjqa8n3kTZOb/ELgLQDPotcHOQXDwDmK2OYpc
	fRG2RgGt22\nMXdLHawjWItmr2wkzhanojapdssiCU9NDb209eHOUpT82pz0Vouw9LlzV26J1++Ki
	BoyGMO5Xh+L\nKjm5aNZQHQKBgQC0I4nuCvFMvJ14gIRvVmcCcHbHREVmuSeFOKsXL8kYkYsrUvcJ
	mSkw6GnMtiSh\nfsHwFtJmakZa+QDBNUJhKuvyhfC+9vaUsPjXK20Oa5dd8eQoN9Bz9dTptjx001f
	phFiidNE4+f/1\nsKN/0YnKoBOJsEb7Zv3yzJCMPBoPHvmWgwKBgQCvTl+iCf6N7bUB88a+yIkbf1
	N0iBTVsFpG3vdQ\nCYyAGXYDg2ud6ej9ciTZGCeutMbPmwjGFo+rSDGrDsEvlBzQJJ1i8j56Evb1V
	+AzOFnqry4TRRIl\nIiuSGXiiyoHHApHgW9crnv37oRQyssWwH8GgcOcKnDjYCvzq184a00YI3QKB
	gDWqMLkdwW0e87qm\nbs3Ma7uqTXhnulUz67Ygf7fUoJAVK+SoPrg5TLApTPuTd6402QnxgpTILFW
	FwNfOSgwwgUIq7OG3\nKRZ68mcHpOGa4+k02seweQVSwy8s/y2+mH4U02LycjILKFnFWbAGeIpIzg
	lC3qKeuCDRGG7uqMTA\ncZKJAoGAcrP9/zpxLyyBm8WjmAmC0UVgpCZmBDEEQKZxqNmqP/oIYbXCK
	ClS5sQc7ybeXigyq37B\ncAuyHa+rVVl/FClnWlsgG9DmZOTjqyL7ttJSP9hJjHzlJp5dw6uVVexz
	WheZWFKbGC0obLod5522\nM+n5j+epGNK6tTRWfVERYnXthcc="
	}

Table 9-1 SAML REST API Table

Area	Examples
GET	**Enable/Disable SAML SSO Setting** GET URL: http://<ServerURL>/RequestCenter/nsapi/ucsd/sso Sample response: {"Map":{"mode":"SAML","enable":true,"enableSAMLApiAuthentication":true}}
POST	**Enable/Disable SAML SSO Setting** POST URL: http://<ServerURL>/RequestCenter/nsapi/ucsd/sso Sample Payload: {"Map":{"mode":"SAML","enable":true,"enableSAMLApiAuthentication":true}}

Newly Added Properties

NewScale.Properties

The below table lists the newly added *newscale.properties* that can control settings on other modules.

Table A-1 newscale.properties introduced in patch releases

Property	Description
Serviceformelement.display.instructional.helptext	Enables the legacy display style for non-grid Dictionary field help texts by setting the property to true.
Serviceform.simtask.validation.skip	Bypasses the validation check that prevents the removal of Active Form Components, containing 1 or more service item based Dictionaries, from the service when 1 or more service item tasks are defined based on these dictionaries by setting the property to true.
MAX_ALLOWED_SERVICE_ITEM_EXPORT_RECORD_SIZE	Controls the service item records that can be exported to a .csv file from the My Stuff page and the Service Item Manager module by setting the parameter (Default is 1000).
servicecatalog.display.direct.category.service.association	Enables this to view the direct category associated with the service from Service Catalog by setting the property to true. **Note** This feature supports the display of only three levels of category hierarchy (excluding the placeholder category meant to occupy the Browse Category - All Showcase Location).
serviceform.add.attachment	Add attachments to a requisition at the time of ordering from Service Catalog module by setting the property to true.
ObjectCache.Environment.Name	You can assign name to the CPSC environment by using the property. This property can be referenced in the application with the namespace Site.ENV. For example, this namespace is used in the email templates to identify the source environment from which the email has originated, you can use the pound character (#) as delimiter to resolve the namespace.
localization.languages.export.limit=<export_count> localization.languages.import.limit=<import_count>	You can specify the limit on the number of languages (Default is 10) to be imported and exported by editing the two properties.
servicecatalog.display.req.percentage.completion	You can enable the percentage display in Service Catalog > My Stuff > Open Orders page by setting the property to true.

Table A-1 newscale.properties introduced in patch releases

Property	Description
nsapi.directory.person.hide.secure.information	You can hide the encrypted value of the password attribute and the cleartext value of the cloudpassword attribute by setting the property to true.
localization.languages.column.display.limit = <count>	You can specify the limit on the number of language columns (Default is 4) to be displayed in the grid by editing the property.
modules.ordermanagement.disable	You can disable Order Management for end user from being able to access the module directly from service catalog or service manager by editing the property to true.
serviceform.element.serverside.editable.disable	You can disable Editable on Server–Side only by editing the property to true.
deployment.service.authtask.agent.parameters.override	Overridden parameter values for external Authorization and Review tasks could be made to persist after deployment by editing the property to true.
servicecatalog.orderonbehalf.enable	Enables the new Order on Behalf functionality by setting the property to true.
serviceitem.remove.nextline	Enable this to convert multi–line service item field values to single–line value when exported to *.csv* by setting the property to true.
nsapi.person.create.trigger.email.enable	Enables an email on creation of new user account using nsAPI by setting the property to true.
ldap.enabled.prevent.user.password.update	You can disable the visibility of password related fields when the Directory Integration is enabled by setting the property to true.
enable.legacy.cr.isfjs.show.hide	Enable this to get back to the legacy functionality of Show/Hide CR and ISF JS by setting the property to true.
serviceitem.update.customer.details	You can update the subscription information for the Service Item instance while performing update operation by setting the property to true.
serviceform.ddr.null.value.as.empty	Enables the rendering of blank values as such on the Service Form for all dictionary fields by setting the property to true.
Serviceform.helpicon.clickable	Enables '?' icon to be clicked on to close the help text popup by setting the property to true.
showAllTransactionsMessages	You can disable the default filter on the messages tab in Service Link during page load by setting the property to true.
prevent.queue.task.self.perform	You can disable the queue performers to access Authorization, Review, and Monitor Plan tasks for their own Requisitions, from Service Manager or Order Management modules by setting the property to true.
admin.setlocale.global	Enables the default language setting a global one, affecting both new and existing users in the system by setting the property to true.
session.token.nsapisc.validation	Enables token-based validation for nsAPI calls from Service Catalog by setting the property to 1.
htmlscan.enable	Enables the system to scan and remove the nonconforming content from the user input by setting the property to 1.
serviceitem.nsapi.rbac.check	You can disable the RBAC check for the service item based nsAPI's by setting the property to false.