HP Cloud Service Automation

Software Version: 4.20

API Reference

Document Release Date: December 2014
Software Release Date: December 2014

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

RED HAT READY™ Logo and RED HAT CERTIFIED PARTNER™ Logo are trademarks of Red Hat, Inc.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

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

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

To check for recent updates or to verify that you are using the most recent edition of a document, go to:
http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:
http://h20229.www2.hp.com/passport-registration.html

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

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

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

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

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

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

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

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

To find more information about access levels, go to:

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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

HP CSA 4.x API Reference Introduction

The APIs for HP Cloud Service Automation (CSA) use a REST interface. See http://en.wikipedia.org/wiki/Representational_state_transfer for general REST information. This documentation assumes that you know how to use REST interfaces.

See the *HP Cloud Service Automation API Quick Start* for information on communication with CSA using RESTful calls.

Caution: This document includes information only on the REST APIs introduced in CSA 3.x, and that are still available in CSA 4.x. Information on REST APIs introduced in CSA 4.x can be found in the *HP Cloud Service Automation API Quick Start*.

The base URL for a CSA REST API is https://<host:port>/csa/rest, which is appended with the specific URI for the API call. For example, to access the <example> API, you would use the URL: https://<host:port>/csa/rest/<example>.

Because XML content passed into or returned by CSA REST API calls can be lengthy, example XML content presented in this document will often be abbreviated to include just the more pertinent XML content.

Note: Special or localized characters used in the URL of REST API calls must be encoded before they are sent to the server.

Tip: You should only include one value for a Boolean property and the value must be either true or false.

Chapter 1: Artifact API

Description

Use this API to view, create, and modify CSA artifacts.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

Artifact

URI	Method	Parameters	Description
/artifact	POST	userIdentifier	"Create an artifact" on page 13
/artifact/<artifact_id>	GET	userIdentifier, scope, detail, view	"View an artifact" on page 13
/artifact/<artifact_id>	PUT	userIdentifier, scope, view	"Update an artifact" on page 17
/artifact/<artifact_id>	DELETE	userIdentifier	"Delete an artifact" on page 20
/artifact/fastview/<artifact_id>	GET	userIdentifier, view	"Retrieve a predefined view for an artifact" on page 20

URI	Method	Parameters	Description
/artifact/<artifact_id>/resolveProperties	GET	userIdentifier, propertyName	"Retrieve resolved properties for an artifact" on page 23

Group

URI	Method	Parameters	Description
/artifact/<organization_id>/group	GET	userIdentifier	"List active groups associated with an organization" on page 26
/artifact/<organization_id>/group	POST	userIdentifier	"Add groups to an organization" on page 27
/artifact/<organization_id>/group/<grou_id>	PUT	userIdentifier	"Update group display name, distinguished name" on page 30
/artifact/<organization_id>/group/<group_id>	DELETE	userIdentifier	"Delete or disassociate group from an organization" on page 31

Resource Provider

URI	Method	Parameters	Description
/artifact	GET	userIdentifier, artifactType	"List resource providers" on page 32

Service Offering

URI	Method	Parameters	Description
/artifact/<service_offering_id>/document	POST	userIdentifier	"Add document to service offering" on page 34
/artifact/<service_offering_id>/document	DELETE	userIdentifier	"Delete document from service offering" on page 36
/artifact/<service_offering_id>/document/<document_id>	POST	userIdentifier	"Update document in service offering" on page 38
/artifact/<catalog_id>/publish	POST	userIdentifier	"Publish service offerings to catalog" on page 39
/artifact/<catalog_id>/unpublish	POST	userIdentifier	"Unpublish service offerings from catalog" on page 41

Note: You can view information about an artifact with GET /artifact/<artifact_id> or GET /artifact/fastview/<artifact_id>. The fastview API can traverse associations, while the standard artifact API only returns information for the artifact.

Artifact types

You can work with the following artifact types using the /artifact API. You can use them with the methods marked in the table.

Artifact type	GET	POST	PUT	DELETE
Approval process	X			
Approval template	X			

Artifact type	GET	POST	PUT	DELETE
Approver	X			
Catalog	X	X	X	X
Document	X	X		X
Group	X			
Named approver approval template	X			
Organization	X	X	X	X
Person	X			
Resource binding	X		X	
Resource environment	X			
Resource offering	X	X	X	
Resource pool	X	X	X	X
Resource provider	X	X	X	X
Resource subscription	X		X	
Service component	X	X	X	X
Service blueprint	X	X	X	X
Service instance	X		X	
Service offering	X	X	X	X
Service request	X			
Subscription	X			

Create an artifact

Details

URI	`/artifact`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

View an artifact

Details

URI	`/artifact/<artifact_id>`
Method	GET

URI	`/artifact/<artifact_id>`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree|view]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned. If the value is *vie*, then the view parameter is required.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202
	
	Note: Some API calls do not support all possible values for this parameter.
	
	`detail=FULL`, even as the default value, is not accepted for organization artifacts because the volume of content returned can be excessively large. Specify `detail=BASIC` to avoid an exception message.
	
	`view=<view_type>`
	Required when value for `scope` is *view*; if this parameter is defined, then the value for `scope` is processed as if its value was *view*. The default is *basicinfo*. See "Artifact views" on page 45 for a list of view types.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/artifact/90e72e4f3b00a69e013b0bf7ed55002e?userIdentifier=<user_id>
```

The following XML was returned in the response:

```
<ResourceEnvironment>
  <id>90e72e4f3b00a69e013b0bf7ed55002e</id>
  <objectId>90e72e4f3b00a69e013b0bf7ed55002e</objectId>
  <createdOn>2012-11-16T17:24:55.765-08:00</createdOn>
  <updatedOn>2012-11-16T17:24:55.765-08:00</updatedOn>
  <createdBy>
    <id>90d96588360da0c701360da0f1d5f483</id>
    <objectId>90d96588360da0c701360da0f1d5f483</objectId>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <name>admin</name>
    <displayName>admin</displayName>
    <disabled>false</disabled>
  </createdBy>
  <updatedBy>
    <id>90d96588360da0c701360da0f1d5f483</id>
    <objectId>90d96588360da0c701360da0f1d5f483</objectId>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <name>admin</name>
    <displayName>admin</displayName>
    <disabled>false</disabled>
  </updatedBy>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <description>TestEnv</description>
  <name>TestEnv_November 17, 2012 1:24:55 AM UTC</name>
  <displayName>TestEnv</displayName>
  <state>
    <id>90d96588360da0c701360da0ef470038</id>
    <objectId>90d96588360da0c701360da0ef470038</objectId>
    <createdOn>2012-11-01T15:16:54.687-07:00</createdOn>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <description>Active</description>
    <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
    <name>ACTIVE</name>
    <displayName>Active</displayName>
    <disabled>false</disabled>
    <categoryType>
       <id>90d96588360da0c701360da0ef420037</id>
```

```
      <objectId>90d96588360da0c701360da0ef420037</objectId>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <name>ARTIFACT_STATE</name>
      <displayName>Artifact State</displayName>
      <extensible>false</extensible>
    </categoryType>
  </state>
  <artifactType>
    <id>90d96588360da0c701360da0eedb0020</id>
    <objectId>90d96588360da0c701360da0eedb0020</objectId>
    <createdOn>2012-11-01T15:16:57.787-07:00</createdOn>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <description>Resource Environment</description>
    <iconUrl>
      /csa/images/categories/artifact_type/resource_
environment.png
    </iconUrl>
    <name>RESOURCE_ENVIRONMENT</name>
    <displayName>Resource Environment</displayName>
    <disabled>false</disabled>
    <categoryType>
    <id>90d96588360da0c701360da0eeb40017</id>
      <objectId>90d96588360da0c701360da0eeb40017</objectId>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <name>ARTIFACT_TYPE</name>
      <displayName>Artifact Type</displayName>
      <extensible>false</extensible>
    </categoryType>
  </artifactType>
  <disabled>false</disabled>
  <numberOfProvider>0</numberOfProvider>
  <numberOfServiceDesign>0</numberOfServiceDesign>
  <numberOfCatalog>0</numberOfCatalog>
</ResourceEnvironment>
```

Update an artifact

Details

URI	/artifact/<artifact_id>
Method	PUT

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`view=<view_type>&scope=view`
	Optional; Used to update artifacts based on pre-defined views. See "Artifact views" on page 45 (description column) for a list of view types that support update operation.
	
	`<_artifact_property>_action_=merge`
	Optional; use the *merge* option with the *action* meta tag query parameter to update only a portion of the artifact. The *action* meta tag can either be specified globally for the artifact by including parameter `_action_=merge`, or for a specific property e.g., `_property_values_action_=merge`.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Note: To completely update the artifact, i.e., replace the persistent artifact, do not use the *view* or *merge* parameters and include all artifact content in the request body. Note that if only a portion of the artifact content is sent in the request body, any unspecified content will be removed from the artifact.

To update a portion of the artifact:

- Use a pre-defined view that contains a subset of the artifact properties; only that subset of properties will be updated per the values specified in the request

body. See"Artifact views" on page 45 for a list of view types.

- Use the *merge* option as described under Parameters.
  Note: You can use the *merge* option with the v*iew* parameter to update only the view properties for which you specify values in the request body.

Note: Collection specific behavior

When a *merge* option is specified on a collection, for example `_property_values_action_=merge`, all collection items specified in the PUT request body are updated. Any other collection items are left untouched.
For the *property* attribute of an artifact, the items of this collection attribute are matched by name. For all other attributes, the collection items are matched by id.

Examples

The following examples demonstrate how to update an artifact.

This example shows how to change the finalize flag of a component using the view parameter.
The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/90e72e4f3af5c989013afb471ebc0264?userIdentifier=&scope=view&view=componentfinalize
```

The following XML was sent in the request body:

```
<ServiceComponent>
  <id>90e72e4f3af5c989013afb471ebc0264</id>
  <toFinalize>false</toFinalize>
</ServiceComponent>
```

This example shows changing the display name of a resource provider. This example does not use the view parameters. To use this approach, retrieve the artifact using GET artifact API, modify the necessary value (in this example - displayName), and use that as the body of the PUT request to update the artifact.
The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/90e72e4f3b00a69e013b0c049ab00033?userIdentifier=<user_id>
```

The following XML was sent in the request body:

```
<ResourceProvider>
  <id>90e72e4f3b00a69e013b0c049ab00033</id>
  <objectId>90e72e4f3b00a69e013b0c049ab00033</objectId>
  <createdOn>2012-11-16T17:38:46.576-08:00</createdOn>
  <updatedOn>2012-11-16T17:38:46.576-08:00</updatedOn>
  <createdBy>
    <id>90d96588360da0c701360da0f1d5f483</id>
    ...
  </createdBy>
  <updatedBy>
    <id>90d96588360da0c701360da0f1d5f483</id>
    ...
  </updatedBy>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <description>TestProvider</description>
  <name>TestProvider_November 17, 2012 1:38:46 AM UTC</name>
  <displayName>TestProviderModified</displayName>
  <state>
    <id>90d96588360da0c701360da0ef470038</id>
    ...
  </state>
  <artifactType>
    <id>90d96588360da0c701360da0eed8001f</id>
    ...
  </artifactType>
  <disabled>false</disabled>
  <accessPoint>
    <id>90e72e4f3b00a69e013b0c049a740032</id>
    ...
  </accessPoint>
  <providerType>
    <id>90d96588360da0c701360da0eeac0016</id>
    ...
  </providerType>
  <numberOfResourceOffering>0</numberOfResourceOffering>
  <numberOfEnvironment>0</numberOfEnvironment>
  <numberOfPools>0</numberOfPools>
</ResourceProvider>
```

Delete an artifact

Details

URI	`/artifact/<artifact_id>`
Method	DELETE
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Deletion is subject to a set of business rules which depend on the type of artifact. The business rules for artifacts are explained in the table below. Note that consumption artifacts are not removed from the database when they are deleted; instead, the artifact is marked as retired. Please refer to the "Catalog API" on page 55 for retiring consumption artifacts.

Artifact type	Details
Resource provider	Can only be deleted if no active service subscriptions use the resource provider.
Service design	Can be deleted when all associated service offerings or service instances are retired.

Retrieve a predefined view for an artifact

Because the REST API presented here returns content in a different format when retrieving a single result versus multiple results, and could thereby complicate your using the results, it is recommended that the "View an artifact" on page 13 API be used. Performance intensive applications might still choose to use the following API.

Details

URI	`/artifact/fastview/<artifact_id>`
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`view=<view_type>`
	Required; see "Artifact views" on page 45 for a list of view types.

Returns	200 - Ok 404 - Not found 500 - Server exception

Examples

The following URL was sent. Note that artifact_id is the ID of any artifact that has an accesspoint.

```
https://<host>:port/csa/rest/artifact/fastview/90e72e4f3abe4bf401
3ac24735730010?userIdentifier=<user_id>&view=accesspoint
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resultView>
  <resultMap>
    <entry>
      <key>accessPoint.uri</key>
      <value xsi:type="xs:string">http://amz:443</value>
    </entry>
    <entry>
      <key>accessPoint.password</key>
      <value xsi:type="xs:string">amz</value>
    </entry>
    <entry>
      <key>accessPoint.username</key>
```

```
      <value xsi:type="xs:string">amz</value>
    </entry>
    <entry>
      <key>accessPoint.category.name</key>
      <value xsi:type="xs:string">URL</value>
    </entry>
  </resultMap>
</resultView>
```

Filtering

You can filter the results by providing a value for a property in the URI. The query is then filtered based on that property. You can use the properties listed in "Artifact views" on page 45.

Note: The property name that is specified in the URL must have the period (.) character replaced with the underscore (_) character.

The following example shows the result when the previous example is filtered on a property name.

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/fastview/90e72e4f3b00a69e
013b0c049ab0003?userIdentifier=<user_
id>&view=propertyinfo&property_name=propBLN
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resultView>
  <resultMap>
    <entry>
      <key>property.values.value</key>
      <value xsi:type="xs:string">true</value>
    </entry>
    <entry>
      <key>property.consumerVisible</key>
      <value xsi:type="xs:boolean">true</value>
    </entry>
    <entry>
```

```
      <key>property.displayName</key>
      <value xsi:type="xs:string">Boolean property</value>
    </entry>
    <entry>
      <key>property.valueType.name</key>
      <value xsi:type="xs:string">BOOLEAN</value>
    </entry>
    <entry>
      <key>property.name</key>
      <value xsi:type="xs:string">propBLN</value>
   </entry>
  </resultMap>
</resultView>
```

Retrieve resolved properties for an artifact

A property can have a source binding configured that indicates its value is to be retrieved from a property on another artifact. The REST API discussed here provides a mechanism to retrieve the value from the source property. As part of this retrieval, relevant tokens configured on properties are also resolved.

There are two approaches to retrieving resolved properties:

- Retrieve all properties
- Retrieve a single named property

Details

URI	`/artifact/<artifact_id>/resolveProperties`
Method	GET

URI	`/artifact/<artifact_id>/resolveProperties`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`propertyName=<property_name>`
	Optional; the name of the property you want to retrieve. Only retrieves the value for the property specified.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

The following URL was used to retrieve all properties for an artifact:

```
https://<host>:<port>/csa/rest/artifact/<id>/resolveProperties?us
erIdentifier=<user_id>
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Properties>
    <property> ... </property>
    <property> ... </property>
    <property>
        <id>90d9651a3684c7f0013684cafda80005</id>
        <createdOn>2012-04-05T16:15:57.480-07:00</createdOn>
        <updatedOn>2012-04-05T16:15:57.497-07:00</updatedOn>
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <name>PARENT_SVC_COMPONENT_ID</name>
        <artifact>
            <id>90d9651a3684c7f0013684cafda80008</id>

<isCriticalSystemObject>false</isCriticalSystemObject>
            <name>TEST_CHILD_COMPONENT</name>
            <disabled>false</disabled>
        </artifact>
```

```
        <propertyBindings>
            <id>90d9651a3684c7f0013684cb28820009</id>
            <createdOn>2012-04-05T16:16:08.450-07:00</createdOn>
            <updatedOn>2012-04-05T16:16:08.450-07:00</updatedOn>
            <propertyBindingType>
                <name>SOURCE</name>
                <categoryType>
                    <name>PROPERTY_BINDING_TYPE</name>
                                        ...
                </categoryType>
                                ...
            </propertyBindingType>
            <artifact>90d9651a3684c7f0013684c91f8c0004</artifact>
            <artifactPropertyName>SVC_COMPONENT_
ID</artifactPropertyName>
        </propertyBindings>
        <scope> ... </scope>
        <valueType> ... </valueType>
        <values>
            <id>90d9651a3684c7f0013684c91f8b0002</id>
            <createdOn>2012-04-05T16:13:55.083-07:00</createdOn>
            <updatedOn>2012-04-05T16:13:55.083-07:00</updatedOn>
            <value>90d9651a3684c7f0013684c91f8c0004</value>
        </values>
        <maxOccurs>0</maxOccurs>
        <minOccurs>0</minOccurs>
        <orderIndex>0</orderIndex>
        <confidential>false</confidential>
        <encrypted>false</encrypted>
        <consumerReadOnly>false</consumerReadOnly>
        <consumerVisible>false</consumerVisible>
    </property>
    <property> ... </property>
    <property> ... </property>
</Properties>
```

The following URL was used to retrieve a single property:

```
https://<host>:<port>/csa/rest/artifact/<id>/resolveProperties?
userIdentifier=<user_id>&propertyName=PARENT_SVC_COMPONENT_ID
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Properties>
    <property>
        <id>90d9651a3684c7f0013684cafda80005</id>
        <name>PARENT_SVC_COMPONENT_ID</name>
         ...
    </property>
</Properties>
```

List active groups associated with an organization

Details

URI	`/artifact/<organization_id>/group`
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/8a81818f3d02fb7e013d03088
91d0003/group?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GroupList>
```

```
  <count>1</count>
  <limit>1</limit>
  <group>
    <id>8a81818f3d02fb7e013d030af854000f</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>sc_February 22, 2013 5:54:43 PM UTC</name>
    <displayName>ServiceConsumer1</displayName>
    <state>...</state>
    <artifactType>...</artifactType>
    <disabled>false</disabled>
    <distinguishedName>
      cn=ServiceConsumer,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
    <role>
      ...
    </role>
  </group>
</GroupList>
```

Add groups to an organization

Details

URI	`/artifact/<organization_id>/group`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Note that the role must be specified for each group in the request body. Valid roles are as follows.
For Consumer type organizations:

- SERVICE_CONSUMER

For Provider type organizations:

- CONSUMER_SERVICE_ADMINISTRATOR
- SERVICE_BUSINESS_MANAGER
- SERVICE_DESIGNER
- CSA_ADMIN
- RESOURCE_SUPPLY_MANAGER
- SERVICE_OPERATIONS_MANAGER

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/8a81818f3d1421e7013d14236
35a0003/group?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent:

```
<GroupList>
  <group>
    <displayName>My-Group-Name</displayName>
    <distinguishedName>
      cn=TestConsumer,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
    <role>
      <name>SERVICE_CONSUMER</name>
    </role>
  </group>
  <group>
    <displayName>Another-Group-Name</displayName>
    <distinguishedName>
      cn=TestConsumer2,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
    <role>
      <name>SERVICE_CONSUMER</name>
    </role>
```

```
  </group>
 </GroupList>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GroupList>
  <count>2</count>
  <limit>2</limit>
  <group>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>My-Group-Name</name>
    <displayName>My-Group-Name</displayName>
    <disabled>false</disabled>
    <distinguishedName>
      cn=TestConsumer,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
    <role>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>SERVICE_CONSUMER</name>
      <disabled>false</disabled>
    </role>
  </group>
  <group>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>Another-Group-Name</name>
    <displayName>Another-Group-Name</displayName>
    <disabled>false</disabled>
    <distinguishedName>
      cn=TestConsumer2,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
    <role>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>SERVICE_CONSUMER</name>
      <disabled>false</disabled>
    </role>
  </group>
</GroupList>
```

Update group display name, distinguished name

Use this /artifact URI to update the group display name and/or distinguised name for the specified organization.

Details

URI	`/artifact/<organization_id>/group/<group_id>`
Method	PUT
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/8a81818f3d02fb7e013d03088
91d0003group/8a81818f3d1437e2013d1795d41107ea?userIdentifier=90d9
6588360da0c701360da0f1d5f483
```

The following XML was sent:

```
<GroupList>
  <group>
    <displayName>My-New-Group-Name</displayName>
    <distinguishedName>
      cn=TestConsumer,ou=ConsumerGroup,ou=CSAGroups
    </distinguishedName>
  </group>
</GroupList>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Group>
  <id>8a81818f3d1437e2013d1795d41107ea</id>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <name>sc_February 22, 2013 5:54:43 PM UTC</name>
  <displayName>My-New-Group-Name</displayName>
  <state>
    <id>90d96588360da0c701360da0ef470038</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>ACTIVE</name>
    <disabled>false</disabled>
  </state>
  <artifactType>
    <id>90d96588360da0c701360da0eeff002b</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>GROUP</name>
    <disabled>false</disabled>
  </artifactType>
  <disabled>false</disabled>
  <distinguishedName>
    cn=TestConsumer,ou=ConsumerGroup,ou=CSAGroups
  </distinguishedName>
</Group>
```

Delete or disassociate group from an organization

Use this URI to delete a group or to disassociate it from an organization. If no organization is associated with this group, the group will be deleted. Otherwise, the group will be disassociated from the specified organization.

Details

URI	/artifact/<organization_id>/group/<group_id>
Method	DELETE

URI	`/artifact/<organization_id>/group/<group_id>`
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/8a81818f3d1421e7013d14236
35a0003/group/8a81818f3d1437e2013d1795d41107ea?userIdentifier=90d
96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<messageList>
  <messages>Removed role association for My-New-Group-
Name</messages>
</messageList>
```

List resource providers

Details

URI	`/artifact`
Method	GET

URI	`/artifact`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`artifactType=RESOURCE_PROVIDER`
	Required; the only valid value is *RESOURCE_PROVIDER*. Basic view information is provided in the return content.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact?userIdentifier=
90d96588360da0c701360da0f1d5f483&artifactType=RESOURCE_PROVIDER
```

The following XML was returned in the response:

```
<ResourceProviderList>
  <count>3</count>
  <limit>0</limit>
  <resourceProvider>
    <id>90e72e583d4c6e77013d4c96b1ab0010</id>
    <objectId>90e72e583d4c6e77013d4c96b1ab0010</objectId>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>Sitescope A</description>
    <name>Sitescope_A_March 9, 2013 12:39:36 AM UTC</name>
    <displayName>Sitescope A</displayName>
    <disabled>false</disabled>
  </resourceProvider>
  ...
</ResourceProviderList>
```

Add document to service offering

Details

URI	`/artifact/<service_offering_id>/document`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent with headers:

- Content-type: multipart/form-data
- Content-Disposition: form-data; name="file"
- Content-Type: application/octet-stream

```
https://<host>:<port>/csa/rest/artifact/90cef5de3c63429f013c68b8c
dda0bad/document?userIdentifier=90cef5de3c63429f013c642c7fc708ab
```

To better demonstrate how to use this REST API URI, the following a python script attaches a text file to a service offering. The script was created based on HTML documentation. You can find information on html form content types at: http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2.

```
""" Sample python script that attaches a text file in the current
directory to
an ACTIVE CSA service offering on a local host, with default
credentials.
Script usage:
```

```
    <scrip-name.py> <service offering uuid> <filename> <admin
uuid>
The following example attaches file doc.txt to the service
offering with id 90cec0773c83a11a013c871e4c1a0503 using an admin
user ID
90d96588360da0c701360da0f1d5f483
post-doc.py 90cec0773c83a11a013c871e4c1a0503 doc.txt
90d96588360da0c701360da0f1d5f483
Tested with: ActivePython 3.2.2.3
"""
from xml.dom.minidom import parse, parseString
from http.client import HTTPSConnection
from base64 import b64encode
import mimetypes
import sys
def get_content_type(filename):
    return mimetypes.guess_type(filename)[0] or
'application/octet-stream'
def get_file_contents(filename):
    CRLF = '\r\n'
    f = open(filename, 'r')
    return CRLF.join(f.readlines())
def encode_multipart_formdata(filename):
    BOUNDARY = '----------CSA_r0ck$_$'
    CRLF = '\r\n'
    L = []
    L.append('--' + BOUNDARY)
    L.append('Content-Disposition: form-data; name="file";
filename="%s"' % filename)
    L.append('Content-Type: %s' % get_content_type(filename))
    L.append('')
    L.append(get_file_contents(filename))
    L.append('--' + BOUNDARY + '--')
    L.append('')
    body = CRLF.join(L)
    content_type = 'multipart/form-data; boundary=%s' % BOUNDARY
    return content_type, body
def addDocumentToOffering(offeringId, documentName, userId):
    conn = HTTPSConnection("localhost:8444")
```

```
    userAndPass = b64encode
(b"csaTransportUser:csaTransportUser").decode("ascii")
    post_url = "/csa/rest/artifact/" + offeringId +
"/document?userIdentifier="+userId
    content_type, body = encode_multipart_formdata(documentName)
    content_length =  str(len(body))
    headers = {'Authorization' : 'Basic %s' %  userAndPass,
               'Content-Type' :'%s' % content_type, 'content-
length' : str(len(body)), 'Accept' : 'application/xml'}
    conn.request('POST', post_url, body, headers=headers)
    res = conn.getresponse()
    data = res.read()
    print(data)
def main(offeringId, documentName, userId):
    addDocumentToOffering(offeringId, documentName, userId)
if __name__ == "__main__":
    offeringId = sys.argv[1]
    documentName = sys.argv[2]
    userId = sys.argv[3]
    main(offeringId, documentName, userId)
```

Delete document from service offering

Details

URI	`/artifact/<service_offering_id>/document/<document_id>`
Method	DELETE
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/90cef5de3c63429f013c68b8c
dda0bad/document/90cef5de3c63429f013c68b8cdda0bad?userIdentifier=
90cef5de3c63429f013c642c7fc708ab
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Document>
    <id>90cef5de3c63429f013c68d4146a0bce</id>
    <isCriticalSystemObject> false </isCriticalSystemObject>
    <name>Example Offering Doc.txt</name>
    <state>
        ...
    </state>
    <artifactType>
        ...
    </artifactType>
    <disabled>false</disabled>

<url>/csa//document/download?id=90cef5de3c63429f013c68d4146a0bce<
/url>
    <content>Sample Doc Content</content>
    <mimeType>application/octet-stream</mimeType>
    <documentType>
       ...
    </documentType>
</Document>
```

Update document in service offering

Details

URI	`/artifact/<service_offering_id>/document/<document_id>`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent with headers:

- Content-type: multipart/form-data
- Content-Disposition: form-data; name="file"
- Content-Type: application/octet-stream

```
https://<host>:<port>/csa/rest/artifact/90cec0123ae20db2013ae2111e3e000a/document/90cec0123afffe57013afffec64f0001?userIdentifier=90d96588360da0c701360da0f1d5f483
```

See "Add document to service offering" on page 34 for a more detailed example.

Publish service offerings to catalog

Details

URI	`/artifact/<catlog_id>/publish`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/90e72e323b5330cc013b5358c
0940021/publish?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CatalogItems>
  <catalogItem>
    <artifactContext>
      <name>OFFERING1_January 30, 2013 6:18:47 PM UTC</name>
    </artifactContext>
    <artifactContextType>
      <name>SERVICE_OFFERING</name>
      <categoryType>
        <name>ARTIFACT_TYPE</name>
      </categoryType>
    </artifactContextType>
```

```
    <category>
      <name>CRM</name>
      <categoryType>
        <name>CATALOG_CATEGORY</name>
      </categoryType>
    </category>
  </catalogItem>
  <catalogItem>
    <artifactContext>
      <name>OFFERING2_January 28, 2013 11:32:17 PM UTC</name>
    </artifactContext>
    <artifactContextType>
      <name>SERVICE_OFFERING</name>
      <categoryType>
        <name>ARTIFACT_TYPE</name>
      </categoryType>
    </artifactContextType>
    <category>
      <name>ACCESSORY</name>
      <categoryType>
        <name>CATALOG_CATEGORY</name>
      </categoryType>
    </category>
  </catalogItem>
  <catalogItem>
    <artifactContext>
      <name>OFFERING1_January 30, 2013 6:18:47 PM UTC </name>
    </artifactContext>
    <artifactContextType>
      <name>SERVICE_OFFERING</name>
      <categoryType>
        <name>ARTIFACT_TYPE</name>
      </categoryType>
    </artifactContextType>
    <category>
      <name>AABBCCDD</name>
      <displayName>AABBCCDD</displayName>
      <categoryType>
        <name>CATALOG_CATEGORY</name>
```

```
      </categoryType>
    </category>
  </catalogItem>
</CatalogItems>
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messageList>
  <messages> Publish OFFERING1_January 30, 2013 6:18:47 PM UTC to
category AABBCCDD failed. Category doesn't exist.</messages>
  <messages> Publish OFFERING2_January 28, 2013 11:32:17 PM UTC
to category ACCESSORY succeeded.</messages>
  <messages> Publish OFFERING1_January 30, 2013 6:18:47 PM UTC to
category CRM succeeded.</messages>
</messageList>
```

Unpublish service offerings from catalog

Details

URI	`/artifact/<catlog_id>/unpublish`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/90e72e323b5330cc013b5358c
0940021/unpublish?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request:

```
<CatalogItems>
  <catalogItem>
    <artifactContext>
      <name>OFFERING1_January 30, 2013 6:18:47 PM UTC</name>
    </artifactContext>
    <artifactContextType>
      <name>SERVICE_OFFERING</name>
      <categoryType><name>ARTIFACT_TYPE</name></categoryType>
    </artifactContextType>
    <category>
      <name>CRM</name>
      <categoryType>
        <name>CATALOG_CATEGORY</name>
      </categoryType>
    </category>
  </catalogItem>
  <catalogItem>
    <artifactContext>
      <name>OFFERING2_January 28, 2013 11:32:17 PM UTC</name>
    </artifactContext>
    <artifactContextType>
      <name>SERVICE_OFFERING</name>
      <categoryType><name>ARTIFACT_TYPE</name></categoryType>
    </artifactContextType>
    <category>
      <name>AABBCCDD</name>
      <displayName>AABBCCDD</displayName>
      <categoryType><name>CATALOG_CATEGORY</name></categoryType>
    </category>
  </catalogItem>
</CatalogItems>
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messageList>
  <messages> Unpublish Service Offering OFFERING1_January 30,
```

```
2013 6:18:47 PM UTC from category CRM succeeded.</messages>
  <messages> Unpublish Service Offering OFFERING2_January 28,
2013 11:32:17 PM UTC from category AABBCCDD failed. Category
doesn't exist.</messages>
</messageList>
```

Retrieve artifact state and status

Details

URI	`/artifact/state/<artifact_id>`
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Request Body	None
Return Body	ArtifactStateInfo
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

For all artifacts, possible values for the returned artifactState are:

- ACTIVE
- RETIRED

The following state and status information will be returned for specific artifact types.

For service instance artifacts, state will be returned and will contain one of the following values:

- ACTIVE
- CANCELLED
- CANCELLING
- CANCEL_FAILED
- DEPLOYING
- EXPIRE_FAILED
- EXPIRING
- FAILED
- MODIFYING
- MODIFY_FAILED
- IN_PROGRESS
- PUBLIC_ACTION_FAILED
- RESERVED

For service request artifacts:

- State will be returned and will contain one of the following values:
  - APPROVED
  - CANCELLED
  - COMPLETED
  - IN_PROGRESS
  - PENDING_APPROVAL
  - REJECTED
  - SUBMITTED

- Status will be returned. If no value was set a null value will be returned. Otherwise it will contain one of the following values:
  - FAILURE
  - SUCCESS

For service subscription artifacts, status will be returned and will contain one of the following values:

- ACTIVE
- CANCELLED
- EXPIRED
- PAUSED
- PENDING
- TERMINATED

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/artifact/state/90e72e323ad23bd6013
ad23f63da0016?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following JSON was returned in the response body:

```
{
  "id": "90e72e323fe4c9ae013fe4def98a0125",
  "artifactType": "SERVICE_INSTANCE",
  "artifactState": "ACTIVE""state": "FAILED"
  "status":null>
}
```

Artifact views

Artifact views provide a convenient way to perform retrieve or update actions. When views are used with artifact GET and fastview GET requests, the response includes only the relevant information depending on the type of view requested. When used with PUT requests, the body of the request can include just relevant information, compared to typical PUT requests.

Advantages of Views

- With GET requests, views retrieve only the relevant data for the artifact and avoid loading all the data for the artifact. This leads to better performance.
- With PUT requests, the burden is not on the user to know all the artifact details to update the artifact. The user can pass only the necessary information.
- An example of using views is presented in "Update an artifact" on page 17 (using componentfinalize view).

Note: If you are updating with artifact views, you need to specify all the view's properties.

The following predefined views are available.

View name	Properties	Description
accesspoint	accessPoint.username accessPoint.password accessPoint.uri accessPoint.category.name	This retrieves information from the access point object. Any entity that has the accessPoint property should work with this view.
actioninfo	action.actionState.name action.actionStatus action.errorOnTimeout action.failOnError action.timeout action.processDefinition.name action.consumerVisible action.stateConstraint.lifecycleState.name action.stateConstraint.lifecycleSubstate.name	Retrieves the properties for the action object in addition to all the basicinfo and propertyinfo properties. Note: If the intention is to update an existing action, then action.id must be specified.

View name	Properties	Description
artifactinfo	state.name artifactType.name disabled ownedBy.name	Retrieves the required properties from an artifact object in addition to all the basicinfo properties.
basicinfo	id name displayName description iconUrl detailedDescription isCriticalSystemObject	Retrieves information from any Identity object. All artifacts and some additional entities (e.g., accessPoint) are identity objects.

View name	Properties	Description
candidatepools*	resourceBinding.candidateProvider.candidatePool.id resourceBinding.candidateProvider.candidatePool.objectId resourceBinding.candidateProvider.candidatePool.isCriticalSystemObject resourceBinding.candidateProvider.candidatePool.name resourceBinding.candidateProvider.candidatePool.displayName resourceBinding.candidateProvider.candidatePool.disabled resourceBinding.candidateProvider.candidatePool.useProviderEnv	Use this view to retrieve candidate resource provider pools for a given resource binding.
candidateproviders*	resourceBinding.candidateProvider.resourceProvider.id resourceBinding.candidateProvider.resourceProvider.objectId resourceBinding.candidateProvider.resourceProvider.isCriticalSystemObject resourceBinding.candidateProvider.resourceProvider.name resourceBinding.candidateProvider.resourceProvider.displayName resourceBinding.candidateProvider.resourceProvider.disabled	Use this view to retrieve candidate resource providers for a given resource binding.

View name	Properties	Description
componentchild	componentChild.name componentChild.displayName componentChild.description componentChild.iconUrl componentChild.detailedDescription componentChild.isCriticalSystemObject componentChild.ownedBy.name componentChild.state.name componentChild.artifactType.name componentChild.disabled componentChild.serviceInstance.id componentChild.componentType.name componentChild.lifecycleProperties.lifecycleState.name componentChild.lifecycleProperties.lifecycleSubstate.name componentChild.id	Use this view to add/update service components from the parent service component.
componentfinalize	toFinalize	Use this view to change the finalize flag for a component.
componentlifecycle	lifecycleProperties.lifecycleState.name lifecycleProperties.lifecycleSubstate.name	Use this view to change the lifecycle state and substate for a component.

View name	Properties	Description
componentroot	componentRoot.name componentRoot.displayName componentRoot.description componentRoot.iconUrl componentRoot.detailedDescription componentRoot.isCriticalSystemObject componentRoot.ownedBy.name componentRoot.state.name componentRoot.artifactType.name componentRoot.disabled componentRoot.serviceInstance.id componentRoot.componentType.name componentRoot.lifecycleProperties.lifecycleState.name componentRoot.lifecycleProperties.lifecycleSubstate.name componentRoot.id	Use this view to add a root service component to a ServiceBlueprint or a ServiceInstance object.
disabledesign	ServiceBlueprint.disabled	Use this view to enable or disable a ServiceBlueprint.
propertyinfo	property.name property.values.value property.valueType.name property.consumerVisible property.displayName	Retrieves a list of all the properties. Used for creating new properties.

View name	Properties	Description
propertyvalue	property.name property.values.value property.consumerVisible property.displayName	This view is useful if the need is only to update a property value.
resourcebindinginfo	resourceBinding.isCriticalSystemObject resourceBinding.state.name resourceBinding.artifactType.name resourceBinding.disabled resourceBinding.bindingState.name resourceBinding.resourceOffering.name resourceBinding.lifeCycleProperties.lifecycleState.name resourceBinding.lifeCycleProperties.lifecycleSubstate.name resourceBinding.resourceProvider.name	Use this view to create/update a Resource Binding.
validproviders*	resourceBinding.id resourceBinding.validProvider.resourceBinding.id resourceBinding.validProvider.resourceProvider.id	Use this view to retrieve or to update valid resource providers for a given resource binding.

View name	Properties	Description
validproviderspools*	resourceBinding.id resourceBinding.validProvider.resourceBinding.id resourceBinding.validProvider.resourceProvider.id resourceBinding.validProvider.validPool.id	Use this view to update valid resource providers and valid resource provider pools for a given resource binding.

* For the candidatepools, candidateproviders, validproviders and validproviderpools artifact views, while the provider selection is in progress the list of providers will be returned. Once a provider is selected no list will be returned.

Chapter 2: Availablevalues API

Description

Use this API to retrieve the list of available values for a dynamic property.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/availablevalues/<property_id>` *property_id* is an option model property, and is part of service design, offering and subscription artifacts.
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See Get userIdentifier for the steps required to get the userIdentifier value.
Request Body	Ampersand (&) separated name=value pairs, where the value on the left side of the equal sign (=) represents the parameter name configured for a dynamic property, and the value on the right side is the value the user selected from the parent property. For example, a request body might contain: *first=parent1value&countparam=mycount*.
Returns	200 - Ok 400 - Not authorized 404 - Not found 500 - Server exception

Example use context

From the subscriber portal a property is selected from a drop down list. The values of any associated dynamic properties must be spontaneously populated - they are dynamic and therefore cannot be populated in advance.

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/availablevalues/90e763a43ddc18e501
3ddc2f134c0088 ?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following was sent in the request body:

```
first=parent1Value
```

The following response was returned:

```
<Property>
  <id>90e763a43ddc18e5013ddc2f134c0088</id>
  <name>child1</name>
  <displayName>child1</displayName>
  <dynamicValueEnabled>true</dynamicValueEnabled>
  <dynamicScriptName>sample_jsp.jsp</dynamicScriptName>
  <dynamicScriptParameters>
    first=[CLIENT:parent1]
  </dynamicScriptParameters>
  <availableValues>
    <value>value1</value>
    <displayName>value1 displayName</displayName>
    <description>value1 description</description>
  </availableValues>
  ...
</Property>
```

Chapter 3: Catalog API

Description

Use this API to get information related to CSA catalogs.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

Catalog

A catalog is the collection of services available to a consumer.

URI	Method	Parameters	Description
/catalog	GET	userIdentifier, scope, detail	"List catalogs" on page 59
/catalog/<catalog_id>	GET	userIdentifier, scope, detail	"Get catalog details" on page 62

Category

Categories allow you to classify service offerings in a service catalog.

URI	Method	Parameters	Description
/catalog/<catalog_id>/category	POST	userIdentifier	"Create catalog categories" on page 65
/catalog/<catalog_id>/category/<category_id>	PUT	userIdentifier	"Update catalog categories" on page 67
/catalog/<catalog_id>/category/<category_id>	DELETE	userIdentifier	"Delete catalog category" on page 69

Offering

An offering allows you to define service designs that are published to a service catalog.

URI	Method	Parameters	Description
/catalog/<catalog_id>/offering	GET	userIdentifier, scope, detail, hasApproval	"List offerings in the catalog" on page 70
/catalog/<catalog_id>/offering/<offering_id>	GET	userIdentifier, scope, detail	"Get offering details" on page 72

Request

A request is created whenever a user initiates, changes, or deletes a subscription.

URI	Method	Parameters	Description
/catalog/<catalog_id>/request	GET	userIdentifier, scope, detail, submitter	"List requests in the catalog" on page 73
/catalog/<catalog_id>/request	POST	userIdentifier	"Submit a request" on page 76
/catalog/<catalog_id>/request/<request_id>	GET	userIdentifier, scope, detail	"Get request details" on page 79

URI	Method	Parameters	Description
/catalog/<catalog_id>/request/<request_id>/cancel	GET	userIdentifier	"Cancel a request" on page 82
/catalog/<catalog_id>/request/<request_id>	DELETE	userIdentifier	"Retire a request" on page 82

Approval

An approval is created when the approval manager approves a request.

URI	Method	Parameters	Description
/catalog/<catalog_id>/approval	GET	userIdentifier, scope, detail, approver, returnRetired	"List approvals in the catalog" on page 83
/catalog/<catalog_id>/approval/<approval_id>	GET	userIdentifier, scope, detail	"Get approval details" on page 85
/catalog/<catalog_id>/approval/<approval_id>	PUT	userIdentifier	"Update approval decision using an external approval system" on page 86
/catalog/<catalog_id>/approval/<approval_id>	DELETE	userIdentifier	"Retire an approval" on page 100
/catalog/<catalog_id>/approval/<approval_id>/approver	PUT	userIdentifier	"Update approval decision using CSA approval process" on page 87

Approval policy

URI	Method	Parameters	Description
/catalog/<catalog_id>/policy/<policy_id>/setCatalogApprovalPolicy	POST	userIdentifier	"Update catalog approval policies" on page 89
/catalog/<catalog_id>/policy/<policy_id>/setSOApprovalPolicy	POST	userIdentifier	"Update service offerings approval policy" on page 91

Subscription

A subscription is created when a consumer requests a service offering and includes all of the options selected by the consumer when the subscription was initiated.

URI	Method	Parameters	Description
/catalog/<catalog_id>/subscription	GET	userIdentifier, scope, detail, requestor	"List subscriptions in the catalog" on page 93
/catalog/<catalog_id>/subscription/<subscription_id>	GET	userIdentifier, scope, detail	"Get subscription details" on page 95

Resource Subscription

URI	Method	Parameters	Description
/catalog/<catalog_id>/resourceSubscription	GET	userIdentifier, scope, detail	"Get resource subscription details" on page 100

Instance

An instance is created when a request is approved and includes details about the requested services such as the status of services, IP addresses, etc.

URI	Method	Parameters	Description
catalog/<catalog_id>/instance	GET	userIdentifier, scope, detail, requestor	"List instances in the catalog" on page 97
catalog/<catalog_id>/instance/<instance_id>	GET	userIdentifier, scope, detail	"Get instance details" on page 99

List catalogs

Details

URI	`/catalog`
Method	GET

URI	`/catalog`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *basic*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/catalog
?userIdentifier=90d9652b35f46a930135f35b327e00a0&scope=base&detai
l=basic
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CatalogList>
    <count>12</count>
    <limit>0</limit>
    <catalog>
        <id>402895e566cb32ss0136cb831752000f</id>
        <objectId>402895e566cb32ss0136cb831752000f</objectId>
        <createdOn>2012-04-19T09:23:04.913-06:00</createdOn>
        <updatedOn>2012-04-19T09:23:04.913-06:00</updatedOn>
```

```
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <description>Default catalog for the
organization.</description>
        <name>consumer_catalog_a</name>
        <displayName>Consumer Catalog A</displayName>
        <state>
            <id>90d96588364da0c701370da0ss320037</id>
            <objectId>90d96588364da0c701370da0ss320037</objectId>
            <createdOn>2012-04-19T09:22:25.943-06:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>Active</description>
            <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
            <name>ACTIVE</name>
            <displayName>Active</displayName>
            <disabled>false</disabled>
            <categoryType>
                <id>90d96588364da0c701370da0ss320038</id>

<objectId>90d96588364da0c701370da0ss320038</objectId>

<isCriticalSystemObject>true</isCriticalSystemObject>
                <name>ARTIFACT_STATE</name>
                <displayName>Artifact State</displayName>
                <extensible>false</extensible>
            </categoryType>
        </state>
        <artifactType>
            <id>90d96588364da0c701370da0ss320039</id>
            <objectId>90d96588364da0c701370da0ss320039</objectId>
            <createdOn>2012-04-19T09:22:26.050-06:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>Catalog</description>
            <iconUrl>/csa/images/categories/artifact_
type/catalog.png</iconUrl>
            <name>CATALOG</name>
            <displayName>Catalog</displayName>
            <disabled>false</disabled>
            <categoryType>
```

```
                <id>90d96588364da0c701370da0ss320030</id>

<objectId>90d96588364da0c701370da0ss320030</objectId>

<isCriticalSystemObject>true</isCriticalSystemObject>
                <name>ARTIFACT_TYPE</name>
                <displayName>Artifact Type</displayName>
                <extensible>false</extensible>
            </categoryType>
        </artifactType>
        <disabled>false</disabled>
    </catalog>
    <catalog>
    ...
    </catalog>
...
</CatalogList>
```

Get catalog details

Details

URI	`/catalog/<catalog_id>` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

URI	`/catalog/<catalog_id>` Use "List catalogs" on page 59 to get the catalog ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/catalog/402895e566cb32ss0136cb831752000f
?userIdentifier=90d9652b35f46a930135f35b327e00a0&scope=base&detail=required
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Catalog>
    <id>402895e345cb67dd0136ss331752000f</id>
    <objectId>402895e345cb67dd0136ss331752000f</objectId>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>Catalog_Consumer_A</name>
```

```
        <displayName>Consumer Catalog A</displayName>
        <state>
            <id>90d96567360da0c701360ss0ef470038</id>
            <objectId>90d96567360da0c701360ss0ef470038</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>ACTIVE</name>
            <displayName>Active</displayName>
            <disabled>false</disabled>
            <categoryType>
                <id>90d67588360da0c701360ss0ef420037</id>
                <objectId>90d67588360da0c701360ss0ef420037</objectId>
                <isCriticalSystemObject>true</isCriticalSystemObject>
                <name>ARTIFACT_STATE</name>
                <displayName>Artifact State</displayName>
                <extensible>false</extensible>
            </categoryType>
        </state>
        <artifactType>
            <id>90d96586760da0c701360da0ssd2001d</id>
            <objectId>90d96586760da0c701360da0ssd2001d</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>CATALOG</name>
            <displayName>Catalog</displayName>
            <disabled>false</disabled>
            <categoryType>
                <id>90d67588360da0c701360da0ssb40017</id>
                <objectId>90d67588360da0c701360da0ssb40017</objectId>
                <isCriticalSystemObject>true</isCriticalSystemObject>
                <name>ARTIFACT_TYPE</name>
                <displayName>Artifact Type</displayName>
                <extensible>false</extensible>
            </categoryType>
        </artifactType>
        <disabled>false</disabled>
    </Catalog>
```

Create catalog categories

Details

URI	`/catalog/<catalog_id>/category`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

In the request body:

- Any category specified in the request body that already exists will be left unchanged.
- displayName is required.
- iconUrl and description are optional, and will be set to null if not specified.

Example

The following URL was sent to create an approval policy with two named approvers:

```
https://<host>:<port>/csa/rest/catalog/90e72e323c88421f013c8d7fad
120076/category ?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request:

```
<Catalog>
  <catalogCategory>
    <displayName>Example first Catalog Category</displayName>
    <iconUrl>/catalog/category/x.png</iconUrl>
    <description>description for catalog category</description>
  </catalogCategory>
```

```
  <catalogCategory>
      <displayName>Example second category</displayName>
  </catalogCategory>
</Catalog>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Catalog>
  <id>8a81818f3d02fb7e013d0308894a0004</id>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <description>Default catalog for the
organization.</description>
  <iconUrl>
    /csa/images/library/briefcase-consumer_default_img-60.png
  </iconUrl>
  <name>Catalog_QA_ORG</name>
  <displayName>QA Org Catalog</displayName>
  ...
  <catalogCategory> ... </catalogCategory>
  ...
  <catalogCategory>
    <id>8a81818f3d128500013d1341a5d3000e</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <name>EXAMPLE_SECOND_CATEGORY</name>
    <displayName>Example second category</displayName>
    <disabled>false</disabled>
  </catalogCategory>
  <catalogCategory> ... </catalogCategory>
  ...
  <catalogCategory>
    <id>8a81818f3d128500013d1341a5c6000d</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>description for catalog category</description>
    <iconUrl>/catalog/category/x.png</iconUrl>
    <name>EXAMPLE_FIRST_CATALOG_CATEGORY</name>
    <displayName>Example first Catalog Category</displayName>
    <disabled>false</disabled>
  </catalogCategory>
  <catalogCategory> ... </catalogCategory>
```

```
  ...
</Catalog>
```

Update catalog categories

Details

URI	`/catalog/<catalog_id>/category/<category_id>`
Method	PUT
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for details on categories updated.) 401 - Not authorized 404 - Object not found 500 - Server exception

Note that any CatalogCategory elements not specified in the request body will be left unchanged.

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/8a81818f3d02fb7e013d030889
4a0004/ category/8a81818f3d128500013d1341a5c6000d
?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request:

```
<CatalogCategory>
  <displayName>Changing first example category name</displayName>
  <iconUrl>/catalog/category/x.png</iconUrl>
  <description>New description for first example
```

```
category</description>
</CatalogCategory>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Catalog>
  <id>8a81818f3d02fb7e013d0308894a0004</id>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <description>Default catalog for the
organization.</description>
  <iconUrl>
    /csa/images/library/briefcase-consumer_default_img-60.png
  </iconUrl>
  <name>Catalog_QA_ORG</name>
  <displayName>QA Org Catalog</displayName>
  ...
  <catalogCategory> ... </catalogCategory>
  ...
  <catalogCategory>
    <id>8a81818f3d128500013d1341a5c6000d</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>description for catalog category</description>
    <iconUrl>/catalog/category/x.png</iconUrl>
    <name>CHANGING_FIRST_EXAMPLE_CATEGORY_NAME</name>
    <displayName>Changing first example category
name</displayName>
    <disabled>false</disabled>
  </catalogCategory>
  ...
</Catalog>
```

Delete catalog category

Details

URI	`/catalog/<catalog_id>/category/<category_id>`
Method	DELETE
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for details on category deleted.) 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/8a81818f3d02fb7e013d030889
4a0004/ category/8a81818f3d128500013d1341a5c6000d
?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The XML return content is basic catalog information as returned with the POST and PUT methods and most notably, will not include the category just deleted.

List offerings in the catalog

Details

URI	`/catalog/<catalog_id>/offering` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|view]`
	Optional; default is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`hasApproval=[true|false]`
	Optional; default is *false*. If *true*, then hasApproval attribute is returned. If *false*, then the attribute is not returned.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent to get a list of offerings in a catalog using the default values for scope (base) and detail (basic):

```
https://<host>:<port>/csa/rest/catalog/402895e33732af18013732b6f435006boffering?userIdentifier=90d9652b67ss6a930135f35b327e00a0
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceOfferingList>
  <count>3</count>
  <limit>0</limit>
  <ServiceOffering
    <id>90e763db3dd1a9a4013dd1e16aa16c95</id>
    <objectId>90e763db3dd1a9a4013dd1e16aa16c95</objectId>
    <createdOn>2013-04-03T14:50:43.873-07:00</createdOn>
    <updatedOn>2013-04-03T14:50:43.873-07:00</updatedOn>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>3 Level Option for testing service offerings
Integration Service Offering</description>
    </artifactType>
    <serviceBlueprint>
    <detailedDescription>3 Level Option for testing service
offerings Integration Service Offering</detailedDescription>
    <iconUrl>/csa/images/library/compliance.png</iconUrl>
    <name>ServiceOffering for ApprovalPolicy</name>
    <displayName>ServiceOffering for ApprovalPolicy</displayName>
    <state> … </state>
    <artifactType> … </artifactType>
    <disabled>false</disabled>
    <offeringState> … </offeringState>
  </ServiceOffering>
  <ServiceOffering> … </ServiceOffering>
  <ServiceOffering> … </ServiceOffering>
</ServiceOfferingList>
```

Get offering details

Details

URI	`/catalog/<catalog_id>/offering/<offering_id>` Use "Catalog API" on page 55 to get the catalog ID and "List offerings in the catalog" on page 70 to get the offering ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/catalog/402895e33732af18013732b6f4
35006b/ offering/402895e337326d300137327ce1e30074
?userIdentifier=90d9652b67ss6a930135f35b327e00a0
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceOffering>
    <id>402895e337326d300137327ce1e30074</id>
    <objectId>402895e337326d300137327ce1e30074</objectId>
    <createdOn>2012-05-09T10:44:34.147-06:00</createdOn>
    <updatedOn>2012-05-09T11:48:26.170-06:00</updatedOn>
        ...
</ServiceOffering>
```

List requests in the catalog

Deprecation Notice

The GET /catalog/<catalog_id>/request URI has been deprecated. Use URI /user/myrequest instead as using the deprecated URI will not allow viewing requests created by users who previously had access to the catalog, but no longer have access.

Details

URI	`/catalog/<catalog_id>/request` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

URI	`/catalog/<catalog_id>/request` Use "List catalogs" on page 59 to get the catalog ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|view]`
	Optional; default is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`submitter=<user_name>`
	Required; user name must be valid and must be authorized to view the request.
	
	`returnRetired=[true|false]`
	Optional; default is *false*.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://localhost:8444/csa/rest/catalog/8a8181853810699a0138106dc
ebc0011/request/ ?userIdentifier=8a8181853810699a01381076be5400a0
```

The following XML was returned in the response:

```
<ServiceRequestList>
   <count>21</count>
   <limit>0</limit>
   <ServiceRequest>
      <id>8a8181853810699a01381079190800a7</id>
      <objectId>8a8181853810699a01381079190800a7</objectId>
      <createdOn>2012-06-21T12:16:08.073-07:00</createdOn>
```

```
      <updatedOn>2012-06-21T12:16:50.787-07:00</updatedOn>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <description>SD2 Offering</description>
      <detailedDescription>desc - SD2
offering</detailedDescription>
      <iconUrl>/csa/images/library/application.png</iconUrl>
      <name>request 1</name>
      <displayName>request 1</displayName>
      <state>
         <id>90d96588360da0c701360da0ef470038</id>
         <objectId>90d96588360da0c701360da0ef470038</objectId>
         <createdOn>2012-06-21T11:51:43.267-07:00</createdOn>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <description>Active</description>
         <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
         <name>ACTIVE</name>
         <displayName>Active</displayName>
         <disabled>false</disabled>
         <categoryType>
            <id>90d96588360da0c701360da0ef420037</id>
            <objectId>90d96588360da0c701360da0ef420037</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>ARTIFACT_STATE</name>
            <displayName>Artifact State</displayName>
            <extensible>false</extensible>
         </categoryType>
      </state>
      ...
   </ServiceRequest>
</ServiceRequestList>
```

Submit a request

Details

URI	`/catalog/<catalog_id>/request` Use "List catalogs" on page 59 to get the catalog ID.
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/catalog/90540a9734f502880134f502c82e0011/request ?userIdentifier=90d9667ss5f46a930135f35b327e00a0
```

The following XML was sent in the request:

```
<ServiceRequest>
    <description>description - request BE </description>
    <name>Request BE - Order server</name>
    <displayName>Request BE</displayName>
    <artifactContext>
        <id>8a8181853824bc1d013824c3ae350078</id>
    </artifactContext>
    <requestedAction>
        <name>ORDER</name>
```

```
        <property>
            <name>START_DATE</name>
            <values>
                <value>2012-06-26T10:58:58.233-08:00</value>
            </values>
        </property>
        <property>
            <name>END_DATE</name>
            <values>
                <value>2012-06-30T10:58:58.233-08:00</value>
            </values>
        </property>
        <property>
            <name>SERVICE_NAME</name>
            <values>
                <value>SERVICE_NAME - request BE</value>
            </values>
        </property>
        <property>
            <name>SERVICE_DESCRIPTION</name>
            <values>
                <value>service_desc request BE</value>
            </values>
        </property>
        <property>
            <name>OPTION_MODEL</name>
            <values>
                <optionModel>
                    <name>SD2</name>
                    <optionSets>
                        <name>42FBA7E6-17EF-7B1A-C4DD-
0A5CB0246E55</name>
                        <options>
                            <name>D914556A-6F6A-1BE0-035E-
0A5CB021BE9B</name>
                            <property>
                                <name>INT</name>
                                <values>
                                    <value>33</value>
```

```
                                </values>
                            </property>
                            <property>
                                <name>BOOLEAN</name>
                                <values>
                                    <value>true</value>
                                </values>
                            </property>
                            <property>
                                <name>STR</name>
                                <values>
                                    <value>YY</value>
                                </values>
                            </property>
                        </options>
                    </optionSets>
                    <optionSets>
                        <name>D6A80E16-2977-4111-14B9-
0A5E5D5B2F56</name>
                        <options>
                            <name>18CA9979-7C9D-F5AC-06B5-
0A5E5D5B3B51</name>
                        </options>
                    </optionSets>
                </optionModel>
            </values>
        </property>
    </requestedAction>
</ServiceRequest>
```

The following XML was returned in the response:

```
<ServiceRequest>
    <id>8a818185382a26cc01382abf331c037e</id>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <disabled>false</disabled>
</ServiceRequest>
```

Get request details

Details

URI	`/catalog/<catalog_id>/request/<request_id>` Use "Catalog API" on page 55 to get the catalog ID and "List requests in the catalog" on page 73 to get the request ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	201 - Ok, object returned 401 - Not authorized 404 - Not found 500 - Server exception

Example

The following URL was sent:

```
https://localhost:8444/csa/rest/catalog/8a8181853810699a0138106dc
ebc0011/request/8a8181853810699a01381079190800a7/
?userIdentifier=8a8181853810699a01381076be5400a0
```

The following XML was returned:

```
<ServiceRequest>
   <id>8a8181853810699a01381079190800a7</id>
   <objectId>8a8181853810699a01381079190800a7</objectId>
   <createdOn>2012-06-21T12:16:08.073-07:00</createdOn>
   <updatedOn>2012-06-21T12:16:50.787-07:00</updatedOn>
   <createdBy>
      <id>8a8181853810699a01381076be5400a0</id>
      <objectId>8a8181853810699a01381076be5400a0</objectId>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>consumer1</name>
      <disabled>false</disabled>
   </createdBy>
   <updatedBy>
      <id>90d96588360da0c701360da0f1d5f483</id>
      <objectId>90d96588360da0c701360da0f1d5f483</objectId>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <name>admin</name>
      <displayName>admin</displayName>
      <disabled>false</disabled>
   </updatedBy>
   <isCriticalSystemObject>false</isCriticalSystemObject>
   <description>SD2 Offering</description>
   <detailedDescription>desc - SD2 offering</detailedDescription>
   <iconUrl>/csa/images/library/application.png</iconUrl>
   <name>request 1</name>
   <displayName>request 1</displayName>
   <catalogItem>
      <id>8a8181853810699a01381079202c00d7</id>
      <createdOn>2012-06-21T12:16:09.900-07:00</createdOn>
      <updatedOn>2012-06-21T12:16:09.900-07:00</updatedOn>
      <createdBy>
         <id>8a8181853810699a01381076be5400a0</id>
         <objectId>8a8181853810699a01381076be5400a0</objectId>
         <isCriticalSystemObject>false</isCriticalSystemObject>
         <name>consumer1</name>
```

```
            <disabled>false</disabled>
         </createdBy>
         ...
         <priceCategory>
            <id>90d9650a36897af90136897bc93e0016</id>
            <objectId>90d9650a36897af90136897bc93e0016</objectId>
            <createdOn>2012-06-21T11:51:43.633-07:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>SETUP</description>

   <iconUrl>/csa/images/categories/price/setup.png</iconUrl>
            <name>SETUP</name>
            <displayName>SETUP</displayName>
            <disabled>false</disabled>
            <categoryType>
               <id>90d9650a36897af90136897bc7c70014</id>
               <objectId>90d9650a36897af90136897bc7c70014</objectId>

   <isCriticalSystemObject>false</isCriticalSystemObject>
               <name>PRICE_CATEGORY</name>
               <displayName>Price Category</displayName>
               <extensible>false</extensible>
            </categoryType>
         </priceCategory>
         <fixedPrice>3000.0</fixedPrice>
         <unitPrice>0.0</unitPrice>
      </basePrice>
   </ServiceRequest>
```

Cancel a request

Details

URI	`/catalog/<catalog_id>/request/<request_id>/cancel` Use "Catalog API" on page 55 to get the catalog ID and "List requests in the catalog" on page 73 to get the request ID.
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Retire a request

Details

URI	`/catalog/<calalog_id>/request/<request_id>` Use "Catalog API" on page 55 to get the catalog ID and "List requests in the catalog" on page 73 to get the request ID.
Method	DELETE

URI	`/catalog/<calalog_id>/request/<request_id>` Use "Catalog API" on page 55 to get the catalog ID and "List requests in the catalog" on page 73 to get the request ID.
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

List approvals in the catalog

Deprecation Notice

The GET /catalog/<catalog_id>/approval URI has been deprecated. Use URI /user/myapproval instead as using the deprecated URI will not allow access to new functionality including the ability to list all approvals from all catalogs for a specified approver.

Details

URI	`/catalog/<catalog_id>/approval` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

URI	`/catalog/<catalog_id>/approval` Use "List catalogs" on page 59 to get the catalog ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree|view]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned. If the value is *view*, then the view parameter is required.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *basic*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.
	
	`approver=<user_name>`
	Optional; the name of the approver.
	
	`returnRetired=[true|false]`
	Optional; default is *false*.
	
	Caution: The users specified by `userIdentifier`and `approver`must be in the same organization.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Get approval details

Details

URI	`/catalog/<calalog_id>/approval/<approval_id>` Use "Catalog API" on page 55 to get the catalog ID and "List requests in the catalog" on page 73 to get the approval ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Update approval decision using an external approval system

Details

URI	`/catalog/<calalog_id>/approval/<approval_id>` Use "Catalog API" on page 55 to get the catalog ID and "List approvals in the catalog" on page 83 to get the approval ID.
Method	PUT
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Body	ApprovalProcess instance required in request body.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

The following URL was sent to approve a subscription request:

```
https://<host>:<port>/csa/rest/catalog/90540a9734f502880134f502c8
2e0011/ approval/65920b6356n204770943t567ss2r1503
?userIdentifier=90d9667ss5f46a930135f35b327e00a0
```

The following XML was sent in the request:

```
<ApprovalProcess>
    <approvalResult>
        <name>APPROVED</name>
```

```
    </approvalResult>
</ApprovalProcess>
```

The following URL was sent to reject a subscription request:

```
https://<host>:<port>/csa/rest/catalog/
```

The following XML was sent in the request:

```
<ApprovalProcess>
    <approvalResult>
        <name>REJECTED</name>
    </approvalResult>
    <approvalComment>comment</approvalComment>
</ApprovalProcess>
```

Update approval decision using CSA approval process

Details

URI	`/catalog/<calalog_id>/approval/<approval_id>/approver` Use "Catalog API" on page 55 to get the catalog ID and "List approvals in the catalog" on page 83 to get the approval ID.
Method	PUT
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Body	ApprovalProcess instance required in request body.

URI	`/catalog/<calalog_id>/approval/<approval_id>/approver` Use "Catalog API" on page 55 to get the catalog ID and "List approvals in the catalog" on page 83 to get the approval ID.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

The following URL was sent to approve a subscription request:

```
https://<host>:<port>/csa/rest/catalog/90540a9734f502880134f502c82e0011/ approval/65920b6356n204770943t567ss2r1503/approver?userIdentifier=90d9667ss5f46a930135f35b327e00a0
```

The following XML was sent in the request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Approver>
    <person>
        <userName>approver0@hp.com</userName>
        <organization>
            <name>CSA_CONSUMER</name>
        </organization>
    </person>
    <approverResult>
        <name>APPROVED</name>
    </approverResult>
</Approver>
```

The following URL was sent to reject a subscription request:

```
https://<host>:<port>/csa/rest/catalog/90540a9734f502880134f502c82e0011/ approval/65920b6356n204770943t567ss2r1503/approver?userIdentifier=90d9667ss5f46a930135f35b327e00a0
```

The following XML was sent in the request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Approver>
```

```
    <person>
        <userName>approver0@hp.com</userName>
        <organization>
            <name>CSA_CONSUMER</name>
        </organization>
    </person>
    <approverResult>
        <name>REJECTED</name>
    </approverResult>
    <approvalComment>comment</approvalComment>
</Approver>
```

Update catalog approval policies

Use this API to associate an approval policy with the specified catalog.

Details

URI	`/catalog/<calalog_id>/policy/<policy_id>/setCatalogApprovalPolicy` Use "Catalog API" on page 55 to get the catalog ID. Use "List organization's approval policies" on page 137 to get an organization's approval policy IDs.
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Response Body	The response body will be an ApprovalPolicyVO of base/full.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/8a81818f3d4251ed013d46c2b7
f602bc/
policy/8a81818f3d4251ed013d4259f57c0008/setCatalogApprovalPolicy?
 userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ApprovalPolicyList>
  <count>2</count>
  <limit>0</limit>
  <approvalPolicy>
    <id>8a81818f3d4251ed013d46cc859002c1</id>
    ...
    <approvalTemplate xsi:type="namedApproverApprovalTemplateVO">
     <id>8a81818f3d4251ed013d46cc31ed02be</id>
      <name>EXAMPLE_APPROVAL_POLICY_March 7, 2013 9:40:19 PM UTC
</name>
      ...
      <automaticApproval>false</automaticApproval>
      <automaticPeriodDuration>0</automaticPeriodDuration>
      <minApprovalRequired>0</minApprovalRequired>
      <approvalLevel>1</approvalLevel>
    </approvalTemplate>
    ...
    <catalog>
      <id>8a81818f3d4251ed013d46c2b7f602bc</id>
      ...
      <name>SOFTWARE_CATALOG</name>
      ...
    </catalog>
  </approvalPolicy>
  <approvalPolicy>
    <id>8a81818f3d4251ed013d46cc85e402c2</id>
      ...
    <approvalTemplate>
```

```
      ...
      <name>USER_CONTEXT_APPROVAL_TEMPLATE_QAORG</name>
      ...
      <automaticApproval>false</automaticApproval>
      <automaticPeriodDuration>0</automaticPeriodDuration>
      <minApprovalRequired>0</minApprovalRequired>
      <approvalLevel>1</approvalLevel>
    </approvalTemplate>
    ...>
    <catalog>
      <id>8a81818f3d4251ed013d46c2b7f602bc</id>
      ...
      <name>SOFTWARE_CATALOG</name>
      ...
    </catalog>
  </approvalPolicy>
</ApprovalPolicyList>
```

Update service offerings approval policy

Use this API to update the approval policy for multiple service offerings published in the specified catalog.

Details

URI	`/catalog/<calalog_id>/policy/<policy_id>/setCatalogSOApprovalPolicy` Use "Catalog API" on page 55 to get the catalog ID. Use "List offerings in the catalog" on page 70 to get a catalog's service offering IDs, and "Get offering details" on page 72 to get a service offering's policy ID.
Method	POST

URI	`/catalog/<calalog_id>/policy/<policy_id>/setCatalogSOApprovalPolicy` Use "Catalog API" on page 55 to get the catalog ID. Use "List offerings in the catalog" on page 70 to get a catalog's service offering IDs, and "Get offering details" on page 72 to get a service offering's policy ID.
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Request Body	The request body contains the list of service offering IDs that are to be updated.
Response Body	The response body will be a MessageListVO with success and failure messages.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/8a81818f3d4251ed013d46c2b7f602bc/
policy/8a81818f3d4251ed013d46cc8590012c/
setCatalogSOApprovalPolicy?
userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request body:

```
<ServiceOfferingList>
  <ServiceOffering>
    <id>8a81818f3d4251ed013d427c75e5005d<id>
  </ServiceOffering>
  <ServiceOffering>
    <id>8a81818f3d4251ed013d427c75e127c3<id>
  </ServiceOffering>
```

```
  <ServiceOffering>
    <id>12345<id>
  </ServiceOffering>
</ServiceOfferingList>
```

The following XML was returned:

```
<messageList>
  <messages>Updated approval policy of action of ORDER for
service offering with id
8a81818f3d4251ed013d427c75e5005d</messages>
  <messages>Updated approval policy of action of MODIFY_
SUBCRIPTION for service offering with id
8a81818f3d4251ed013d427c75e127c3</messages>
  <messages>Failed to set approval policy for service offering
with id 12345. The service offering is not found.</messages>
</messageList>
```

List subscriptions in the catalog

Deprecation Notice

The GET /catalog/<catalog_id>/subscription URI has been deprecated. Use URI /user/mysubscription instead as using the deprecated URI will not allow viewing subscriptions created by users who previously had access to the catalog, but no longer have access.

Details

URI	`/catalog/<catalog_id>/subscription` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

URI	`/catalog/<catalog_id>/subscription` Use "List catalogs" on page 59 to get the catalog ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	`scope=[base|view]`
	Optional; default is *base*.
	`detail=basic`
	Optional; The only valid value is *basic*.
	`requestor=<user_name>`
	Optional; user name must be valid and must be authorized to view the request.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/402895e33732af18013732b6f435006b/
subscription?userIdentifier=90d9652b67ss6a930135f35b327e00a0
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceSubscriptionList>
    <count>6</count>
    <limit>0</limit>
    <ServiceSubscription>
        <id>90d957ea3806fa7e013807acc79000b3</id>

<iconUrl>/csa/images/library/serviceOfferingDefault58.png</iconUrl>
```

```
        <name>MY SR</name>
        <displayName>MY SR</displayName>
        <state>...</state>
        <artifactType>
            <name>SUBSCRIPTION</name>
            ...
        </artifactType>
        <disabled>false</disabled>
        <serviceOffering>...</serviceOffering>
        <subscriptionStatus>...</subscriptionStatus>
        <initiatingServiceRequest>...</initiatingServiceRequest>
        ...
    </ServiceSubscription>
     ...
</ServiceSubscriptionList>
```

Get subscription details

Details

URI	`/catalog/<catalog_id>/subscription/<subscription_id>` Us "Catalog API" on page 55 to get the catalog ID and "List subscriptions in the catalog" on page 93 to get the subscription ID.
Method	GET

URI	`/catalog/<catalog_id>/subscription/<subscription_id>` Us "Catalog API" on page 55 to get the catalog ID and "List subscriptions in the catalog" on page 93 to get the subscription ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/402895e33732af18013732b6f4
35006b/ subscription/90d957ea3806fa7e013807acc79000b3
?userIdentifier=90d9652b67ss6a930135f35b327e00a0
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceSubscription>
    <id>90d957ea3806fa7e013807acc79000b3</id>
    <name>MY SR</name>
    <displayName>MY SR</displayName>
    <optionModel>...</optionModel>
```

```
    <serviceInstance>...</serviceInstance>
    <serviceOffering>...</serviceOffering>
    <subscriptionStatus>...</subscriptionStatus>
    <initiatingServiceRequest>...</initiatingServiceRequest>
    <basePrice xsi:type="recurrentPricingVO">...</basePrice>
    <basePrice xsi:type="initialPricingVO">...</basePrice>
    <totalPrice xsi:type="recurrentPricingVO">...</totalPrice>
    <totalPrice xsi:type="initialPricingVO">...</totalPrice>
    <pricingModel>...</pricingModel>
    <associatedRequest>
        <id>90e396c83a4cfd80013a4d06e29702bc</id>
        <name>Test Modify subscription</name>
    </associatedRequest>
    <associatedRequest>
        <id>90e396c83a4cfd80013a4d017e740003</id>
        <name>Test2 Modify subscription</name>
    </associatedRequest>
    ...
</ServiceSubscription>
```

List instances in the catalog

Deprecation Notice

The GET /catalog/<catalog_id>/instance URI has been deprecated. Use URI /user/myinstance instead as using the deprecated URI will not allow viewing instances created by users who previously had access to the catalog, but no longer have access.

Details

URI	`/catalog/<catalog_id>/instance` Use "List catalogs" on page 59 to get the catalog ID.
Method	GET

URI	`/catalog/<catalog_id>/instance` Use "List catalogs" on page 59 to get the catalog ID.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	`scope=[base|view]`
	Optional; default is *base*.
	`detail=basic`
	Optional; The only valid value is *basic*.
	`requestor=<user_name>`
	Optional; user name must be valid and must be authorized to view the request.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/catalog/402895e33732af18013732b6f4
35006b/ instance?userIdentifier=90d9652b67ss6a930135f35b327e00a0
```

The following XML was returned:

```
<ServiceInstanceList>
     <count>6</count>
     <limit>0</limit>
     <ServiceInstance>
        <id>90d957ea3806fa7e01380f957d11070a</id>
        <name>MYSD_June 5, 2012 5:19:51 PM UTC</name>
        <displayName>MYSD</displayName>
        <state></state>
        <serviceInstanceState>...</serviceInstanceState>
        ...
```

```
      </ServiceInstance>
      ...
<ServiceInstanceList>
```

Get instance details

Details

URI	`/catalog/<catalog_id>/instance/<instance_id>` Use "Catalog API" on page 55 to get the catalog ID and "List instances in the catalog" on page 97 to get the instance ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Retire an approval

Details

URI	`/catalog/<catalog_idid>/approval/<approval_id>`
Method	DELETE
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Caution: The approval is retired regardless of whether it was rejected or approved.

Get resource subscription details

Details

URI	`/catalog/<catalog_id>/resourceSubscription/<resource_subscription_id>`
Method	GET

URI	`/catalog/<catalog_id>/resourceSubscription/<resource_subscription_id>`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone|subtree]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the object and its first level children are returned. If value is *subtree*, then the object and all of its descendants are returned.
	
	`detail=[required|basic|standard|template|full]`
	Optional; default is *full*. See "Values for the detail parameter" on page 202. Some API calls do not support all possible values for this parameter.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Chapter 4: Export API

Description

Use this API to export a supported artifact as a content archive. Supported artifacts include resource environments, resource offerings, service designs, service offerings, and catalogs.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/export/<artifact_id>`
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Bad request 404 - Not found 500 - Server exception

Note that the directory the archive will be downloaded to is defined by the REST client or browser in use.

Artifact archives are structured as following.

- Resource environment archive contains:
  - Resource environment XML
  - Manifest XML

- Resource offering archive contains:
  - Resource offering XML
  - Icons used for customizing resource offering
  - Manifest.XML

- Service design archive contains:
  - Service design XML
  - Resource offering XMLs
  - Icons used for customizing resource offerings and service design
  - Dynamic option JSP files
  - Manifest XML

- Service offing archive contains:
  - Service offering XML
  - Service design XML
  - Resource offering XMLs
  - Icons used for customizing service offering, service design, and resource offerings
  - Dynamic option JSP files
  - Manifest XML

- Catalog archive contains:
  - Catalog XML
  - Service offering XMLs
  - Service design XMLs
  - Resource offering XMLs
  - Resource environment XMLs
  - Icons used for customizing catalog, service offerings, service designs, and resource offerings
  - Dynamic option JSP files
  - Manifest XML

- Component palette archive contains:
  - Component palette XML
  - Service component type XMLs
  - Service component template XMLs
  - Resource offering type XMLs
  - Artifact relationship XMLs
  - Icon used for customizing component palette, service component types and templates
  - Manifest XML

Example

The following URL was sent to export a catalog:

```
https://<host>:<port>/csa/rest/export/8a81818f3d02fb7e013d0308894
a0004 ?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following response headers were returned, and include the name of the downloaded archive zip file, CATALOG_QA_Org_Catalog_ 8a81818f3d02fb7e013d0308894a0004.zip:

```
Status Code: 200 OK
Cache-Control: must-revalidate, post-check=0, pre-check=0
Content-Disposition: attachment;filename="CATALOG_QA_Org_Catalog_
8a81818f3d02fb7e013d0308894a0004"
Content-Type: application/zip;charset=UTF-8
Date: Tue, 26 Feb 2013 00:09:36 GMT
Expires: 0
Pragma: public
Server: Apache-Coyote/1.1
Transfer-Encoding: chunked
content-transfer-encoding: binary
```

Chapter 5: Import API

Description

Use this API to import artifacts from a CSA content archive. CSA archives are created via the export REST API, the content archive tool, or the CSA management console. The import operation imports the primary artifact and all associated artifacts.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/import`
Method	POST

URI	`/import`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`file=<multipart_representation_of_service_offering_zip_file>`
	Required; type 'Multipart/Form-Data' CSA archive zip to be imported. Note that the directory the file will be imported from is defined by the REST client or browser in use. Typically the file parameter will not include a directory path.
	
	`update=[true|false]`
	Optional; default is false. If true, any existing artifacts of the same type and with the same name will be overwritten; otherwise the artifacts are created. Cannot be used with *updatePreserveExisting* parameter.
	
	`updatePreserveExisting=[true|false]`
	Optional; default is false. If true, any existing artifacts of the same type with the same name will be preserved (not overwritten) and renamed. Existing references to these artifacts will automatically use the renamed artifacts. New artifacts are created from the archive. Cannot be used with *update* parameter.
	
	`orgForCatalogImport=<organization_name>`
	Required for import of catalog archive; the name of the organization to be used when creating the imported catalog.
	
	`associateProviders=[true|false]`
	Optional; default is false. If true, resource providers in the archive are bound to existing resource offerings and resource environments of the same provider type and display name in the database.
	
	`validateType=[COMPONENT_PALETTE|CATALOG|SERVICE_OFFERING|SERVICE_BLUEPRINT|RESOURCE_`

URI	`/import`
	`OFFERING|RESOURCE_ENVIRONMENT]` Optional; if specified, content archive is verified to be of the specified type. ||
Returns	200 - Updated 400 - Bad request 404 - Not found 500 - Server exception

Caution: Component palette import is an update operation, and so *associateProviders* and *updatePreserveExisting* parameters will be ignored.

The following headers must be set when using this API to upload the content archive:

- Content-type: multipart/form-data
- Content-Disposition: form-data; name="file"
- Content-Type: application/octet-stream

Example

The following URL was sent to import the contents of the specified archive.

```
https://<host>:<port>/csa/rest/import?userIdentifier=90d96588360d
a0c701360da0f1d5f483&update=true&file=SERVICE_OFFERING_SO_ONE_
90cec2ff3c81b896013c81b8c1780097.zip
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ImportOperationResult>
  <additionalSummaryMessages>
    Service offering SO ONE imported successfully.
(Id=90cec2ff3dc1d73b013dc1df0b750028)
  </additionalSummaryMessages>
  <additionalSummaryMessages>
    Service design SD ONE imported successfully.
(Id=90cec2ff3dc1d73b013dc1df09db0018)
  </additionalSummaryMessages>
  <importResultRecord>
    ea933ec5-ad7b-42b6-b47c-965ff76f7693
```

```
    </importResultRecord>
    <importStatus>SUCCESS</importStatus>
    <importSummary>
      Import of Service offering archive successful.
    </importSummary>
  </ImportOperationResult>
```

Chapter 6: Importzip API

Deprecation Notice

The GET /importzip API has been deprecated. Use /import instead as using the deprecated API will not include new functionality.

Description

Use this API to import an artifact from a CSA artifact archive. CSA archives are created via the export REST API, the content archive tool, or the CSA management console. The import operation imports the primary artifact and all associated artifacts.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/importzip`
Method	POST

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`file=<file_name>`
	Required; multipart CSA archive zip to be imported. Note that the directory the file will be imported from is defined by the REST client or browser in use. Typically the file parameter will not include a directory path.
	
	`forceCreation=[true|false]`
	Optional; default is false. If true, any existing artifacts of the same type with the same name will be preserved (not overwritten) and renamed. Existing references to these artifacts will automatically use the renamed artifacts. New artifacts are created from the archive. Cannot be used with *update* parameter.
	
	Caution: Produces same results as *updatePreserveExisting*. Allows for backward compatibility.
	
	`update=[true|false]`
	Optional; default is false. If true, any existing artifacts of the same type and with the same name will be overwritten; otherwise the artifacts are created. Cannot be used with *updatePreserveExisting* parameter.
	
	`updatePreserveExisting=[true|false]`
	Optional; default is false. If true, any existing artifacts of the same type with the same name will be preserved (not overwritten) and renamed. Existing references to these artifacts will automatically use the renamed artifacts. New artifacts are created from the archive. Cannot be used with *update* parameter.
	
	`orgForCatalogImport=<organization_name>`
	Required for import of catalog archive; the name of the organization to be used when creating the imported catalog.
	
	`associateProviders=[true|false]`
	Optional; default is false. If true, resource providers in the archive are bound to existing resource offerings and resource

	environments of the same provider type and display name in the database. `validateType=[COMPONENT_PALETTE | CATALOG | SERVICE_OFFERING | SERVICE_DESIGN | RESOURCE_OFFERING | RESOURCE_ENVIRONMENT]` Optional; if specified, content archive is verified to be of the specified type.
Returns	200 - Updated 400 - Bad request 404 - Not found 500 - Server exception

Caution: Component palette import is an update operation, and so associateProviders and updatePreserveExisting parameters will be ignored.

The following headers must be set when using this API to upload the content archive:

- Content-type: multipart/form-data
- Content-Disposition: form-data; name="file"
- Content-Type: application/octet-stream

Example

The following URL was sent:

```
https://localhost:8444/csa/rest/importzip
       ?userIdentifier=90d96588360da0c701360da0f1d5f483&file=
       SERVICE_DESIGN_Simple_Compute_Linux_
90cec2023cc75f8a013cc7643ad00034.zip
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<importMessageList>

  <messages>Service design Simple Compute Linux imported successfully.

(Id=90cec2ff3dd11298013dd1139eb7004f)</messages>

  <messages>Resource offering SA Offering imported successfully.

(Id=90cec2ff3dd11298013dd113992d000d)</messages>
```

```
  <messages>Resource offering uCMDB Offering imported successfully.
(Id=90cec2ff3dd11298013dd113977d0008)</messages>
  <messages>Resource offering vCenter Offering imported successfully.
(Id=90cec2ff3dd11298013dd1139a4f0016)</messages>
</importMessageList>
```

Chapter 7: Import_result API

Use this API to view detailed result information from importing a content archive. See "Import API" on page 105 for details on importing a content archive.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/import_result/<importResultRecord_id>` importResultRecord IDs are returned by the "Import API" on page 105"Import API" on page 105
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	`format=HTML`
	Optional; returns results as HTML rather than the default XML

Returns	200 - Ok 400 - Bad request 404 - Not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/import_result/ea933ec5-ad7b-42b6-b47c-965ff76f7693
```

```
?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<ImportResult>
  <importResultLogEntry>
    <limit>0</limit>
    <artifactDescription>SO ONE</artifactDescription>
    <artifactDisplayName>SO ONE</artifactDisplayName>
    <artifactName>
      SO_ONE_February 11, 2013 7:08:05 PM UTC
    </artifactName>
    <artifactType>SERVICE_OFFERING</artifactType>
    <importOperation>Error</importOperation>
  </importResultLogEntry>
  <importResultLogEntry>
    <artifactDescription>SD ONE</artifactDescription>
    <artifactDisplayName>SD ONE</artifactDisplayName>
    <artifactName>SD_ONE_February 11, 2013 7:03:41 PM
UTC</artifactName>
    <artifactType>SERVICE_BLUEPRINT</artifactType>
    <importOperation>Error</importOperation>
  </importResultLogEntry>
  <importSummary>
    Import of Service offering archive successful.
  </importSummary>
  <status>SUCCESS</status>
  <timeOfImport>March 31, 2013 7:19:01 PM UTC</timeOfImport>
  <userInformation>admin</userInformation>
</ImportResult
```

Chapter 8: Lifecycle engine API

Description

Use this API to work with lifecycle actions.Base URL

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

URI	Method	Parameters	Description
/lifecycleengine	GET	userIdentifier, serviceInstanceId	"Get latest lifecycle execution record for a service instance" on page 117
/lifecycleengine/<lifecycle_action_id>	GET	userIdentifer	"Get details for a lifecycle execution record" on the next page
/lifecycleengine/execute	POST	userIdentifier	"Schedule lifecycle transition for service instance" on page 118

Get details for a lifecycle execution record

Details

URI	`/lifecycleengine/<lifecycle_action_id>`
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/lifecycleengine/90d96588360da0c701
360da0f25400c2 ?userIdentifier=90s96588670da0c701360da0f1d540a1
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LifecycleExecutionRecord>
    <callbackPending>false</callbackPending>
    <executionState>
        <id>90d96588360da0c701360da0f25400c2</id>
        <objectId>90d96588360da0c701360da0f25400c2</objectId>
        <isCriticalSystemObject>true</isCriticalSystemObject>
        <name>READY</name>
        <disabled>false</disabled>
        <categoryType>
            <id>90d96588360da0c701360da0f25200c1</id>
            <objectId>90d96588360da0c701360da0f25200c1</objectId>
```

```
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>LIFECYCLE_EXECUTION_STATE</name>
            <extensible>false</extensible>
        </categoryType>
    </executionState>
    <reverse>false</reverse>
    <serviceInstance>
        <id>90d96588372d758101372d75aeb90087</id>
        <objectId>90d96588372d758101372d75aeb90087</objectId>
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <disabled>false</disabled>
    </serviceInstance>
    <targetState>
        <id>90d96588360da0c701360da0f23700bb</id>
        <objectId>90d96588360da0c701360da0f23700bb</objectId>
        <isCriticalSystemObject>true</isCriticalSystemObject>
        <name>DEPLOYED</name>
        <disabled>false</disabled>
        <categoryType>
            <id>90d96588360da0c701360da0f21300ae</id>
            <objectId>90d96588360da0c701360da0f21300ae</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>LIFECYCLE_STATE</name>
            <extensible>false</extensible>
        </categoryType>
    </targetState>
</LifecycleExecutionRecord>
```

Get latest lifecycle execution record for a service instance

Details

URI	/lifecycleengine
Method	GET

URI	`/lifecycleengine`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`serviceInstanceId=<service_instance_id>`
	Required; the ID of the service instance.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Schedule lifecycle transition for service instance

Details

URI	`/lifecycleengine/execute`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Examples

Use the following URL:

```
https://<host>:<port>/csa/rest/lifecycleengine/execute
?userIdentifier=90s96588670da0c701360da0f1d540a1
```

The following XML was sent in the request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LifecycleExecutionRequest>
    <reverse>false</reverse>

<serviceComponentId>90d96588372d758101372d75aebb009f</serviceComp
onentId>

<serviceInstanceId>90d96588372d758101372d75aeb90087</serviceInsta
nceId>
    <targetStateName>DEPLOYED</targetStateName>
</LifecycleExecutionRequest>
```

Chapter 9: Login API

Description

Use this API to provide credentials for CSA REST APIs.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

URI	Method	Parameters	Description
/login/<organization_name>/<user_name>	GET	none	"Get userIdentifier" on the next page
/login/<organization_name>/userLookup	GET	userName	"Get userIdentifier for user name with slash" on page 122

See "orgInformation API" on page 152 for getting an organzation's credentials.

Get userIdentifier

Details

URI	`/login/<organization_name>/<user_name>/` Where <organization_name> and <user_name> are your credentials for logging in to HP Cloud Service Automation.
Method	GET
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Note: Unlike most API calls, the arguments (organization name and user name) are part of the URL path rather than parameters.

Caution: If the trailing slash is not included, a portion of the user name might be truncated when the user name includes non-alphanumeric characters. A best practice is to always include a trailing slash.

Example

To get the userIdentifier, we used the following URL:

```
https://<host>:<port>/csa/rest/login/MyOrganization/admin/
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<person>
    <id>90s96588670da0c701360da0f1d540a1</id> <!-- This is
userIdentifier -->
    ... <!-- Remaining XML is not relevant for this example. -->
</person>
```

The value for `userIdentifier` is the first <id> value returned in the XML.

Get userIdentifier for user name with slash

Details

URI	`/login/<organization_name>/userLookup`
Method	GET
Parameters	`userName=<user_name>` Required; the user name you want to use as credentials for CSA.
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Example

To get the userIdentifier, we used the following URL:

```
https://<host>:<port>/csa/rest/login/MyOrganization/userLookup
?userName=admin/domain
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<person>
    <id>90s96588670da0c701360da0f1d540a1</id> <!-- This is
userIdentifier -->
    ... <!-- Remaining XML is not relevant for this example. -->
</person>
```

The value for `userIdentifier` is the first <id> value returned in the XML.

Chapter 10: Notification API

Description

Use this API to retrieve the notification objects associated with <party_id>, or to send a notification to users or organizations.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

URI	Method	Parameters	Description
/notification/party/<party_id>	GET	userIdentifier, maxResults	"View list of notification objects" on the next page
/notification/party	POST	userIdentifier	"Send notification" on page 125

View list of notification objects

Details

URI	`/notification/party/<party_id>` Where the party ID is the UUID of a person, organization, or group. See How to find a party ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	`maxResults=<n>`
	Optional; where n is the number of notification objects to return. Notification objects are ordered by createdOn date with most recent first. By default all notification objects will be returned.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/notification/party/BFA0DB53DA414B9
0E04059106D1A24B5
?userIdentifier=BFA0DB53DA414B90E04059106D1A24B5
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NotificationList>
    <notification>
        <notifContentBody>Your service subscription for {0} is
```

```
now active. To view the details or make modifications,  go to the
{1} and click the Subscriptions tab.</notifContentBody>
        <notifSubject>Your subscription is now
active</notifSubject>

<notifactionId>90d957ea3806fa7e01380f95d38d073a</notifactionId>

<recepientId>BFA0DB53DA414B90E04059106D1A24B5</recepientId>
        <recepientName>consumer</recepientName>
        <senderContextArtifactName>CSA_
CONSUMER</senderContextArtifactName>

<senderContextArtifactTypeName>ORGANIZATION</senderContextArtifac
tTypeName>
        <tokens><tokenSequence>0</tokenSequence><value>my
request</value></tokens>
        <tokens><tokenSequence>1</tokenSequence><value>CSA
Consumer</value></tokens>
        ...
    </notification>
    ...
</NotificationList>
```

Send notification

Details

URI	`/notification/party/<party_id>`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.

URI	/notification/party/<party_id>
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Request body format

```
<Notification>
    <subject>Notification Subject goes in here</subject>
    <contentBody>Enter any text here, optionally including
tokens: token0 = {0}. Token1 = {1}</contentBody>
    <!-- Each recipient must have an id and type. Only PERSON and
ORGANIZATION are valid types. Notifications will be sent to valid
recipients and an error message returned for the invalid ones.
Response code 200 OK will be returned if there is at least one
valid recipient.
    -->
    <recipient>
        <id>UUID of the recipient</id>
        <type>PERSON</type>
    </recipient>
    <recipient>
        <id>UUID of organization</id>
        <type>ORGANIZATION</type>
    </recipient>
    <!-- tokens must be specified if your contentBody contains
tokens. No token validation is done.
     -->
    <tokens>
        <tokenSequence>0</tokenSequence>
        <value>Token 0 content</value>
    </tokens>
    <tokens>
        <tokenSequence>1</tokenSequence>
        <value>Token 1 content</value>
    </tokens>
</Notification>
```

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/notification/party?userIdentifier=
90cec3a03a93ef89013a93f07b880001
```

The following XML was sent in the request (note that the second recipient.id is bad):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Notification>
    <subject>Test Subject</subject>
    <contentBody>Test content body, token0 = {0}. Token1 = {1}
</contentBody>
    <recipient>
        <id>90cec3a03a93ef89013a93f07b880001</id>
        <type>PERSON</type>
    </recipient>
    <recipient>
        <id>bad org id</id>
        <type>ORGANIZATION</type>
    </recipient>
     <tokens>
        <tokenSequence>0</tokenSequence>
        <value>Token 0 test content</value>
    </tokens>
    <tokens>
        <tokenSequence>1</tokenSequence>
        <value>Token 1 test content</value>
    </tokens>
</Notification>
```

The following response header status code was returned: `200 OK`

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NotificationPostResponse>
    <count>1</count>
    <errorMessage>Please enter a valid value for the recipient
user id = bad org id</errorMessage>
```

```
    <notification>
        <notifContentBody>Test content body, token0 = {0}. Token1
= {1}</notifContentBody>
        <notifCreatedOn>2012-11-14T10:25:06.021-
07:00</notifCreatedOn>
        <notifSubject>Test Subject</notifSubject>

<notifactionId>90cec39c3ae64d82013afff3e9c3002c</notifactionId>
        </recepientArtifactTypeId>
            90d96588360da0c701360da0ef03002c
        </recepientArtifactTypeId>

<recepientArtifactTypeName>PERSON</recepientArtifactTypeName>
        <recepientEmailAddr>
            acctgconsumer@econ-csa.com
        </recepientEmailAddr>

<recepientId>90cec3a03a93ef89013a93f07b880001</recepientId>
        <recepientName>acctgconsumer</recepientName>
        <senderContextArtifactId>
            90cec3a03a93ef89013a93f07b880001
        </senderContextArtifactId>
        <senderContextArtifactTypeId>
            90d96588360da0c701360da0ef03002c
        </senderContextArtifactTypeId>
        <senderContextArtifactTypeName>
            PERSON
        </senderContextArtifactTypeName>
        <source>EXTERNAL</source>
        <tokens>
            <tokenSequence>1</tokenSequence>
            <value>Token 1 test content</value>
        </tokens>
        <tokens>
            <tokenSequence>0</tokenSequence>
            <value>Token 0 test content</value>
        </tokens>
    </notification>
</NotificationPostResponse>
```

Chapter 11: Organization API

Description

Use this API to view HP CSA organizations.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

URI	Method	Parameters	Description
/organization	GET	userIdentifier, scope, detail	"View a list of organizations" on the next page Note: The user identified by userIdentifier must have admin access.
/organization/<organization_id>	GET	userIdentifier, scope, detail	"View an organization" on page 134
/organization/<organization_id>/approvalPolicy	GET	userIdentifer	"List organization's approval policies" on page 137
/organization/<organization_id>/approvalPolicy	POST	userIdentifer	"Create approval policy" on page 139

URI	Method	Parameters	Description
/organization/<organization_id>/approvalPolicy/<policy_id>	PUT	userIdentifer	"Update approval policy" on page 144
/organization/<organization_id>/approvalPolicy/<policy_id>	DELETE	userIdentifer	"Delete approval policy" on page 146
/organization/accessPoint	GET	orgName	"Retrieve organization LDAP access point information" on page 148
/organization/offering	PUT	userIdentifier, queryType, newReleases	"List most requested, recently requested, or new offerings" on page 149

View a list of organizations

Details

URI	`/organization/`
Method	GET

URI	`/organization/`

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	Note: The user identified by userIdentifier must have admin access.
	
	`scope=base`
	Optional; the only valid value is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Example

The following URL was sent:

```
https://localhost:8444/csa/rest/organization/
?userIdentifier=90d96588360da0c701360da0f1d5f483&scope=base&detai
l=basic
```

The following XML was returned:

```
<OrganizationList>
  <count>3</count>
   <limit>0</limit>
   <organization>
      <id>8a8181853810699a0138106dcdd00003</id>
      <objectId>8a8181853810699a0138106dcdd00003</objectId>
      <createdOn>2012-06-21T12:03:47.920-07:00</createdOn>
      <updatedOn>2012-06-21T12:04:10.597-07:00</updatedOn>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <description>desc - CSA organization</description>
      <iconUrl>/csa/images/library/Earth_48.png</iconUrl>
      <name>CSA</name>
```

```
      <displayName>CSA</displayName>
      <state>
         <id>90d96588360da0c701360da0ef470038</id>
         <objectId>90d96588360da0c701360da0ef470038</objectId>
         <createdOn>2012-06-21T11:51:43.267-07:00</createdOn>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <description>Active</description>
         <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
         <name>ACTIVE</name>
         <displayName>Active</displayName>
         <disabled>false</disabled>
         <categoryType>
            <id>90d96588360da0c701360da0ef420037</id>
            <objectId>90d96588360da0c701360da0ef420037</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>ARTIFACT_STATE</name>
            <displayName>Artifact State</displayName>
            <extensible>false</extensible>
         </categoryType>
      </state>
      <artifactType>
         <id>90d96588360da0c701360da0eefc002a</id>
         <objectId>90d96588360da0c701360da0eefc002a</objectId>
         <createdOn>2012-06-21T11:51:43.253-07:00</createdOn>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <description>Organization</description>
         <iconUrl>/csa/images/categories/artifact_
type/organization.png</iconUrl>
         <name>ORGANIZATION</name>
         <displayName>Organization</displayName>
         <disabled>false</disabled>
         <categoryType>
            <id>90d96588360da0c701360da0eeb40017</id>
            <objectId>90d96588360da0c701360da0eeb40017</objectId>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <name>ARTIFACT_TYPE</name>
            <displayName>Artifact Type</displayName>
            <extensible>false</extensible>
```

```
            </categoryType>
         </artifactType>
         <disabled>false</disabled>
         <partyType>
            <id>90d96588360da0c701360da0eff1005d</id>
            <objectId>90d96588360da0c701360da0eff1005d</objectId>
            <createdOn>2012-06-21T11:51:43.293-07:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>Organization</description>
            <iconUrl>/csa/images/categories/party_
   type/organization.png</iconUrl>
            <name>ORGANIZATION</name>
            <displayName>Organization</displayName>
            <disabled>false</disabled>
            <categoryType>
               <id>90d96588360da0c701360da0efe20059</id>
               <objectId>90d96588360da0c701360da0efe20059</objectId>
               <isCriticalSystemObject>true</isCriticalSystemObject>
               <name>PARTY_TYPE</name>
               <displayName>Party Type</displayName>
               <extensible>false</extensible>
            </categoryType>
         </partyType>
         <businessRole>
            <id>90d96588360da0c701360da0f0020061</id>
            <objectId>90d96588360da0c701360da0f0020061</objectId>
            <createdOn>2012-06-21T11:51:43.300-07:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>Consumer</description>
            <iconUrl>/csa/images/categories/business_
   role/consumer.png</iconUrl>
            <name>CONSUMER</name>
            <displayName>Consumer</displayName>
            <disabled>false</disabled>
            <categoryType>
               <id>90d96588360da0c701360da0effd0060</id>
               <objectId>90d96588360da0c701360da0effd0060</objectId>
               <isCriticalSystemObject>true</isCriticalSystemObject>
               <name>BUSINESS_ROLE</name>
```

```
            <displayName>Business Role</displayName>
            <extensible>false</extensible>
         </categoryType>
      </businessRole>
   </organization>
   ...
   </organization>
</OrganizationList>
```

View an organization

Details

URI	`/organization/<organization_id>` "View a list of organizations" on page 130 to get the organization ID.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|baseplusone]`
	Optional; default is *base*. If value is *base*, then the object is returned. If value is *baseplusone*, then the base plus the first level children are returned. Note that *subtree* is **not** a valid value for this API call.
	
	`detail=[required|basic]`
	Optional; default is *basic*. See "Values for the detail parameter" on page 202.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Example

The following URL was sent:

```
https://localhost:8444/csa/rest/organization/8a8181853810699a0138
106dcdd00003/ ?userIdentifier=8a8181853810699a01381076be5400a0
```

The following XML was returned:

```
<Organization>
   <id>8a8181853810699a0138106dcdd00003</id>
   <objectId>8a8181853810699a0138106dcdd00003</objectId>
   <createdOn>2012-06-21T12:03:47.920-07:00</createdOn>
   <updatedOn>2012-06-21T12:04:10.597-07:00</updatedOn>
   <isCriticalSystemObject>false</isCriticalSystemObject>
   <description>desc - CSA organization</description>
   <iconUrl>/csa/images/library/Earth_48.png</iconUrl>
   <name>CSA</name>
   <displayName>CSA</displayName>
   <state>
      <id>90d96588360da0c701360da0ef470038</id>
      <objectId>90d96588360da0c701360da0ef470038</objectId>
      <createdOn>2012-06-21T11:51:43.267-07:00</createdOn>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Active</description>
      <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
      <name>ACTIVE</name>
      <displayName>Active</displayName>
      <disabled>false</disabled>
      <categoryType>
         <id>90d96588360da0c701360da0ef420037</id>
         <objectId>90d96588360da0c701360da0ef420037</objectId>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <name>ARTIFACT_STATE</name>
         <displayName>Artifact State</displayName>
         <extensible>false</extensible>
      </categoryType>
   </state>
   <artifactType>
```

```
      <id>90d96588360da0c701360da0eefc002a</id>
      <objectId>90d96588360da0c701360da0eefc002a</objectId>
      <createdOn>2012-06-21T11:51:43.253-07:00</createdOn>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Organization</description>
      <iconUrl>/csa/images/categories/artifact_
type/organization.png</iconUrl>
      <name>ORGANIZATION</name>
      <displayName>Organization</displayName>
      <disabled>false</disabled>
      <categoryType>
         <id>90d96588360da0c701360da0eeb40017</id>
         <objectId>90d96588360da0c701360da0eeb40017</objectId>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <name>ARTIFACT_TYPE</name>
         <displayName>Artifact Type</displayName>
         <extensible>false</extensible>
      </categoryType>
   </artifactType>
   <disabled>false</disabled>
   <partyType>
      <id>90d96588360da0c701360da0eff1005d</id>
      <objectId>90d96588360da0c701360da0eff1005d</objectId>
      <createdOn>2012-06-21T11:51:43.293-07:00</createdOn>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Organization</description>
      <iconUrl>/csa/images/categories/party_
type/organization.png</iconUrl>
      <name>ORGANIZATION</name>
      <displayName>Organization</displayName>
      <disabled>false</disabled>
      <categoryType>
         <id>90d96588360da0c701360da0efe20059</id>
         <objectId>90d96588360da0c701360da0efe20059</objectId>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <name>PARTY_TYPE</name>
         <displayName>Party Type</displayName>
         <extensible>false</extensible>
      </categoryType>
```

```
   </partyType>
   <businessRole>
      <id>90d96588360da0c701360da0f0020061</id>
      <objectId>90d96588360da0c701360da0f0020061</objectId>
      <createdOn>2012-06-21T11:51:43.300-07:00</createdOn>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Consumer</description>
      <iconUrl>/csa/images/categories/business_
role/consumer.png</iconUrl>
      <name>CONSUMER</name>
      <displayName>Consumer</displayName>
      <disabled>false</disabled>
      <categoryType>
         <id>90d96588360da0c701360da0effd0060</id>
         <objectId>90d96588360da0c701360da0effd0060</objectId>
         <isCriticalSystemObject>true</isCriticalSystemObject>
         <name>BUSINESS_ROLE</name>
         <displayName>Business Role</displayName>
         <extensible>false</extensible>
      </categoryType>
   </businessRole>
</Organization>
```

List organization's approval policies

Details

URI	`/organization/<organization_id>/approvalPolicy`
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.

URI	/organization/<organization_id>/approvalPolicy
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/organization/8a81818f3d1421e7013d1
423635a0003/
approvalPolicy?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ApprovalPolicyList>
  <UserContextApprovalTemplate>
    <id>8a81818f3d1421e7013d1423646e0008</id>
    <objectId>8a81818f3d1421e7013d1423646e0008</objectId>
    <createdOn>2013-02-25T17:34:56.623-08:00</createdOn>
    <updatedOn>2013-02-25T17:34:56.627-08:00</updatedOn>
    ...
    <name>USER_CONTEXT_APPROVAL_TEMPLATE_QA_ORG_1</name>
    <displayName>
      QA Org 1 User Context Approval Template
    </displayName>
    <state> ... </state>
    <artifactType> ... </artifactType>
    ...
    <automaticApproval>false</automaticApproval>
    <automaticPeriodDuration>0</automaticPeriodDuration>
    <minApprovalRequired>0</minApprovalRequired>
    <approvalType>
      <id>90d96588360da0c701360da0f0b40094</id>
      <objectId>90d96588360da0c701360da0f0b40094</objectId>
      <createdOn>2013-02-25T17:32:55.620-08:00</createdOn>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>User Context Template</description>
```

```
      <iconUrl>
        /csa/images/categories/approval_type/user_context_
template.png
      </iconUrl>
      <name>USER_CONTEXT_TEMPLATE</name>
      <displayName>User Context Template</displayName>
      <disabled>false</disabled>
      <categoryType>
        <id>90d96588360da0c701360da0f0ac0092</id>
        <objectId>90d96588360da0c701360da0f0ac0092</objectId>
        <isCriticalSystemObject>true</isCriticalSystemObject>
        <name>APPROVAL_TYPE</name>
        <displayName>Approval Type</displayName>
        <extensible>false</extensible>
      </categoryType>
    </approvalType>
    <approvalLevel>1</approvalLevel>
  </UserContextApprovalTemplate>
  <NamedApproverApprovalTemplate> ...
</NamedApproverApprovalTemplate>
  <NamedApproverApprovalTemplate> ...
</NamedApproverApprovalTemplate>
</ApprovalPolicyList>
```

Create approval policy

Details

URI	`/organization/<organization_id>/approvalPolicy`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.

URI	/organization/<organization_id>/approvalPolicy
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

The types of approval policies supported are: NamedApproverApprovalTemplate, NamedGroupApprovalTemplate, UserContextApprovalTemplate, and DelegatedApprovalTemplate.

In NamedApproverApprovalTemplate request body:

- displayName is required.
- approver is optional.
- minApprovalRequired is optional. If no approver is specified, defaults to 0. If any approver is specified, defaults to 1. Value cannot be greater than the number of approvers specified.
- automaticApproval is optional; if not present, default to false.If automaticApproval is true:
  - automaticPeriodDuration (in days) is optional; if not present, defaults to 0. Valid value is integer from 0 to 365.
  - automaticApprovalDecision is required; valid value is APPROVED or REJECTED.

Example request body:

```
<NamedApproverApprovalTemplate>
  <displayName>Name Ex Template 03</displayName>
  <approver><userName>consumer</userName></approver>
  ...
  <approver><userName>consumer04</userName></approver>
  <minApprovalRequired>1</minApprovalRequired>
  <automaticApproval>true</automaticApproval>
  <automaticPeriodDuration>12</automaticPeriodDuration>
  <automaticApprovalDecision>
    <name>REJECTED</name>
  </automaticApprovalDecision>
</NamedApproverApprovalTemplate>
```

In NamedGroupApprovalTemplate request body:

- displayName is required.
- group's distinguishedName is required.

- minApprovalRequired is optional and defaults 1. Value cannot be greater than the number of users in the group.
- automaticApproval is optional; if not present, defaults to false.If automaticApproval is true:
  - automaticPeriodDuration (in days) is optional; if not present, defaults to 0. Valid value is integer from 0 to 365.
  - automaticApprovalDecision is required; valid value is APPROVED, or REJECTED.

Example request body:

```
<NamedGroupApprovalTemplate>
  <displayName>Name Group Template 01</displayName>
  <group>
       <distinguishedName>
              cn=ServiceConsumer,ou=ConsumerGroup,ou=CSAGroups
       </distinguishedName>
  </group>
  <minApprovalRequired>0</minApprovalRequired>
  <automaticApproval>true</automaticApproval>
  <automaticApprovalDecision><automaticApprovalDecision>
    <name>REJECTED</name>
  </automaticApprovalDecision>
</NamedGroupApprovalTemplate>
```

In UserContextApprovalTemplate request body:

- displayName is required.
- automaticApproval is optional; if not present, defaults to false.If automaticApproval is set to true:
  - automaticPeriodDuration (in days) is optional; if not present, defaults to 0. Valid value is integer from 0 to 365.
  - automaticApprovalDecision is required; valid value is APPROVED, or REJECTED.

Example request body:

```
<UserContextApprovalTemplate>
  <displayName>User ContextTemplate 01</displayName>
  <automaticApproval>true</automaticApproval>
  <automaticPeriodDuration>7</automaticPeriodDuration>
  <automaticApprovalDecision>
    <name>APPROVED</name>
```

```
  </automaticApprovalDecision>
</UserContextApprovalTemplate>
```

In DelegatedApprovalTemplate request body:

- displayName is required.
- processDefinition is required, id is needed.
- automaticApproval is optional; if not present, defaults to false. If automaticApproval is set to true:
  - automaticPeriodDuration (in days) is optional; if not present, defaults to 0. Valid value is integer from 0 to 365.
  - automaticApprovalDecision is required; valid value is APPROVED, or REJECTED.

Example request body:

```
<DelegatedApprovalTemplate>
  <displayName>Name Ex Template 03</displayName>
  <processDefinition>
    <id>90d96507362a000a01362a0048bc00ad</id>
  </processDefinition>
  <automaticApproval>true</automaticApproval>
  <automaticPeriodDuration>30</automaticPeriodDuration>
  <automaticApprovalDecision>
    <name>REJECTED</name>/
  </automaticApprovalDecision>
</DelegatedApprovalTemplate>
```

Example

The following URL was sent to create an approval policy with two named approvers:

```
https://<host>:<port>/csa/rest/organization/8a81818f3d1421e7013d1
423635a0003/
approvalPolicy?userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request:

```
<NamedApproverApprovalTemplate>
  <displayName>My-New-Approval-Template</displayName>
  <approver><userName>ProjectManager</userName></approver>
  <approver><userName>BudgetManager</userName></approver>
  <minApprovalRequired>2</minApprovalRequired>
```

```
  <automaticApproval>true</automaticApproval>
  <automaticPeriodDuration>14</automaticPeriodDuration>
  <automaticApprovalDecision>
    <name>APPROVED</name>
  </automaticApprovalDecision>
</NamedApproverApprovalTemplate>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NamedApproverApprovalTemplate>
  <id>8a81818f3d1437e2013d17f249c30a0c</id>
  <objectId>8a81818f3d1437e2013d17f249c30a0c</objectId>
  <createdOn>2013-02-26T11:19:47.397-08:00</createdOn>
  <updatedOn>2013-02-26T11:19:47.440-08:00</updatedOn>
  <createdBy> ... </createdBy>
  <updatedBy> ... </updatedBy>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <name>
    My-New-Approval-Template_February 26, 2013 7:19:47 PM UTC
  </name>
  <displayName>My-New-Approval-Template</displayName>
  <state> ... </state>
  <artifactType>
    <id>90d96588360da0c701360da0eef20027</id>
    <objectId>90d96588360da0c701360da0eef20027</objectId>
    <createdOn>2013-02-25T17:32:55.667-08:00</createdOn>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <description>Approval Template</description>
    ...
  </artifactType>
  <ownedBy> ... </ownedBy>
  <disabled>false</disabled>
  <automaticApproval>true</automaticApproval>
  <automaticPeriodDuration>14</automaticPeriodDuration>
  <minApprovalRequired>2</minApprovalRequired>
  <approvalType>
    <id>90d96588360da0c701360da0f0b00093</id>
    <objectId>90d96588360da0c701360da0f0b00093</objectId>
    <createdOn>2013-02-25T17:32:55.620-08:00</createdOn>
```

```
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <description>Named Approver Template</description>
    ...
  </approvalType>
  <automaticApprovalDecision> ... </automaticApprovalDecision>
  <approver>
    <id>8a81818f3d1437e2013d17eec74408d8</id>
    ...
    <userName>BudgetManager</userName>
    ...
  </approver>
  <approver>
    <id>8a81818f3d1437e2013d177b13f107dd</id>
    ...
    <userName>ProjectManager</userName>
    ...
  </approver>
</NamedApproverApprovalTemplate>
```

Update approval policy

Details

URI	`/organization/<organization_id>/approvalPolicy/<policy_id>`
Method	PUT
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/organization/8a81818f3d1421e7013d1
423635a0003/ approvalPolicy/8a81818f3d1437e2013d17f249c30a0c?
userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was sent in the request. It will change the list of approvers to user ProjectManager, set automaticApproval to false, and set automaticApprovalDecision to REJECTED.

```
<NamedApproverApprovalTemplate>
  <displayName>Updated-My-Approval-Template</displayName>
  <approver><userName>ProjectManager</userName></approver>
  <minApprovalRequired>0</minApprovalRequired>
  <automaticApproval>false</automaticApproval>
  <automaticPeriodDuration>0</automaticPeriodDuration>
  <automaticApprovalDecision>
    <name>REJECTED</name>
  </automaticApprovalDecision>
</NamedApproverApprovalTemplate>
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NamedApproverApprovalTemplate>
  <id>8a81818f3d1437e2013d17f249c30a0c</id>
  <objectId>8a81818f3d1437e2013d17f249c30a0c</objectId>
  <createdOn>2013-02-26T11:19:47.397-08:00</createdOn>
  <updatedOn>2013-02-26T11:45:06.900-08:00</updatedOn>
  ...
  <name>
    My-New-Approval-Template_February 26, 2013 7:19:47 PM UTC
  </name>
  <displayName>Updated-My-Approval-Template</displayName>
  ...
  <automaticApproval>false</automaticApproval>
  <automaticPeriodDuration>0</automaticPeriodDuration>
  <minApprovalRequired>1</minApprovalRequired>
  <approvalType>
```

```
    <id>90d96588360da0c701360da0f0b00093</id>
    ...
    <name>NAMED_APPROVER_TEMPLATE</name>
    ...
  </approvalType>
  <automaticApprovalDecision>
    <id>90d96588360da0c701360da0f091008c</id>
    ...
    <name>REJECTED</name>
    <displayName>Denied</displayName>
    ...
  </automaticApprovalDecision>
  <approver>
    <id>8a81818f3d1437e2013d177b13f107dd</id>
    ...
    <name>ProjectManager</name>
    ...
    <organization>
      <id>8a81818f3d1421e7013d1423635a0003</id>
      <objectId>8a81818f3d1421e7013d1423635a0003</objectId>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>QA_ORG_1</name>
      <displayName>QA Org 1</displayName>
      <disabled>false</disabled>
    </organization>
  </approver>
</NamedApproverApprovalTemplate>
```

Delete approval policy

Details

URI	/organization/<organization_id>/approvalPolicy/<policy_id>
Method	DELETE

URI	`/organization/<organization_id>/approvalPolicy/<policy_id>`
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 404 - Object not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/organization/8a81818f3d1421e7013d1
423635a0003/ approvalPolicy/8a81818f3d1437e2013d17f249c30a0c?
userIdentifier=90d96588360da0c701360da0f1d5f483
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NamedApproverApprovalTemplate>
  <id>8a81818f3d1437e2013d17f249c30a0c</id>
  <objectId>8a81818f3d1437e2013d17f249c30a0c</objectId>
  <createdOn>2013-02-26T11:19:47.397-08:00</createdOn>
  <updatedOn>2013-02-26T11:45:06.900-08:00</updatedOn>
    ...
  <name>
    My-New-Approval-Template_February 26, 2013 7:19:47 PM UTC
  </name>
  <displayName>Updated-My-Approval-Template</displayName>
  <state>
    <id>90d96588360da0c701360da0ef4b0039</id>
    <objectId>90d96588360da0c701360da0ef4b0039</objectId>
    <createdOn>2013-02-25T17:32:55.547-08:00</createdOn>
    <isCriticalSystemObject>true</isCriticalSystemObject>
    <description>Retired</description>
    <iconUrl>/csa/images/categories/artifact_
```

```
state/retired.png</iconUrl>
    <name>RETIRED</name>
    <displayName>Retired</displayName>
    <disabled>false</disabled>
    <categoryType> ... </categoryType>
  </state>
  ...
</NamedApproverApprovalTemplate>
```

Retrieve organization LDAP access point information

Details

URI	`/organization/accessPoint`
Method	GET
Parameters	`orgName=<organization_name>` Required; the name of the organization for which the LDAP access point information is to be retrieved.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/organization/accessPoint?orgName=ACCOUNTING
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessPointInfo>
  <password>JfyPejrtsQc0J4bKDPmznlg==</password>
  <searchSubtree>false</searchSubtree>
```

```
    <uri>ldap://localhost:389/dc=csa,dc=org</uri>
    <userSearchBase>
      ou=eConsumerUsers,ou=EnterpriseUsers
    </userSearchBase>
    <userSearchFilter>uid={0}</userSearchFilter>
    <username>cn=admin,dc=csa,dc=org</username>
</AccessPointInfo>
```

List most requested, recently requested, or new offerings

Details

URI	`/organization/offering` Organization is determiend by userIdentifier.
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`queryType=[mostRequested|recentlyRequested|newReleases]`
	Optional; the ten most requested, most recently requested, or newly released offerings in all catalogs for the organization are returned.

Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/organization/offering?userIdentifier=90e763db3ed8fe91013ed90155e600b0&queryType=mostRequested
```

The following XML was returned in the response:

```
<ServiceOfferingList>
   <count>1</count>
   <limit>0</limit>
   <ServiceOffering>
     <id>90e763db3ed8fe91013ed9017c4e0264</id>
     <createdOn>2013-05-24T17:05:53.358-07:00</createdOn>
     <updatedOn>2013-05-24T17:06:05.287-07:00</updatedOn>
     <isCriticalSystemObject>false</isCriticalSystemObject>
     <description>Ready to do more today? Standardize on Red Hat
Enterprise Linux and get more of everything you expect from an
enterprise-class open source operating system with flexibility,
efficiency and control.</description>
     <detailedDescription>Ready to do more today? Standardize on
Red Hat Enterprise Linux and get more of everything you expect
from an enterprise-class open source operating system with
flexibility, efficiency and control.</detailedDescription>
     <iconUrl>/csa/images/offerings/Microsoft-Frontpage-
icon.png</iconUrl>
     <name>Enterprise Red Hat Linux Server</name>
     <displayName>Enterprise Red Hat Linux Server</displayName>
     <catalogItem>
       <id>90e763db3ed8fe91013ed901a9ae047d</id>
       <catalog>
         <id>90e763db3ed8fe91013ed90156b100b4</id>
         <isCriticalSystemObject>false</isCriticalSystemObject>
         <disabled>false</disabled>
       </catalog>
       <category>
         <id>90d96588360da0c701360da0f4cc00ec</id>
         <isCriticalSystemObject>false</isCriticalSystemObject>
         <name>APPLICATION_SERVERS</name>
         <displayName>Application Servers</displayName>
         <disabled>false</disabled>
       </category>
     </catalogItem>
     …
     <catalogItem>…</catalogItem>
     <state>
```

```
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <disabled>false</disabled>
      </state>
      <artifactType>
         <id>90d96588360da0c701360da0eecb001b</id>
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <disabled>false</disabled>
      </artifactType>
      <disabled>false</disabled>
    </ServiceOffering>
  </ServiceOfferingList>
```

Chapter 12: orgInformation API

Description

Use this API to get an organization's credentials.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/orgInformation/<organization name>`
Method	GET
Returns	200 - Ok 404 - Not found 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/orgInformation/my_organization
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Organization>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>My Organization</description>
    <name>my_organization</name>
    <disabled>false</disabled>
</Organization>
```

Chapter 13: Processinstances API

Description

The Processinstance API is used to return execution results from HP Operations Orchestration flows. When HP Operations Orchestration makes one of these REST calls, the process instance properties are updated accordingly. When the process completes, the process notification handler passes these properties to the caller.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

URI	Method	Parameters	Description
/processinstances/<process_instance_id>	GET	None	"Retrieve a process instance" on page 155
/processinstances	POST	userIdentifier	"Create a process instance" on page 157
/processinstances/<process_instance_id>	PUT	userIdentifier, scope, view, action	"Update a process instance" on page 159
/processinstances/<process_instance_id>/execute	POST	userIdentifie	"Execute a process instance" on page 162

Process Instance structure

XML structure

An abbreviated version of the XML for a process instance is provided below. Only properties that are relevant for APIs are shown; the XML returned by an API call will usually include much more than what is shown here.

```
<ProcessInstance>>
   <property>
     <isCriticalSystemObject></isCriticalSystemObject>
     <name></name>
     <paramRoleType>
       <isCriticalSystemObject></isCriticalSystemObject>
       <name></name>
       <disabled></disabled>
     </paramRoleType>
     <scope>
       <isCriticalSystemObject></isCriticalSystemObject>
       <name></name> </scope>
     <valueType>
       <isCriticalSystemObject></isCriticalSystemObject>
       <name></name>
       <disabled></disabled>
     </valueType>
     <values>
       <value></value>
     </values>
     <maxOccurs></maxOccurs>
     <minOccurs></minOccurs>
     <orderIndex></orderIndex>
     <confidential></confidential>
     <encrypted></encrypted>
     <readOnly></readOnly>
   </property>
   <processDefinition>
     <id></id>
     <name></name>
```

```
   </processDefinition>
   <processInstanceState>
     <isCriticalSystemObject></isCriticalSystemObject>
     <name></name>
     <disabled></disabled>
   </processInstanceState>
   <context></context>
 </ProcessInstance>
```

Retrieve a process instance

Details

URI	`/processinstances/<process_instance_id>` Where <process_instance_id> is the process instance ID.
Method	GET
Parameters	None
Response Body	ProcessInstance VO
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

The following was sent to retrieve a process instance:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b3752ad4f0
13752ae38cb0065
```

The following XML was sent in the response:

```
<ProcessInstance>>
   <property>
     <isCriticalSystemObject>false</isCriticalSystemObject>
     <name>Unit Test Process Instance Property</name>
     <paramRoleType>
       <isCriticalSystemObject>false</isCriticalSystemObject>
```

```
        <name>INPUT</name>
        <disabled>false</disabled>
      </paramRoleType>
      <valueType>
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <name>INPUT</name>
        <disabled>false</disabled>
      </valueType>
      <values>
        <value>Unit Test Process Instance Property Value</value>
      </values>
      <maxOccurs>0</maxOccurs>
      <minOccurs>0</minOccurs>
      <orderIndex>0</orderIndex>
      <confidential>false</confidential>
      <encrypted>false</encrypted>
      <readOnly>false</readOnly>
    </property>
    <processDefinition>
      <id>90d9652b35f41cbc0135f41cf1310004</id>
      <name>Unit Test Process Definition 1</name>
    </processDefinition>
    <processInstanceState>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>INITIALIZED</name>
      <disabled>false</disabled>
    </processInstanceState>
    <context>Context for Unit Test Process Instance</context>
    <artifactId>ID of the artifact executing the
  action</artifactId>
    <timeout>3600</timeout>
    <errorOnTimeout>false</errorOnTimeout>
   </ProcessInstance>
```

Create a process instance

Details

URI	`/processinstances`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Request Body	ProcessInstance VO
Response Body	ProcessInstance VO
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

The following must be provide in the request body:

- Process definition ID.
- Context string. Includes contextual information that is relevant for the caller after the process instance has been created.
- Artifact ID.
- Properties. The properties provided here will be merged with the process definition properties.

Two implicit tokens (reserved flow input variable names) are provided to process instances executed through the HP Operations Orchestration execution engine.

- CSA_PROCESS_ID - will be provided the value of the UUID of the process instance.
- CSA_CONTEXT_ID - will be provided the value of the Artifact ID. This is the artifact for which this process instance executes.

The following was sent to create a process instance:

```
https://<host>:<port>/csa/rest/processinstances/?userIdentifier=9
0d9652b35f35a930135f35b327e00a0
```

The following XML was sent in the request:

```
<ProcessInstance>>
   <property>
     <isCriticalSystemObject>false</isCriticalSystemObject>
     <name>Unit Test Process Instance Property</name>
     <paramRoleType>
       <isCriticalSystemObject>false</isCriticalSystemObject>
       <name>INPUT</name>
       <disabled>false</disabled>
     </paramRoleType>
     <valueType>
       <isCriticalSystemObject>false</isCriticalSystemObject>
       <name>STRING</name>
       <disabled>false</disabled>
     </valueType>
     <values>
       <value>Unit Test Process Instance Property Value</value>
     </values>
     <maxOccurs>0</maxOccurs>
     <minOccurs>0</minOccurs>
     <orderIndex>0</orderIndex>
     <confidential>false</confidential>
     <encrypted>false</encrypted>
     <readOnly>false</readOnly>
   </property>
   <processDefinition>
     <id>90d9652b35f41cbc0135f41cf1310004</id>
     <name>Unit Test Process Definition 1</name>
   </processDefinition>
   <processInstanceState>
     <isCriticalSystemObject>false</isCriticalSystemObject>
     <name>INITIALIZED</name>
     <disabled>false</disabled>
   </processInstanceState>
   <context>Context for Unit Test Process Instance</context>
   <artifactId>ID of the artifact executing the
action</artifactId>
```

```
  <timeout>3600</timeout>
  <errorOnTimeout>false</errorOnTimeout>
</ProcessInstance>
```

Update a process instance

Details

URI	`/processinstances/<process_instance_id>` Where <process_instance_id> is the process instance ID.
Method	PUT

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`view=<view_type>&scope=view`
	Optional; used to update process instance based on pre-defined views. Possible values for *view_type* are `propertyinfo` and `processinstancestate`. With `processinstancestate`, the process instance state, return code and status can be changed.
	
	`<property_>action<_>=merge`
	Optional; use the *merge* option with the *action* meta tag query parameter to update only a portion of the process instance. The following the use cases are allowed:
	• With `view=propertyinfo`, specifying `property_action_=merge` will merge the properties specified in the request body with the existing process instance properties.
	• With `view=processinstancestate`, specifying `action=merge` will merge the state information specified in the request body with the existing process instance state.

Request Body	ProcessInstance VO

URI	/processinstances/<process_instance_id> Where <process_instance_id> is the process instance ID.
Response Body	Updated ProcessInstance VO
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

Process instance state (processInstanceState) values:

- INITIALIZED
- PENDING
- READY
- ACTIVE
- COMPLETED
- ERROR
- CANCELED

Process instance return code (processReturnCode) values:

- SUCCESS
- FAILURE
- RUNNING
- TIMEOUT

Examples

The following was sent to update propertyinfo of a process instance:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b362d4ecd0
1362d4fb7be0f71?userIdentifier=90d9652b362d4ecd01362d4ef51e00a5&v
iew=propertyinfo&scope=view&property_action_=merge
```

The following XML was sent in the request to update a property:

```
<ProcessInstance>>
   <id>90d9652b362d4ecd01362d4fb7be0f71</id>
   <property>
     <name>Property Name</name>
     <valueType>
       <name>STRING</name>
```

```
    </valueType>
    <values>
      <value>Hello World!</value>
    </values>
  </property>
 </ProcessInstance>
```

The following was sent to update the process instance state:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b362d4ecd0
1362d4fb7be0f71?userIdentifier=90d96588360da0c701360da0f1d5f483&s
cope=view&view=processinstancestate&action=merge
```

```
<ProcessInstance>>
   <id>90d9652b3752ad4f013752ae38cb0065</id>
   <processInstanceState>
     <name>COMPLETED</name>
   </processInstanceState>
 </ProcessInstance>
```

The following was sent to update the process instance return code:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b3752ad4f0
13752ae38cb0065?userIdentifier=90d96588360da0c701360da0f1d5f483&s
cope=view&view=processinstancestate&action=merge
```

```
<ProcessInstance>>
   <id>90d9652b3752ad4f013752ae38cb0065</id>
   <processReturnCode>
     <name>SUCCESS</name>
   </processReturnCode>
 </ProcessInstance>
```

The following was sent to update the process instance state, status and return code:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b3752ad4f0
13752ae38cb0065?userIdentifier=90d96588360da0c701360da0f1d5f483&s
cope=view&view=processinstancestate&action=merge
```

```
<ProcessInstance>>
   <id90d9652b3752ad4f013752ae38cb0065</id>
   <processInstanceState>
     <name>COMPLETED</name>
   </processInstanceState>
```

```
   <processReturnCode>
     <name>SUCCESS</name>
   </processReturnCode>
   <status>The Process Instance has successfully
completed.</status>
 </ProcessInstance>
```

Execute a process instance

Details

URI	`/processinstances/<process_instance_id>/execute`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Response Body	ProcessInstance VO
Returns	200 - Ok 401 - Not authorized 404 - Not found 500 - Server exception

The following was sent to retrieve a process instance:

```
https://<host>:<port>/csa/rest/processinstances/90d9652b362d4ecd0
1362d4fb7be0f71/execute/?userIdentifer=90d9652b35f35a930135f35b32
7e00a0
```

Chapter 13: Resource Provider API

Description

Use this API to retrieve a list of resource providers that matches the access point URL with the access point URL specified when calling this API.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/resourceProvider`
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	`accessPointUrl`
	Required; the URL that refers to the access point URL of the resource provider to be searched.
	Note: The `accesspointUrl` parameter is case insensitive.

Returns	200 - Updated 400 - Bad request 404 - Not found 401 - Not authorized

Example

The following URL was sent to import the contents of the specified archive.

```
https://<host>:<port>/csa/rest/resourceProvider?userIdentifier=8a
8185f448fbf5840148fcddd03d000d&accessPointUrl=http://admin:8082/
```

The following XML was returned:

```
<ResourceProviderList>
<count>0</count>
<limit>0</limit>
<resourceProvider>
<id>8a8185f4497d011b01497d028a210003</id>
<createdOn>22014-11-04T14:52:38.817-08:00</createdOn>
<updatedOn>2014-11-04T14:52:38.817-08:00</updatedOn>
<isCriticalSystemObject>false</isCriticalSystemObject>
<iconUrl>
/csa/images/categories/provider_type/vmware_vcenter.png
</iconUrl>
<name>test</name>
<displayName>test</displayName>
<property>
<id>8a8185f4497d011b01497d028a9d0004</id>
<createdOn>2014-11-04T14:52:38.943-08:00</createdOn>
<updatedOn>2014-11-04T14:52:44.673-08:00</updatedOn>
<createdBy>
<id>90d96588360da0c701360da0f1d600a1</id>
<isCriticalSystemObject>false</isCriticalSystemObject>
<disabled>false</disabled>
<distinguishedName>csa://Organizations/CSA-
Provider/users/admin</distinguishedName>
<userName>admin</userName>
<organization>
<id>90d96588360da0c701360da0f15b009e</id>
<isCriticalSystemObject>false</isCriticalSystemObject>
<disabled>false</disabled>
</organization>
</createdBy>
<updatedBy>
<id>90d96588360da0c701360da0f1d600a1</id>
```

```
<isCriticalSystemObject>false</isCriticalSystemObject>
<disabled>false</disabled>
<distinguishedName>csa://Organizations/CSA-
Provider/users/admin</distinguishedName>
<userName>admin</userName>
<organization>
<id>90d96588360da0c701360da0f15b009e</id>
<isCriticalSystemObject>false</isCriticalSystemObject>
<disabled>false</disabled>
</organization>
</updatedBy>
<isCriticalSystemObject>false</isCriticalSystemObject>
<description>
 This property has been automatically created, and must be set to a
datacenter name
</description>
<iconUrl>/csa/images/categories/value_type/string.png</iconUrl>
<name>DATACENTERNAME</name>
<displayName>DATACENTERNAME</displayName>
<valueType>
<id>90d96588360da0c701360da0efb50054</id>
<createdOn>2014-11-04T14:35:47.523-08:00</createdOn>
<isCriticalSystemObject>true</isCriticalSystemObject>
<description>String</description>
<iconUrl>/csa/images/categories/value_type/string.png</iconUrl>
<name>STRING</name>
<displayName>String</displayName>
<disabled>false</disabled>
<categoryType>
<id>90d96588360da0c701360da0efa0004e</id>
<createdOn>2014-11-04T14:35:47.270-08:00</createdOn>
<isCriticalSystemObject>true</isCriticalSystemObject>
<description>Value Type</description>
<name>VALUE_TYPE</name>
<displayName>Value Type</displayName>
<extensible>false</extensible>
</categoryType>
</valueType>
<values>
<id>8a8185f4497d011b01497d02a1010006</id>
<createdOn>2014-11-04T14:52:44.673-08:00</createdOn>
```

```
<updatedOn>2014-11-04T14:52:44.673-08:00</updatedOn>
<value>CSA</value>
<maxOccurs>1</maxOccurs>
<minOccurs>0</minOccurs>
<orderIndex>-1</orderIndex>
<confidential>false</confidential>
<encrypted>false</encrypted>
<consumerReadOnly>false</consumerReadOnly>
<consumerVisible>true</consumerVisible>
<measurable>false</measurable>
</property>
<state>
<id>90d96588360da0c701360da0ef470038</id>
<isCriticalSystemObject>false</isCriticalSystemObject>
<name>ACTIVE</name>
<disabled>false</disabled>
</state>
<artifactType>
<id>90d96588360da0c701360da0eed8001f</id>
<isCriticalSystemObject>false</isCriticalSystemObject>
<name>RESOURCE_PROVIDER</name>
<disabled>false</disabled>
</artifactType>
<disabled>false</disabled<providerType>
<id>90d96588360da0c701360da0ee93000f</id>
<isCriticalSystemObject>true</isCriticalSystemObject>
<description>VMware vCenter</description>
<iconUrl>
/csa/images/categories/provider_type/vmware_vcenter.png
</iconUrl>
<name>VMWARE_VCENTER</name>
<displayName>VMware vCenter</displayName>
<disabled>false</disabled>
</providerType>
</resourceProvider>
<ResourceProviderList>
```

Chapter 14: Search API

Details

URI	`/search`
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`query=<string>`
	Required; returns search results that contain *string* in their display name or description.
	
	`type=[all|subscription|service_offering|service_request|approval_process]`
	Optional; default is *all*. The type of objects to search.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Chapter 15: User API

Description

Use this API to get information related to CSA users.

Base URL

https://<host>:<port>/csa/rest

URIs

The following URIs are appended to the base URL:

Request

A request is created whenever a user initiates, changes, or deletes a subscription.

URI	Method	Parameters	Description
/user/instance/<instance_id>/request	GET	userIdentifier	"List service requests for subscription" on page 171
/user/myrequest	GET	userIdentifier, scope, detail, submitter, returnRetired, submitStartDate, submitEndDate	"List active requests for user" on page 172
/user/request/count	GET	userIdentifier	"Get count of requests for user" on page 175

URI	Method	Parameters	Description
/user/multipleRequest/cancel	POST	userIdentifier	"Cancel multiple service requests" on page 175
/user/multipleRequest/delete	POST	userIdentifier	"Delete multiple service requests" on page 177

Approval

An approval is created when the approval manager approves a request.

URI	Method	Parameters	Description
/user/approval/count	GET	userIdentifier	"Get count of approvals for user" on page 181
/user/multipleApprovals/delete	POST	userIdentifier	"Delete multiple approval requests" on page 181
/user/myapproval	GET	userIdentifier, scope, detail, returnRetired, approver, creationStartDate, CreationEndDate	List approvals for approver

Subscription

A subscription is created when a consumer requests a service offering and includes all of the options selected by the consumer when the subscription was initiated.

URI	Method	Parameters	Description
/user/mysubscription	GET	userIdentifier, scope, detail, requestor, returnRetired, creationStartDate, creationEndDate, modificationStartDate, modificationEndDate	"List subscriptions for user" on page 183
/user/subscription/count	GET	userIdentifier	"Get count of subscriptions for user" on page 186
/user/subscription	GET	userIdentifier, queryType	"Get list of recent or expiring soon subscriptions for user" on page 186
/user/multipleSubscription/delete	POST	userIdentifier	"Delete multiple subscriptions" on page 195

Instance

An instance is created when a request is approved and includes details about the requested services such as the status of services, IP addresses, etc.

URI	Method	Parameters	Description
/user/myinstance	GET	userIdentifier, scope, detail, requestor, returnRetired, creationStartDate, creationEndDate	"List instances for user" on page 197

List service requests for subscription

Details

URI	`/user/instance/<instance_id>/request` Retuns the list of service requests for the specified subscription. A request's details will include information on any failed action instances.
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Method	GET
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/instance/90cec3a03a94d689013a
a91e1b9b1218/request
?userIdentifier=BFA0DB53DA414B90E04059106D1A24B5
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceRequestList>
  <count>3</count>
  <limit>0</limit>
  <ServiceRequest>
    <id>90cec3a03a667c69013a9331365f3a54</id>Â Â Â
    <createdOn>2012-10-11T22:03:27.361-07:00</createdOn>
    <updatedOn>2012-10-11T22:04:20.553-07:00</updatedOn>
    <createdBy> ... </createdBy>
```

```
    <updatedBy> ... </updatedBy>
    <isCriticalSystemObject>false</isCriticalSystemObject>
    <description>OctoberSampleService</description>
    <iconUrl>/csa/images/library/Icon034_48.png</iconUrl>
    <name>Wednesday Demo</name>
    <displayName>Wednesday Demo</displayName>
    <disabled>false</disabled>
    <requestState> ... </requestState>
    <requestedAction> ... </requestedAction>
    <actionInstance>
      <id>90cec3a03a667c69013a9331a2800070</id>
      <createdOn>2012-10-11T22:04:05.248-07:00</createdOn>
      <updatedOn>2012-10-11T22:04:15.152-07:00</createdOn>
      <action> ... </action>
      <processInstance> ... </processInstance>
      <actionInstanceState> ... </actionInstanceState>
      <context>90cec3a03a667c69013a9331365f3a54</context>

<callbackBean>requestPorcessorActionCallbackHandler</callbackBea
n>
      <callbackPending>false</callbackPending>
      <status> ... </status>
      <actionInstanceReturnCode> ... </actionInstanceReturnCode>
      <archive>false</archive></actionInstance>
  </ServiceRequest>
  ...
</ServiceRequestList>
```

List active requests for user

Details

URI	/user/myrequest
Method	GET

URI	`/user/myrequest`

Parameters	`userIdentifier=<user_id>`
	Required; this user must be in the same organization as submitter, and must have the necessary permissions for the data you want to work with.See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|view]`
	Optional; default is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`sbmitter=<user_name>`
	Required; user name must be valid, and is the user whose request list will be returned by this call. *submitter* must be in the same organization as the user specified by *userIdentifier*.
	
	`returnRetired=[true|false]`
	Optional; default is *false*.
	
	`submitStartDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; requests submitted **on or after** start date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.
	
	`submitEndDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; requests submitted **before** end date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/myrequest?userIdentifier=8a81
81853810699a01381076be5400a0 &submitter=acctgconsumer
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceRequestList>
    <count>21</count>
    <limit>0</limit>
    <ServiceRequest>
        <id>8a8181853810699a01381079190800a7</id>
        <objectId>8a8181853810699a01381079190800a7</objectId>
        <createdOn>2012-06-21T12:16:08.073-07:00</createdOn>
        <updatedOn>2012-06-21T12:16:50.787-07:00</updatedOn>
        <isCriticalSystemObject>false</isCriticalSystemObject>
        <description>SD2 Offering</description>
        <detailedDescription>desc - SD2
offering</detailedDescription>
        <iconUrl>/csa/images/library/application.png</iconUrl>
        <name>request 1</name>
        <displayName>request 1</displayName>
        <state>
            <id>90d96588360da0c701360da0ef470038</id>
            <objectId>90d96588360da0c701360da0ef470038</objectId>
            <createdOn>2012-06-21T11:51:43.267-07:00</createdOn>
            <isCriticalSystemObject>true</isCriticalSystemObject>
            <description>Active</description>
            <iconUrl>/csa/images/categories/artifact_
state/active.png</iconUrl>
            <name>ACTIVE</name>
            <displayName>Active</displayName>
            <disabled>false</disabled>
            <categoryType>
                <id>90d96588360da0c701360da0ef420037</id>

<objectId>90d96588360da0c701360da0ef420037</objectId>

<isCriticalSystemObject>true</isCriticalSystemObject>
                <name>ARTIFACT_STATE</name>
                <displayName>Artifact State</displayName>
                <extensible>false</extensible>
```

```
            </categoryType>
          </state>
          ...
      </ServiceRequest>
</ServiceRequestList>
```

Get count of requests for user

Details

URI	`/user/request/count` Gets the number of requests for the user associated with userIdentifier. The results are grouped by request state.
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Cancel multiple service requests

Details

URI	`/user/multipleRequest/cancel`
Method	POST

URI	`/user/multipleRequest/cancel`
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for whether all specified requests were canceled.) 401 - Not authorized 404 - Object not found 500 - Server exception

A request can only be cancelled if its state is PENDING.
Note:

- When a pending subscription modification request is canceled, only the modification is canceled; the subscription is not changed.
- Once a subscription request has been approved, the request can no longer be canceled. The subscription, though, can be canceled.

Example

Use the following URL:

```
https://<host>:<port>/csa/rest/user/multipleRequest/cancel
?userIdentifier=90d965c0379fd06601379fd192b30ee6
```

The following XML was sent in the request:

```
<ServiceRequestList>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2c015103a4</id>
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceRequest>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
```

```
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceRequest>
</ServiceRequestList>
```

The following XML was returned in the response. The count value indicates the number of service requests successfully canceled. A ServiceRequest element is included for each canceled service request

```
<ServiceRequestList>
    <count>2</count>
    <limit>0</limit>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2c015103a4</id>
  ...
    </ServiceRequest>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
  ...
    </ServiceRequest>
</ServiceRequestList>
```

Delete multiple service requests

Details

URI	`/user/multipleRequest/delete`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.

URI	/user/multipleRequest/delete
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for whether all specified requests were deleted.) 401 - Not authorized 404 - Object not found 500 - Server exception

A request can only be deleted if its state is APPROVED, DENIED, or CANCELED.

Example

Use the following URL:

```
https://<host>:<port>/csa/rest/user/multipleRequest/delete
?userIdentifier=90d965c0379fd06601379fd192b30ee6
```

The following XML was sent in the request:

```
<ServiceRequestList>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2c015103a4</id>
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceRequest>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceRequest>
</ServiceRequestList>
```

The following XML was returned in the response. The count value indicates the number of service requests successfully deleted. A ServiceRequest element is included for each deleted service request.

```
<ServiceRequestList>
    <count>2</count>
    <limit>0</limit>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2c015103a4</id>
        ...
    </ServiceRequest>
    <ServiceRequest>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
        ...
    </ServiceRequest>
/ServiceRequestList>
```

List approvals for approver

Details

URI	`/user/myapproval`
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with.See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|view]`
	Optional; default is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`returnRetired=[true|false]`
	Optional; default is *false*.
	
	`approver=<user_name>`
	Required; user name must be valid, and is the user whose approval list will be returned by this call. *approver* must be in the same organization as the user specified by *userIdentifier*.
	
	`creationStartDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; approval requests created on or after start date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.
	
	`creationEndDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; approval requests created before end date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.
	
	Caution: The users specified by `userIdentifier` and `approver` must be in the same organization.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Get count of approvals for user

Details

URI	`/user/approval/count` Gets the number of approvals for the user associated with userIdentifier. The results are grouped by approval result state.
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Delete multiple approval requests

Details

URI	`/user/multipleApprovals/delete`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.

URI	/user/multipleApprovals/delete
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for whether all specified approval requests were deleted.) 401 - Not authorized 404 - Object not found 500 - Server exception

An approval request can only be deleted if its state is APPROVED or DENIED.

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/multipleApprovals/delete
?userIdentifier=90d965c0379fd06601379fd192b30ee6
```

The following XML was sent in the request:

```
<ApprovalProcessList>
    <approvalProcess>
        <id>90e72e713a94e0ab013aae76618e0e39</id>
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </approvalProcess>
    <approvalProcess>
        <id>90e2a4133a75430b013a7a1328560377</id>
        <catalogItem>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </approvalProcess>
</ApprovalProcessList>
```

The following XML was returned in the response. The count value indicates the number of approval requests successfully deleted. An ApprovalProcess element is included for each deleted approval request.

```
<ApprovalProcessList>
    <count>2</count>
    <limit>0</limit>
    <ApprovalProcess>
        <id>90e72e713a94e0ab013aae76618e0e39</id>
        ...
    </ApprovalProcess>
    <ApprovalProcess>
        <id>90e2a4133a75430b013a7a1328560377</id>
        ...
    </ApprovalProcess>
</ServiceSubscriptionList>
```

List subscriptions for user

Details

URI	`/user/mysubscription`
Method	GET

URI	/user/mysubscription

Parameters	`userIdentifier=<user_id>`
	Required; this user must be in the same organization as requestor, and must have the necessary permissions for the data you want to work with.See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=[base|view]`
	Optional; default is *base*.
	
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`requestor=<user_name>`
	Required; user name must be valid, and is the user whose subscription list will be returned by this call. *requestor* must be in the same organization as the user specified by *userIdentifier*.
	
	`returnRetired=[true|false]`
	Optional; default is *false*.
	
	`creationStartDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions created **on or after** start date and time will be included in the return.
	
	`creationEndDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions created **before** end date and time will be included in the return. Note that creationStartDate must also be specified.
	
	`modificationStartDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions updated **on or after** specified start date and time will be included in the return.
	
	`modificationEndDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions updated **before** specified end date and time will be included in the return. Note that modificationStartDate must also be specified.
	
	Note that in all date parameters:
	
	• `T` serves as a separator between date and time.

URI	`/user/mysubscription`
	• Time will default to 00:00:00 if not specified. • Date and time are assumed to be in the time zone of the CSA server.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/mysubscription?userIdentifier
=90d9652b67ss6a930135f35b327e00a0 &requestor=RnDUser
```

The following XML was returned:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceSubscriptionList>
    <count>6</count>
    <limit>0</limit>
    <ServiceSubscription>
        <id>90d957ea3806fa7e013807acc79000b3</id>
        <iconUrl>
            /csa/images/library/serviceOfferingDefault58.png
        </iconUrl>
        <name>MY SR</name>
        <displayName>MY SR</displayName>
        <state>...</state>
        <artifactType>
            <name>SUBSCRIPTION</name>
            ...
        </artifactType>
        <disabled>false</disabled>
        <serviceOffering>...</serviceOffering>
        <subscriptionStatus>...</subscriptionStatus>
        <initiatingServiceRequest>...</initiatingServiceRequest>
        ...
```

```
    </ServiceSubscription>
     ...
</ServiceSubscriptionList>
```

Get count of subscriptions for user

Details

URI	`/user/subscription/count` Gets the number of subscriptions for the user associated with userIdentifier. The resutls are grouped by subscription status.
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Get list of recent or expiring soon subscriptions for user

Details

URI	`/user/subscription` Returns a list of subscriptions for the user associated with userIdentifier.
Method	GET

URI	`/user/subscription` Returns a list of subscriptions for the user associated with userIdentifier.

Parameters	`userIdentifier=<user_id>`
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user who has the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`queryType=[expiringSoon|recent]`
	Optional; *expiringSoon* returns the user's subscriptions that will expire within the next 30 days, and *recent* returns the last five subscriptions initiated by the user.

Returns	200 - Ok 401 - Not authorized 500 - Server exception

Get all components of a specific type for a specific user

Details

URI	/user/mycomponents?userIdentifier=<userId>&componentType=<*comma-separated values of a componentType*>&componentTypeWildcard=<*comma-separated values of type that are treated with wild card*>&creationDateBegin=<creationDateBegin>&creationDateEnd=<creationDateEnd>&updatedDateBegin=<updatedDateBegin>&updatedDateEnd=<updatedDateEnd>&returnActiveOnly=<true|false>&GMT=<true/false>
Method	GET

URI	`/user/mycomponents?userIdentifier=<userId>&componentType=<comma-separated values of a componentType>&componentTypeWildcard=<comma-separated values of type that are treated with wild card>&creationDateBegin=<creationDateBegin>&creationDateEnd=<creationDateEnd>&updatedDateBegin=<updatedDateBegin>&updatedDateEnd=<updatedDateEnd>&returnActiveOnly=<true|false>&GMT=<true/false>`

Parameters	`userIdentifier=<user_id>`
	
	Required; the user ID you want to use as credentials for this API call. This user should be a consumer user with the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121" for the steps required to retrieve the *userIdentifier* value.
	
	`componentType=<comma-separated values>`
	
	Optional; Specify the component name to retrieve. This parameter takes comma-separated values such as SERVER,INFRASTRUCTURE.
	
	`componentTypeWildCard=<comma-separated values>`
	
	Optional; use this parameter to specify the component name to retrieve.
	
	This parameter takes comma-separated values. The names are appended with wildcard characters and should not be specified. For example, the value Server retrieves all components of type SERVER, SERVER_GROUP, or Server.
	
	Using this parameter may affect the performance of the API and HP recommends using the `componentType` parameter whenever possible.
	
	`creationDateBegin: <yyyy-MM-ddTHH:mm:ss>`
	
	Optional; subscriptions created on or after the start date and time are included in the return.
	
	`creationDateEnd: <yyyy-MM-ddTHH:mm:ss>`
	
	Optional; subscriptions created before the end date and time are included in the return. Note that the `creationStartDate` parameter

URI	`/user/mycomponents?userIdentifier=<userId>&componentType=<comma-separated values of a componentType>&componentTypeWildcard=<comma-separated values of type that are treated with wild card>&creationDateBegin=<creationDateBegin>&creationDateEnd=<creationDateEnd>&updatedDateBegin=<updatedDateBegin>&updatedDateEnd=<updatedDateEnd>&returnActiveOnly=<true|false>&GMT=<true/false>`

	must also be specified.
	`updatedDateBegin: <yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions updated on or after the specified start date and time are included in the return.
	`updatedDateEnd: <yyyy-MM-ddTHH:mm:ss>`
	Optional; subscriptions updated before the specified end date and time are included in the return. Note that the `modificationStartDate` parameter must also be specified.
	`returnActiveOnly: true/false`
	Optional; specifying true for this parameter returns only subscriptions that are not in retired state.
	`GMT : true/false`
	Optional; specifying true for this parameter configures HP CSA to interpret date-related parameters in GMT. This is important if the application communicating to HP CSA is not in the same time zone.
	Note: In all date parameters, `T` serves as a separator between date and time and time defaults to 00:00:00:00 if not specified.

Returns	200 - OK 401 - Not authorized 500 - Server exception

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/mycomponents?userIdentifier=90s96588670da0c701360da0f1d540a1&componentType=Server
```

The following XML was returned:

```
<UserSubscriptionComponentsList>
 <ServiceSubscription>

  <serviceInstanceId>9038afc8476159bc0147626081c01291</serviceInstanceId>
  <serviceInstanceState>ACTIVE</serviceInstanceState>
  <serviceOfferingName>TestOffering2</serviceOfferingName>
  <subscriberUserOrg>RND</subscriberUserOrg>

  <subscriptionId>9038afc8476159bc014762606e621232</subscriptionId>
  <subscriptionState>ACTIVE</subscriptionState>
  <subscriptionStatus>ACTIVE</subscriptionStatus>
  <subsriptionDisplayName>
Subscription 0 for 9038afc8465b3f9801465b56ad5e001b
</subsriptionDisplayName>
 <ServiceComponent>
  <componentId>9038afc8476159bc0147626081c51316</componentId>
  <componentLifecycleState>DEPLOYED</componentLifecycleState>
  <componentType>SERVER</componentType>
  <isVisible>true</isVisible>
  <Property>
   <confidential>false</confidential>
   <consumerReadOnly>false</consumerReadOnly>
   <consumerVisible>false</consumerVisible>
   <encrypted>false</encrypted>
   <propertyName>memory</propertyName>
   <valueType>Integer</valueType>
   <values>
    <value>1024</value>
   </values>
```

```
</Property>
<Property>
 <confidential>false</confidential>
 <consumerReadOnly>false</consumerReadOnly>
 <consumerVisible>false</consumerVisible>
 <encrypted>false</encrypted>
 <propertyName>datacenterName</propertyName>
 <valueType>String</valueType>
 <values />
</Property>
<Property>
 <confidential>false</confidential>
 <consumerReadOnly>false</consumerReadOnly>
 <consumerVisible>false</consumerVisible>
 <encrypted>false</encrypted>
 <propertyName>customSpec</propertyName>
 <valueType>String</valueType>
 <values />
</Property>
<Property>
 <confidential>false</confidential>
 <consumerReadOnly>false</consumerReadOnly>
 <consumerVisible>false</consumerVisible>
 <encrypted>false</encrypted>
 <propertyName>nCPU</propertyName>
 <valueType>Integer</valueType>
 <values>
 <value>2</value>
 </values>
</Property>
<Property>
 <confidential>false</confidential>
 <consumerReadOnly>false</consumerReadOnly>
 <consumerVisible>false</consumerVisible>
 <encrypted>false</encrypted>
 <propertyName>HostName</propertyName>
 <valueType>String</valueType>
 <values>
 <value>CentOS6.4_x64-Less-3601</value>
```

```
    </values>
   </Property>
   <Property>
    <confidential>false</confidential>
    <consumerReadOnly>false</consumerReadOnly>
    <consumerVisible>false</consumerVisible>
    <encrypted>false</encrypted>
    <propertyName>HostName</propertyName>
    <valueType>String</valueType>
    <values>
    <value>CentOS6.4_x64-Less-3601</value>
    </values>
   </Property>
   <Property>
    <confidential>false</confidential>
    <consumerReadOnly>false</consumerReadOnly>
    <consumerVisible>false</consumerVisible>
    <encrypted>false</encrypted>
    <propertyName>osType</propertyName>
    <valueType>String</valueType>
    <values>
    <value>Linux</value>
    </values>
   </Property>
  </ServiceComponent>
  <ServiceComponent>
   <componentId>9038afc8476159bc0147626081c51317</componentId>
   <componentLifecycleState>DEPLOYED</componentLifecycleState>
   <componentType>INFRASTRUCTURE_SERVICE</componentType>
   <isVisible>true</isVisible>
   <Property>
    <confidential>false</confidential>
    <consumerReadOnly>false</consumerReadOnly>
    <consumerVisible>false</consumerVisible>
    <encrypted>false</encrypted>
    <propertyName>templateReference</propertyName>
    <valueType>String</valueType>
    </Property>
   </ServiceComponent>
```

```
<ServiceComponent>
 <componentId>9038afc8476159bc0147626081c51318</componentId>
 <componentLifecycleState>DEPLOYED</componentLifecycleState>
 <componentType>SERVICE_COMPOSITE</componentType>
 <isVisible>true</isVisible>
</ServiceComponent>
<ServiceComponent>
 <componentId>9038afc8476159bc0147626081c51319</componentId>
 <componentLifecycleState>DEPLOYED</componentLifecycleState>
 <componentType>SERVER_GROUP</componentType>
 <isVisible>true</isVisible>
 <Property>
  <confidential>false</confidential>
  <consumerReadOnly>false</consumerReadOnly>
  <consumerVisible>false</consumerVisible>
  <encrypted>false</encrypted>
  <propertyName>serverCount</propertyName>
  <valueType>Integer</valueType>
  <values>
   <value>1</value>
  </values>
 </Property>
 <Property>
  <confidential>false</confidential>
  <consumerReadOnly>false</consumerReadOnly>
  <consumerVisible>false</consumerVisible>
  <encrypted>false</encrypted>
  <propertyName>VCENTER_PROVIDER</propertyName>
  <valueType>String</valueType>
  <values>
   <value>90cd93d9439a378501439b1d000b1b02</value>
  </values>
 </Property>
 <Property>
  <confidential>false</confidential>
  <consumerReadOnly>false</consumerReadOnly>
  <consumerVisible>false</consumerVisible>
  <encrypted>false</encrypted>
  <propertyName>HostName</propertyName>
```

```
    <valueType>String</valueType>
    <values />
   </Property>
  </ServiceComponent>
 </ServiceSubscription>
<ServiceSubscription>
       :
       :
</ ServiceSubscription>
</UserSubscriptionComponentsList>
```

Delete multiple subscriptions

Details

URI	`/user/multipleSubscription/delete`
Method	POST
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Ok (Indicates the REST call executed without error. See XML return content for whether all specified subscriptions were deleteed.) 401 - Not authorized 404 - Object not found 500 - Server exception

Example

Use the following URL:

```
https://<host>:<port>/csa/rest/user/multipleSubscription/delete
?userIdentifier=90d965c0379fd06601379fd192b30ee6 Â
```

The following XML was sent in the request:

```
<ServiceSubscriptionList>
    <ServiceSubscription>
        <id>90e72e283b05aff1013b0b2c015103a4</id>
        <catalogItem>
               <id>90cef5de3c63429f013c6489245b09a2</id>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceSubscription>
    <ServiceSubscription>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
        <catalogItem>
      <id>90cef5de3c63429f013c648924bf09b4</id>
            <catalog>
                <id>90d965c0379fd06601379fd1936a0f05</id>
            </catalog>
        </catalogItem>
    </ServiceSubscription>
</ServiceSubscriptionList>
```

The following XML was returned. count indicates the number of subscriptions successfully deleted. A ServiceSubscription element is included for each deleted subscription.

To make this example more informative, note in the following return information that the first subscription is not deleted. A subscription will not be deleted during this call if the subscription was already deleted, it is not in the canceled state, permission is denied, or there is an authorization failure

```
<ServiceSubscriptionList>
    <count>1</count>
    <limit>0</limit>
    <ServiceSubscription>
        <id>90e72e283b05aff1013b0b2b43fc0356</id>
  <isCriticalSystemObject>false</isCriticalSystemObject>
  <disabled>false</disabled>
</ServiceSubscription>
</ServiceSubscriptionList>
```

List instances for user

Details

URI	`/user/myinstance`
Method	GET

Parameters	`userIdentifier=<user_id>`
	Required; this user must be in the same organization as requestor, and must have the necessary permissions for the data you want to work with. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
	
	`scope=base`
	Optional; the only valid value is *base*.
	`detail=basic`
	Optional; The only valid value is *basic*.
	
	`requestor=<user_name>`
	Required; user name must be valid, and is the user whose subscription list will be returned by this call. *requestor* must be in the same organization as the user specified by *userIdentifier*.
	
	`creationStartDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; instances for subscriptions created on or after start date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.
	
	`creationEndDate=<yyyy-MM-ddTHH:mm:ss>`
	Optional; instances for subscriptions created before end date and time will be included in the return. *T* serves as a separator between data and time. Time will default to 00:00:00 if not specified. Date and time are assumed to be in the time zone of the CSA server.

URI	/user/myinstance
Returns	200 - Ok 401 - Not authorized 500 - Server exception

Examples

The following URL was sent:

```
https://<host>:<port>/csa/rest/user/myinstance?userIdentifier=90d
9652b67ss6a930135f35b327e00a0 &requestor=johnsmith
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceInstanceList>
     <count>6</count>
     <limit>0</limit>
     <ServiceInstance>
        <id>90d957ea3806fa7e01380f957d11070a</id>
        <name>MYSD_June 5, 2012 5:19:51 PM UTC</name>
        <displayName>MYSD</displayName>
        <state></state>
        <serviceInstanceState>...</serviceInstanceState>
        ...
     </ServiceInstance>
     ...
<ServiceInstanceList>
```

Chapter 16: Utilization API

Description

Use this API to retrieve a list of resource utilization objects for the specified subscription.

Base URL

https://<host>:<port>/csa/rest

Details

URI	`/utilization/<subscription_id>` See List subscriptions for user for information on obtaining subscription IDs.
Method	GET
Parameters	`userIdentifier=<user_id>` Required; the user ID you want to use as credentials for this API call. See "Get userIdentifier" on page 121 for the steps required to get the userIdentifier value.
Returns	200 - Updated 404 - Not found 500 - Server exception

Example use context

An organization's policy is to do resource provider selection based on available capacity for all providers. The information returned by this API can be used to determine what resources are already in use.

Example

The following URL was sent:

```
https://<host>:<port>/csa/rest/utilization/90cec3a03a667c69013a6d
7f0eea2cb3
```

The following XML was returned in the response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<UtilizationList
  <utilization>

<resourceBindingId>90e72d913d936a9a013d937bdb570161</resourceBind
ingId>
    <resourcePool>
      <id>90e72d913d936a9a013d936c74040006</id>
      <objectId>90e72d913d936a9a013d936c74040006</objectId>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>Pool_1_March 22, 2013 6:46:31 PM UTC</name>
      <displayName>Pool 1</displayName>
      <disabled>false</disabled>
      <poolReference>poolref</poolReference>
    </resourcePool>
    <resourceProvider>
      <id>90e72d913d936a9a013d936c262b0004</id>
      <objectId>90e72d913d936a9a013d936c262b0004</objectId>
      <isCriticalSystemObject>false</isCriticalSystemObject>
      <name>Provider_1_March 22, 2013 6:46:11 PM UTC</name>
      <displayName>Provider 1</displayName>
      <disabled>false</disabled>
    </resourceProvider>
    <resourceType>
      <id>6A883A543D1248DDBB6045AD6A06BEA2</id>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Megabytes of memory</description>
      <name>MEMORY</name>
      <displayName>Memory</displayName>
      <disabled>false</disabled>
    </resourceType>
```

```
<serviceComponentId>90e72d913d936a9a013d937bdb56015a</serviceComp
onentId>

<serviceInstanceId>90e72d913d936a9a013d937bdb530138</serviceInsta
nceId>
    <unit>
      <id>ED2CE3264F764F55877EE0CDFEE0EFB4</id>
      <isCriticalSystemObject>true</isCriticalSystemObject>
      <description>Megabytes</description>
      <name>MB</name>
      <displayName>MB</displayName>
      <disabled>false</disabled
    </unit>
    <usage>1024</usage>
  </utilization>
  <utilization> … </utilization>
  <utilization> … </utilization>
  …
</UtilizationList>
```

Chapter 17: Values for the detail parameter

The detail parameter has the following values:

- **Required**: Retrieves all the non-null and non-optional fields of an artifact. This includes fields with Java primitive types – byte, short, int, long, float, double, boolean and char.
- **Basic**: In addition to the Required fields of an artifact, retrieves nullable primitive types (Java wrapper classes) and Date, Time, TimeStamp and BigDecimal.
- **Standard**: In addition to the Basic fields of an artifact, retrieves all non-CSA type fields.
- **Template**: Retrieves all fields of an artifact except for artifact collection references and artifacts that are part of a containment relationship. This is same as the Full option without artifact collection references. This option is mainly used for performance reasons, because you could end up retrieving a large amount of data. For example, you can use this option if you do not want to load the base service instances when retrieving its service blueprint. Service instances are artifacts and are associated through a collection reference with its blueprint.
- **Full**: All fields except for artifacts that are part of a containment relationship. Retrieval of the internal artifacts is controlled by the scope parameter. Using this option has a performance impact (See Template option) because this option also returns artifact collection references.

Note: Some API calls do not support all possible values for this parameter.

Chapter 18: Values for the scope parameter

The scope parameter has the following values:

- **Base**: Retrieves the root entity of the artifact including all required attributes.
- **Baseplusone**: In addition to Base, retrieves all first level child artifacts.
- **Subtree**: In addition to Base, retrieves all its descendants. Using this option could have a performance impact because the information being retrieved can be very large.
- **View**: Retrieves only those attributes defined for the view specified in the *detail* parameter.

Chapter 18: Primitive values reset to default if no value is provided

When you make a PUT request from the REST API, if you do not provide values for primitive properties, these properties are set to their default values.

Some properties of an artifact are modeled in Java using primitive types. This can cause certain issues when using the PUT requests via REST API. When a PUT request is sent to the HP CSA instance using a REST API, the data that represents an artifact or a part of an artifact is converted to a java object. All properties with primitive types in a Java object always need to have values. When you do not provide values for these properties, the default values are used to satisfy the requirement.

The following are the primitive types in Java:

- byte
- short
- int
- long

- float
- double
- char
- boolean

Workaround: Invoke a GET call first, and then modify only the necessary properties from the GET response. This modified response should be sent as part of the PUT request.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on API Reference (Cloud Service Automation 4.20)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to CSAdocs@hp.com.

We appreciate your feedback!


Java™
COMPATIBLE