Citrix® CloudPortal™ Services Manager 11.0 API User's Guide

Document Version 1.1

Version	Release Date	Edits
1.0	Aug. 10, 2012	Initial 10.0 Release
1.1	June 10, 2013	11.0 Release

Contents

Chapter 1

Introduction

Topics:

- ***Why use the API?***
- ***Who should use the API?***
- ***How it works.***
- ***CloudPortal Services Manager Entities – Customers, Users and Services***

The application programming interface (API) is a powerful interface that allows you to interact directly with the CloudPortal Services Manager (CPSM) without using the CPSM Web User Interface (UI).

The API grants a user, with some development knowledge, the ability to easily manage all core objects within CPSM (Customers, Users, Services) from virtually any programming language.

CPSM uses a Web based API service, which sends and receives Extensible Mark-up Language (XML) formatted requests.

Why use the API?

Using the API, you can automate tasks or allow other external software products and solutions, which are not controlled by CPSM, to interact with CPSM directly.

For example, you can interface CPSM with a billing application to generate billing every month, further automating an otherwise manual process.

The API is also secure, as only a user that has the rights to carry out a task in the CSPM UI will have the same rights in the API. Granting a customer access to the API is safe, as they can only alter the details of the customer they belong to.

The API is useful in migration tasks. Correctly formatted data is used to automatically provision users and services. The API can also be used to automate complex bulk tasks – for example, adding a particular service to a large volume of users.

Who should use the API?

Anyone with a basic understanding of scripting or programming should have sufficient knowledge to leverage the power of the API.

The API test tool allows you to construct XML requests and run them without having to create a full C Sharp (C#) or Visual Basic (VB) application.

How it works

XML requests are sent to the CPSM API web service which interprets the request and either returns information or performs the action requested.

The API web service passes the request onto the provisioning engine and then the request is carried out in a similar way as it would if the request had been performed manually through the CPSM UI.

CPSM Entities – Customers, Users and Services

The CPSM API provides programmatic access to the same entities that are managed by the CPSM user interface:

- Customers are the main entities within CPSM.
- Each customer contains users.
- Both customers and users have services.
- Each of these entities has properties that can be changed using the API.

Provisioning

CPSM uses a database to store customer, user, and service configuration information. A process called provisioning is used to push these configurations out to the systems and servers that provide the service. When an API call is made, the CPSM database is updated immediately. However, changes are not propagated out to the servers until an entity is provisioned. Provisioning kicks off what can sometimes be a lengthy process of updating systems and servers. This happens as a background process and does not delay the response you will receive from an API request.

Customers, customer services, users, and user services can be provisioned.

Each entity has a provisioning status that reflects the current state of the entity. The following table describes these states:

Status	CPSM UI Indicator (color)	CSPM API Status
Not provisioned	Grey	NotProvisioned
Provisioning requested	Yellow	Requested
Provisioning in progress	Orange	InProgress
Provisioned	Green	Provisioned
Provisioning failed	Red	Failed
Pending Changes	Blue	Pending

Services

Services can be provisioned to a customer. These are called customer services. After a customer has been provisioned with a customer service, the service can then be provisioned to a user. These are called user services.

Provisioning a customer service sets up all the customer-related properties for that service and provisioning a user service sets up all the user-related properties for that service.

For example, when provisioning the Hosted Exchange service to a customer you will need to decide whether or not to create public folders and add the domain names for which mail will be accepted. When provisioning the Hosted Exchange service to a user, you need to set the user's mail.

Customer Plans

Customer plans (formerly Packages or Package Templates) group together common customer service properties. Some customer services don't have customer plans. For those that do, you can only select one customer plan when provisioning the customer with the service.

User Plans

User plans (formerly Service Access Levels) group together common user service properties. Some user services don't have a user plan. In the context of the Hosted Exchange service, User plans are also referred to as user packages.

When provisioning a customer service, you can specify which user plans are available for provisioning the service to a user. Then, when provisioning the user service, you can select from one of the user plans enabled by the customer service.

Provisioning limits can be applied to each user plan. For example, you can allow 10 users to be created with the Gold user plan and 100 users with the Silver user plan.

Chapter 2

API Usage

Topics:

This chapter discusses how to use the API with CPSM. Many examples and explanations will be provided throughout the section.

Accessing the API

Where is the Citrix API web service located?

By default, the CPSM API web service is located on the same server as the CPSM Web platform server. If you are a reseller accessing the CPSM API web service, you need to request the CPSM API URL from your service provider.

For the service provider:

1. Connect to the CPSM Web server and launch Internet Information Services (IIS) Manager (**Start > Run > inetmgr.exe**)
2. Expand the **Sites** folder, select **Cortex Management**, and then select **CortexAPI**.
3. Browse to the **Default.aspx** page.
4. When prompted, input your CPSM username and password.

If the API is working correctly, the following response appears:

```
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0" />
```

*Note: This does not necessarily mean that your CPSM Username and Password is working correctly.

The URL for this page should be similar to the following:

https:// *CPSM-Web-Server*/CortexAPI/Default.aspx

For example, https://www.cpsmdemo.com/cortexapi/default.aspx might be the external URL that you provide to your clients so they can use the API.

SSL

Citrix strongly recommends accessing this API only over SSL.

Although the installers install the API with SSL, the API will still work, even without SSL.

Granting users access to the API

To grant a user access to the API, you will need to give the user a security role that has the API Access role permission within CPSM.

Citrix recommends creating a new security role that includes only the API Access permission at the level of access you specify. This allows you to keep the current security roles you have assigned to the user already. To create a new role, perform the following actions:

1. From the Services Manager menu bar, select **Configuration > Security > Security Roles**.
2. Click **New Role** and name the role **API Access**.
3. Under **Role Permissions**, on the **Customers** tab, select **API Access** and click to choose the level of access.
4. Click **Save**.

Afterward, you provision the security role to the customer so the customer can assign the security role to a specific user. To assign the security role to a customer, perform the following actions:

1. From the Services Manager menu bar, select **Customers** and then locate the customer to whom you want to assign the security role. Click **Edit Customer**.
2. Click **Advanced Properties** and, in **Allowed Roles**, select the API Access role you created earlier.

3. Click **Provision** to enable the security role for the customer.
4. Assign the security role to users:
    a. Log in to CPSM as the customer administrator, or go to the customer's Users list.
    b. Pick a user and click **Edit User**.
    c. Click **Account Settings** and then click **Advanced Options**.
    d. Select **Configure a custom role collection** and then select the API Access role.
    e. Click **Provision** to assign the role to the user.

Be sure to provide the user with sufficient privileges that allow them to fully carry out required tasks. If the user can only access the current customer's users, they will not be able to create any sub customers. So, granting the user the Reseller Administrator security role might be required if the API is being used by resellers.

Request Basics

Requests are sent using an HTTP Post to the CPSM API's URL. The request is formatted using XML with a content type of text/xml. An action attribute determines what type of operation to perform.

Authentication

All requests must be authenticated and using HTTP Basic authentication. The use of SSL is strongly recommended to protect the username and password.

The identity used to authenticate must be a valid CPSM administrative user. The user's security permissions will determine what the user can see and what actions the user can perform. The security role of the identity must include administrative permissions and the API Access permission.

Request Structure

The following XML defines the request structure:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="x">
       …entity…
</request>
```

All of the tags and attributes names are case sensitive. Invalid names or those with the wrong case will be ignored.

The Action attribute is mandatory and defines what operation the request performs. Valid definitions are FIND, GET, SET or DELETE. Not all actions are available on all entities. You can use a GET action on an entity which will return a response that can then be used as the basis of the SET request for that same entity.

The Version attribute is mandatory and should be set to "1.0". This is used internally to allow backward compatible changes to the API interface.

The Entity is the object on which the request is operating. Valid objects are customer, location, or template. The customer is the most common entity and it contains service and user entities.

The layout of requests follows a similar pattern for every action. Here's an example to get a customer:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
       <customer>
              <name>CS</name><!-- The customer short name -->
       </customer>
</request>
```

The **action** attribute on the **request** tag determines the action that is executed. Since we want detailed information, we set it to GET. The **name** tag in the **customer** tag expects the customer short name.

Request Actions

There are four actions that you can perform when passing in information to the API. These actions dictate how the information is handled and the returned responses.

FIND Action

The FIND action executes a search and retrieves a summarized list of matching entities. Search criteria can be specified to filter the results. If search criteria are not specified, then all entities accessible to the user are returned in the response.

Customers can be searched by using any combination of these filters:

- Name

- Fullname
- Id (CustomerID)
- BillingId
- PrimaryDomain
- Status
- Parent

Customer services can be searched for by using any combination of these filters:

- Name
- Fullname
- Location
- Status

Users can be searched for by using any combination of these filters:

- Name
- Fullname
- Id (UserID)
- UPN
- Location
- Department
- Status

User services can be searched for by using any combination of these filters:

- Name
- Fullname
- Status

GET Action

The GET action retrieves details for the entities you request.

Customers can be selected with one of these fields:

- Name
- Fullname
- Id
- BillingId

Users can be selected with one of these fields:

- Name
- Fullname
- Id
- UPN

- Properties

User services can be selected with one of these fields:

- Name
- Fullname

SET Action

The SET action either creates or updates a customer, service or user. The entity definition response from a GET action can be used as the basis for a SET action. The required fields (refer to entity reference section) must be specified if you are creating the entity, not just updating it.

Customer creation requires all of these fields:

- Fullname
- ContactName
- ContactEmail
- PrimaryDomain

The Name field will be auto-generated if not specified.

User creation requires one of these fields:

- Name
- UPN

The Name field will be auto-generated if not specified.

DELETE Action

The DELETE action deletes the entity, using the same format as the GET request. For example, if you specify a customer, then that customer is deleted. The DELETE action can also be applied to users. Only service instances can be deleted; services without instances can be de-provisioned only.

Response Status

If the request is successful the HTTP response has a status of “200 OK.” This is returned along with the relevant response.

If there is a problem processing the request, the response has an error tag with a numeric ID and a textual message explaining the reason for the error. Possible errors are:

Error code	Description
400	Invalid Request (Action specified was invalid, or the request format was invalid): Resending the request will always fail.
401	Unauthorized Access: The credentials used do not have the authorization to perform the action.
500	Request could not be processed: Resending the request may work at some time in the future.

Response Examples

There are three main response types: Find, Get, and Set / Delete.

Find Action Response

The Find action returns data according to the parameters you specify.

For example if you specify a service that does not exist, the action still returns the customer information, but not the service details.

```
<?xml version="1.0" encoding="utf-8" ?>
 <response version="1.0">
        <customer>
.
.
        </customer>
</response>
```

If the action returns the following response, it means the requested customer information is not found. A possible reason may be that the customer is not in the system.

```
  <?xml version="1.0" encoding="utf-8" ?>
  <response version="1.0" />
```

Get Action Response

The Get action returns all information it can about the entities you specify in the request.

```
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0">
       <customer>
.
.
       </customer>
</response>
```

If all the specified entities cannot not be found, then the action returns an error message stating what could not be found.

```
 <?xml version="1.0" encoding="utf-8" ?>
 <response version="1.0">
       <error>
              <id>###</id>
              <message>Customer 'BAC' not found.</message>
       </error>
 </response>
```

Set / Delete Action Responses

A successful Set or Delete command returns the following response:

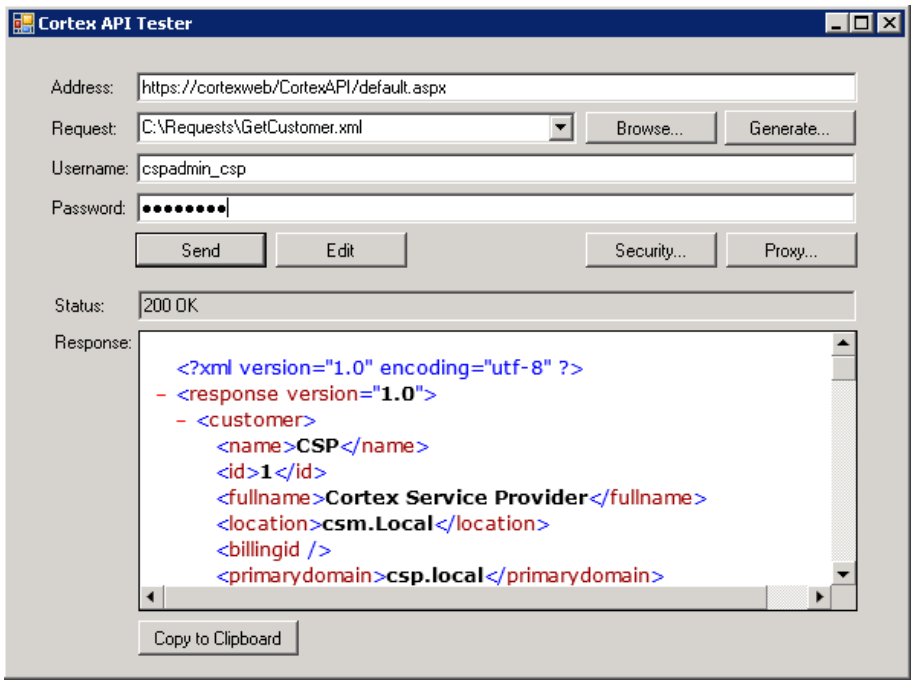
```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <request>
    <id>14452</id>
  </request>
</response>
```

If the following response is returned, then you need to refer to the error message to find out why the requested action could not be completed.

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
       <error>
              <id>###</id>
              <message>Text of error message</message>
       </error>
</response>
```

Cortex API Tester

The Cortex API Tester allows you to send request files to the API and view the responses it returns. It's not available on the Windows menu; however, it can be run directly from the API web service's binary folder. The file is called TestWebRequest.exe and is typically installed into the "C:\inetpub\Cortex Management\CortexAPI\bin" folder.



When using the tool, enter the following information:

- **Address:** CPSM API's web service address
- **Request:** The name of the file with the request
- **Username** and **Password:** The credentials used for the request

Additionally, the **Send** button sends the request and the **Generate** button creates new request files for customers, users, and services.

Basic API Customer Demo

The best way to learn how to effectively leverage the API is to test it in a lab environment.

In this demo, most of the code samples below provide the minimum parameters required for provisioning to occur. Some samples also include recommended parameters as well.
This demo customer will be called “Basic API Customer”, and we will provision services and users to it. We will then de-provision and delete all the entities.

Find Customer(s)

Step 1: Check whether or not the customer name you want to use is already taken by another customer. For this, use the Find command.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
       <customer>
              <fullname>Basic API Customer</fullname>
       </customer>
</request>
```

The Action attribute on the Request tag is set to FIND. Since we want to know whether or not the customer’s Fullname attribute exists, we put it inside the Customer tag. If the customer’s Fullname attribute doesn't exist the following response is returned:

```
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0" />
```

This response means no information was returned in from the Find request.

However, if a customer with the Fullname attribute of “Basic API Customer” does exist, the response appears as follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0">
       <customer>
              <name>BAC</name>
              <id>325</id>
              <fullname>Basic API Customer</fullname>
....
       </customer>
 </response>
```

In this response, the action returned information contained within a Customer tag. All tags in between the customer tags represent information about the customer specified in the Find request. If the customer is already there, you can alter the demo XML to use a new customer name or you can de-provision and delete the existing customer first.

If you want to include all the customers in the Find action, rather than just a specific one, use this request:

```
<?xml version="1.0" encoding="utf-8" ?>
<request version="1.0" action="FIND">
       <customer />
</request>
```

Create a Customer

When you create a customer in the CPSM control panel, there are required fields that you must complete. These fields are Full Name, Contact Name, Email Address, and Domain Name. These fields are also required for the CPSM API .

Request XML

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<fullname>Basic API Customer</fullname>
		<contactname>Admin</contactname>
		<contactemail>Admin@csm.local</contactemail>
		<primarydomain>csm.local</primarydomain>
		<parent>
			<fullname>Citrix Service Provider</fullname>
		</parent>
	</customer>
</request>
```

The action attribute on the Request tag is set to SET as we want to provision a customer. The tags in between the Customer tags are required for creating a customer as previously mentioned. The Parent tag specifies the Fullname attribute of the reseller under which this customer will be created in the customer hierarchy. Specifying the parent reseller in this way is not required but recommended if you are dealing with multiple tiers of resellers within the customer hierarchy.

Response XML

A response like this is returned for a successful request.

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <request>
    <id>14452</id>
  </request>
</response>
```

If you leave one of the required fields empty, an error like this is returned.

```
Bad Request 6 Unable to create customer. A new customer must specify the name,
fullname, contactname, contactemail and primarydomain.
```

You don't have to specify a Name tag because CPSM automatically generates the short code (short name) for the customer. However, if you want the customer to have a specific code, you can specify it here.

Provision/Find Users

Provision Users

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<user>
			<name>User1</name>
		</user>
	</customer>
</request>
```

The Action attribute on the Request tag is set to SET as we want to create a user. The Name tag in the Customer tag contains the short name of the customer to whom the user belongs. In the User tag, we specify the Name attribute of the user. The user's upn, firstname, lastname, and externalemail attributes default to your environment settings.

Find Users

If you want to find all the users for a customer, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
	<customer>
		<fullname>Basic API Customer</fullname>
		<user />
	</customer>
</request>
```

Provision a Customer Service – Hosted Exchange

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
	<customer>
		<name>BAC</name>
		<service>
			<name>HE</name>
			<version>2010</version>
			<userpackage>
				<fullname>Ex10 Basic</fullname>
				<enabled>True</enabled>
			</userpackage>
			<package>
				<name>Basic</name>
				<enabled>True</enabled>
			</package>
		</service>
	</customer>
</request>
```

The Action attribute on the Request tag is set to SET as we want to provision the Hosted Exchange service to a customer. The Name tag in the Customer tag expects the customer short name. The Service tag specifies that we want to alter details about a service. In the sample code above, we want to provision the service. So, we need to specify the Name attribute of the service. Additionally, we need to specify the Version attribute for the version of Exchange that users will receive.

The Userpackage tag is the user plan that can be provisioned to users. For backward compatibility, Hosted Exchange is the only service where this tag is named “userpackage.” For all other services, the tag is named “userplan.”

The Userpackage tag must include an Enabled tag set to True; otherwise, it will be disabled by default. The Package tag is the package that will be provisioned to the customer. Commonly, packages are called “Basic” and “Public Folders.”

Provision a User Service – Hosted Exchange

The syntax to provision a user with a service is very similar to provisioning a customer with a service.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<user>
			<name>User1</name>
			<service>
				<name>HE</name>
				<userpackage>
					<fullname>Ex10 Basic</fullname>
				</userpackage>
			</service>
		</user>
	</customer>
</request>
```

Here, the Service tag is contained within the User tag. We do not have to specify the Exchange version or the package because they are defined at the customer level.

De-provision a User/Customer Service and the Customer itself

De-provisioning a service is very simple as you only have to specify the minimum requirements for the API to find the service.

De-provision a User Service – Hosted Exchange

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<user>
			<name>User1</name>
			<service>
				<name>HE</name>
				<status>NotProvisioned</status>
			</service>
		</user>
	</customer>
</request>
```

The Action attribute on the Request tag is set to SET as we want to alter the status of a service provisioned to the user. We add the Status tag as it specifies the status we want to set. The valid statuses are Provisioned, Failed, Pending, Requested, InProgress, and NotProvisioned.

De-provision a Customer Service – Hosted Exchange

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<service>
			<name>HE</name>
			<status>Notprovisioned</status>
		</service>
	</customer>
</request>
```

In this example, we want to de-provision the customer service. By doing so, CPSM de-provisions all users with the HE (Hosted Exchange) service. The Status tag is contained within the Service tag.

De-provision a Customer

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<status>Notprovisioned</status>
	</customer>
</request>
```

Here we want to de-provision the customer which will de-provision all users and their services. The Status tag is contained within the Customer tag.

Delete a User/Customer

WARNING: Please take care when using the Delete action. Unlike the CPSM UI, you may specify a Delete action to a user or customer that is yet to be de-provisioned. If you make a mistake, the user or customer might lose their information.

Delete User

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="DELETE">
	<customer>
		<name>BAC</name>
		<user>
			<name>User1</name>
		</user>
	</customer>
</request>
```

The Action attribute on the Request tag is set to DELETE as we want delete the user. As you can see, the same details are required for deleting the user as for de-provisioning, except you do not need to specify the Status tag.

Error Message

If you have been following all the examples up to this point, you should have one customer with one user in your environment. So, when you run the Delete User script, you will get the following error message:

```
<message>Unable to delete the last user from the customer</message>
```

CPSM requires at least one user to exist for each customer. So, if the customer has only one user, CPSM will not allow you to delete it.
Before you test this script, create another user.

Delete a Customer

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="DELETE">
	<customer>
		<name>BAC</name>
	</customer>
</request>
```

The Action attribute on the Request tag is set to DELETE as we want delete the customer. In the example above, the only tag required is the customer's Name tag.

Creating SET requests

SET requests are requests that update something. They are designed to be somewhat discoverable. You can use the result of a GET request to discover what can be sent in a SET request.

SET request discovery process

The response from a GET request can be used to create a SET request. The process goes like this:

1. Send a FIND request to get the name of the customer, customer service, user, or user service.
2. Send a GET request with this name to get the customer or user's details.
3. Convert the response of the GET request into a SET request.
4. Remove any tags that don't need to change to create a template.
5. Update the remaining tags with the values you want and send the new request.

Step 1: Send a FIND request

For this example, we know the name of a customer and want to change a property of the File Sharing service. We send a FIND request to find all services for the customer:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
       <customer>
              <fullname>ATest Customer</fullname>
              <service/>
       </customer>
</request>
```

This request returns the name of the File Sharing service:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <customer>
    <name>ATest</name>
    <id>498</id>
    <fullname>ATest Customer</fullname>
    <billingid />
    <primarydomain>atest.local</primarydomain>
    <status>Provisioned</status>
…
    <service>
      <name>FSS</name>
      <fullname>File Sharing</fullname>
      <location>csm.Local</location>
      <status>Provisioned</status>
    </service>
…
  </customer>
</response>
```

Step 2: Send a GET request

We update the request to include the service name from our last response and change the action to GET:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
       <customer>
              <fullname>ATest Customer</fullname>
              <service>
                     <name>FSS</name>
              </service>
       </customer>
</request>
```

The request returns detailed information about both the customer and the service:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <customer>
    <name>ATest</name>
    <id>498</id>
    <fullname>ATest Customer</fullname>
…
    <service>
      <name>FSS</name>
      <fullname>File Sharing</fullname>
      <location>csm.Local</location>
      <userlimit>Unlimited</userlimit>
      <isbilled>True</isbilled>
      <status>Provisioned</status>
      <approvalpending>False</approvalpending>
      <userplan>
        <name>FULL</name>
        <fullname>Full</fullname>
        <userlimit>Unlimited</userlimit>
        <used>False</used>
        <enabled>True</enabled>
      </userplan>
      <userplan>
        <name>READ</name>
        <fullname>Read</fullname>
        <userlimit>Unlimited</userlimit>
        <used>False</used>
        <enabled>True</enabled>
      </userplan>
      <userplan>
        <name>LIST</name>
        <fullname>List</fullname>
        <userlimit>Unlimited</userlimit>
        <used>False</used>
        <enabled>True</enabled>
      </userplan>
      <package>
        <name>FULL</name>
        <fullname>FULL</fullname>
        <enabled>True</enabled>
      </package>
    </service>
  </customer>
</response>
```

Step 3: Convert the GET response to a SET request

Change the response tag into a request tag and add the SET action attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
  <customer>
    <name>ATest</name>
    <id>498</id>
    <fullname>ATest Customer</fullname>
…
</request>
```

Step 4: Remove unneeded tags

Remove any settings that you don't want to change. In this example, we only want to change the user limit:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
  <customer>
    <name>ATest</name>
    <service>
      <name>FSS</name>
      <userlimit>Unlimited</userlimit>
    </service>
  </customer>
</request>
```

Step 5: Update values and send

Update the user limit to **5** and send the request:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
  <customer>
    <name>ATest</name>
    <service>
      <name>FSS</name>
      <userlimit>5</userlimit>
    </service>
  </customer>
</request>
```

Advanced properties

Customers, customer services, users, and user services have properties that are not returned in a default GET request. Adding a Properties tag to these items returns more information in the response. The optional Detail attribute returns even more information and should only be added if required. Similar to the example in the previous section, these properties can then be used in a SET request.

For customers, use the Properties tag with the optional Detail attribute. The details attribute returns a much more detailed description of the property which isn't normally required.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
  <customer>
    <name>ATest</name>
    <properties detail="true"/>
  </customer>
</request>
```

For users, you can use the Properties tag and the Additionalproperties tag. The Additionalproperties tag represents the users' Active Directory properties. Only use the Detail attribute if required.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
  <customer>
    <name>ATest</name>
    <user>
      <name>test_ATest</name>
      <properties detail="true"/>
      <additionalproperties/>
    </user>
  </customer>
</request>
```

For customer and user services, there is an optional Filter attribute that can be used to show hidden properties. Only use the Detail attribute if required.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
  <customer>
    <name>ATest</name>
    <service>
      <name>FSS</name>
      <properties detail="true" filter="all"/>
    </service>
  </customer>
</request>
```

The Properties tag can also be used for customer and user plans, with the optional Filter attribute. Only use the Detail attribute if required.

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
  <customer>
    <name>ATest</name>
    <service>
      <name>FSS</name>
      <userplan>
        <name>FULL</name>
        <properties detail="true" filter="all"/>
      </userplan>
    </service>
  </customer>
</request>
```

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
  <customer>
    <name>ATest</name>
    <user>
      <name>test_ATest</name>
      <service>
        <name>FSS</name>
        <userplan>
          <name>FULL</name>
          <properties detail="true" filter="all"/>
        </userplan>
      </service>
    </user>
  </customer>
</request>
```

Sending an API request

PowerShell Example

Here's a sample PowerShell script that uses the .NET WebClient to send a request to the API:

```
##################################
# Sample API Request in PowerShell
##################################

# API Request to find the CSP Customer
# Note: Be careful when using @" "@ Here-String PowerShell construction format to
create the request.
# Extra spacing could cause "bad request" to be returned. However, Here-String
makes this request much more readable.
$apiRequest = @"
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
      <customer>
            <name>CSP</name>
      </customer>
</request>
"@

# URL to the CortexAPI
$apiUrl = "http://cortexweb/cortexapi/default.aspx"

# Initialize the client
$client = New-Object System.Net.WebClient
$client.Credentials = New-Object
System.Net.NetworkCredential("cspadmin_csp","<YourPassword>")
$client.UploadString($apiUrl,$apiRequest)
```

The response was a string dumped to the console by the "UploadString()" method call on the .NET WebClient object:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <customer>
    <name>CSP</name>
    <id>1</id>
    <fullname>Cortex Service Provider</fullname>
    <billingid />
    <primarydomain>csp.local</primarydomain>
    <status>Provisioned</status>
  </customer>
</response>
```

VBScript Example

```
'################################
'# Sample API Request in VBScript
'################################

' API Request to find all customers

On Error Resume Next

Dim Request, Response

Const Url      = "http://cortexweb/cortexapi/default.aspx"
Const Username = "cspadmin_csp"
Const Password = "<YourPassword>"

Request = "<?xml version=""1.0"" encoding=""utf-8""?>" & _
          "<request action=""FIND"" version=""1.0"">" & _
          "  <customer/>" & _
          "</request>"

Err.Clear
Set Response = SendRequest(Url, Username, Password, Request)
If Err.Number<>0 Then
    WScript.Echo "Failed [0x" & Hex(Err.Number) & "] " & Trim(Err.Description)
Else
    If Response.Status = 200 Then
        WScript.Echo "OK"
    Else
        WScript.Echo "Failed [" & Response.Status & "] " & Response.StatusText
        Set ResponseNode = Response.responseXML.DocumentElement
        WScript.Echo ">>" &
ResponseNode.SelectSingleNode("/response/error/message").Text
    End If
    WScript.Echo
    WScript.Echo Response.responseText
End If

Private Function SendRequest(ByVal Url, ByVal Username, ByVal Password, ByVal Request)
    Dim XmlHttp

    Set XmlHttp = CreateObject("MSXML2.XmlHttp")
    XmlHttp.Open "POST", Url, False, Username, Password
    XmlHttp.SetRequestHeader "Content-Type", "text/xml"
    XmlHttp.Send Request
    Set SendRequest = XmlHttp
End Function
```

Useful Techniques

Find all Customers' Users

To find all of the users provisioned for the customer "CSP" use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" trace="true" version="1.0">
  <customer>
    <name>CSP</name>
    <user />
  </customer>
</request>
```

Find all Services for a Customer

To find all of the services (including those that are not provisioned) that are available to a customer, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" trace="false" version="1.0">
   <customer>
      <name>CSP</name>
      <service />
   </customer>
</request>
```

Find all Services for a User

To find all of the services or userplans (including those that are not provisioned) that are available to a user, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" trace="false" version="1.0">
  <customer>
    <name>CSP</name>
    <user>
      <name>cspadmin_CSP</name>
      <service />
    </user>
  </customer>
</request>
```

Find all Services Provisioned to Customer

To find information about all the services a customer was provisioned, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" version="1.0">
       <customer>
              <name>BAC</name>
              <service />
       </customer>
</request>
```

Find all Services a Reseller can Resell

To see all the services a reseller can resell, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
	<customer>
		<name>AA</name>
		<service>
			<name>Reseller</name>
		</service>
	</customer>
</request>
```

You can also use this with multiple locations. The Fullname setting will be different, but the Name setting will always be "Reseller."

Multi-Item Provision

Instead of sending the same request multiple times to provision users, you can chain the items into one request which can be done as many times as you require. This is useful for multiple users, services, and service settings.

Multi-User Provision

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<name>BAC</name>
		<user>
			<name>User3</name>
		</user>
		<user>
			<name>User4</name>
		</user>
	</customer>
</request>
```

In this example, we are provisioning two users. By closing and opening the User tag, both users will be provisioned under customer BAC.

Multi-User Plan (userpackage) Provisioning

This example is really helpful for provisioning multiple user plans in the Hosted Exchange Example.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
	<customer>
		<name>BAC</name>
		<service>
			<name>HE</name>
			<version>2010</version>
			<userpackage>
				<fullname>Ex10 Basic</fullname>
				<enabled>True</enabled>
			</userpackage>
			<userpackage>
				<fullname>Ex10 Full</fullname>
				<enabled>True</enabled>
			</userpackage>
			<package>
				<name>Basic</name>
				<enabled>True</enabled>
			</package>
		</service>
	</customer>
</request>
```

Create a Customer and Provision the Hosted Exchange Service

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="SET">
	<customer>
		<fullname>Basic API Customer</fullname>
		<contactname>Admin</contactname>
		<contactemail>Admin@csm.local</contactemail>
		<primarydomain>csm.local</primarydomain>
		<parent>
			<fullname>Citrix Service Provider</fullname>
		</parent>
		<service>
			<name>HE</name>
			<version>2010</version>
			<userpackage>
				<fullname>Ex10 Basic</fullname>
				<enabled>True</enabled>
			</userpackage>
			<package>
				<name>Basic</name>
				<enabled>True</enabled>
			</package>
		</service>
	</customer>
</request>
```

Get Customer Information

The easiest way to create an API request is to use the Get command. Using this command with a customer returns the requested information in a format that you can cut and paste into a Set action and use again. To reuse the information, you change the Response tag to “request,” change the request action to “SET,” and modify or add any other settings according to the information you want to create or update.

```
<?xml version="1.0" encoding="utf-8"?>
<request action='GET' version='1.0'>
       <customer>
              <name>BAC</name>
       </customer>
</request>
```

Advanced Techniques

The following sections show how to use the Get and Set actions to work with extra properties or information. Altering some these properties should be done with caution as several of them set system configurations and should never be altered.

Get all User plans and Packages

If you wish to see all the user plan and package names under a service, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
	<customer>
		<name>BAC</name>
		<service>
			<name>HE</name>
		</service>
	</customer>
</request>
```

Get all Instances for Multi-Instance Services

To view information about the instances of the CRM 2011, SharePoint 2010, Microsoft SQL Server Hosting, or Windows Web Hosting services, use this request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
	<customer>
		<name>BAC</name>
		<service>
			<name>Sharepoint2010</name>
			<instances />
		</service>
	</customer>
</request>
```

Identity

With the identity tag you can get information about the requesting user.

Get requesters' details

The requesting user is the user whose credentials were used to authenticate the API request. To get information about the requesting user, send the following request:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <identity/>
</request>
```

The request returns something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <identity>
    <customer>
      <name>CSP</name>
      <id>1</id>
      <fullname>Cortex Service Provider</fullname>
      <billingid />
      <primarydomain>csp.local</primarydomain>
      <user>
        <name>cspadmin_CSP</name>
        <id>1</id>
        <fullname>CSPAdmin</fullname>
        <upn>cspadmin@csp.local</upn>
        <location>Unassigned</location>
        <department>Unassigned</department>
      </user>
    </customer>
  </identity>
</response>
```

Global

The Global tag returns version information about the API and the database:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <global/>
</request>
```

The request returns something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <global>
    <version>
      <api>11.0.0.0</api>
      <database>11.0.0.0</database>
    </version>
  </global>
</response>
```

Locations

The Location tag allows you to manage CPSM locations. You can FIND, GET, SET, and DELETE locations.

Finding Locations

To get a list of locations:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
    <location/>
</request>
```

To get the services available at each location:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
    <location>
        <service/>
    </location>
</request>
```

You can also specify a service name or a service fullname to get a specific service:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="FIND">
    <location>
        <service>
          <name>HE</name>
        </service>
    </location>
</request>
```

To get detailed information about a location:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <location>
        <name>csm.Local</name>
    </location>
</request>
```

Get Customer Properties for a Location

This request gets the properties available for a customer in the specified location:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <location>
        <name>csm.Local</name>
        <customer>
            <customerproperties/>
        </customer>
    </location>
</request>
```

Get User Properties for a Location

This request gets the Active Directory properties for a user in the specified location:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <location>
        <name>csm.Local</name>
        <user>
            <additionalproperties/>
        </user>
    </location>
</request>
```

Provisioning Requests

Every time you provision or delete a customer, user, or service, you generate one or more provisioning requests that carry out the desired action. These actions consist of a top level request that has one or more children. The ID of the top level request is returned in the response to SET and DELETE actions.

Get Requests

To get a request with its ID:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <request>
        <id>14882</id>
    </request>
</request>
```

To get all children requests with the request's parent ID:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <request>
        <parentid>14882</parentid>
    </request>
</request>
```

Get the Logging from a Request

To get the current logging for a request:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <request>
        <id>14882</id>
        <log/>
    </request>
</request>
```

Get the Processing Times of a Request

To get the times of various stages of request processing:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1.0" action="GET">
    <request>
        <id>14882</id>
        <times/>
    </request>
</request>
```

Querying Data

Although CPSM has a comprehensive Data Warehouse with its own API, you might need the results of a simple query or stored procedure. That's where this simple reports feature is useful.

You can use the CPSM API to run simple report queries on the CPSM database and return the results in XML or tab delimited (TSV) format. To generate these queries, perform the following actions:

1. On the CPSM web server, open the web.config file for the API. This is located in the /%Program Files (x86)%/Citrix/Cortex/CortexWeb/CortexAPI/ directory.
2. Copy the connection string value for the APIConnectionString key.
3. On the CPSM database server hosting the OLM database, open the **dbo.ReportQueries** table and update all reports with the connection string that you have copied from the API's web.config file. The dbo.ReportsQueries table defines the report queries that can be executed by the CPSM API. You can add more report queries to this table.

WARNING: If you add your own report queries, they can potentially expose information about customers that the API caller would not normally have permissions to view. To mitigate this situation, the customer ID of the API caller is automatically added to the GET report query parameters. The query or stored procedure should use this ID to limit the results returned. See the example 'Customers with Query' in the **dbo.ReportQueries** table that uses the tf_CustomerHierarchy function. The tf_CustomerHierarchy function returns the ID of all customers under the specified customer ID (if the second parameter is 1 it includes the specified customer ID as well). This is then used in a join to filter the results to customers under the API caller's customer:

select customers.* from customers inner join tf_CustomerHierarchy(@CallingCustomerID,1) ch ON ch.CustomerID = customers.customerID where Name like @Name

Find Reports

The following request returns all reports that are configured in the dbo.ReportQueries table:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" version="1.0">
       <report />
</request>
```

For each report, a list of parameter names is displayed. These parameters can be used to specify the context of the report to be produced.

Get Reports

The Get action attribute will generate a report via the API.

The following tags can be used:

Tag	Description
<name>	The name of the report.
<parameter name="[ParameterName]">	The report 's parameter name . A value should be entered for the parameter tag.
<timeout>	Sets a timeout period in milliiseconds. The value 0 indicates no timeout.
<column>	Returns the data for the specified column only.

The format that the API will return the data can be defined by using one of the following options:

Format	Description
<dataformat columnnames="[True]">xml</dataformat>	Report is generated in XML. If **columnnames** is set to **True**, the column names appear in the returned tags.
<dataformat header="[True]">tsv</dataformat>	Report is generated in TSV format. If **header** is set to **True**, the column names appear at the top of the report.

The following API query returns all customers where the customer name begins with “C.” The report will be in XML format with the column names appearing in the returned tags.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
       <report>
            <name>Customers with Stored Procedure</name><!-- Name of the Report -->
            <parameter name=”Name”>C%</parameter>
            <dataformat columnnames=”true”>xml</dataformat>
       </report>
</request>
```

The API returns the data as:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1.0">
  <report>
    <name>Customers with Stored Procedure</name>
    <description />
    <result>
      <column type="int">CustomerID</column>
      <column type="int">LocationID</column>
      <column type="nvarchar">Name</column>
      <column type="nvarchar">Label</column>
      <column type="nvarchar">BillingID</column>
      <column type="nvarchar">PrimaryDomain</column>
      <column type="int">StatusID</column>
      <row>
        <CustomerID>3</CustomerID>
        <LocationID>1</LocationID>
        <Name>CS</Name>
        <Label>Citrix Systems</Label>
        <BillingID />
        <PrimaryDomain>citrix.com</PrimaryDomain>
        <StatusID>1</StatusID>
      </row>
      <row>
        <CustomerID>1</CustomerID>
        <LocationID>1</LocationID>
        <Name>CSP</Name>
        <Label>Cortex service Provider</Label>
        <BillingID />
        <PrimaryDomain>csp.local</PrimaryDomain>
        <StatusID>1</StatusID>
      </row>
    </result>
  </report>
</response>
```

The following API query returns the details for customer code “CSP.” The report will be in TSV.format with the column names shown. There is no timeout set for the report query.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
       <report>
              <name>Customers with Query</name>
              <timeout>0</timeout>
              <parameter name="Name">CSP</parameter>
              <dataformat header="true">tsv</dataformat>
       </report>
</request>
```

The API returns the data as:

```
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0">
  <report>
    <name>Customers with Query</name>
    <description />
    <result>
      <column type="int">CustomerID</column>
      <column type="nvarchar">Name</column>
      <column type="nvarchar">Label</column>
      <column type="nvarchar">Location</column>
      <column type="nvarchar">ContactName</column>
      <column type="nvarchar">ContactEMail</column>
      <column type="nvarchar">Description</column>
      <column type="datetime">Created</column>
      <column type="nvarchar">PhoneNumber</column>
      <column type="nvarchar">FaxNumber</column>
      <column type="int">MinPWLength</column>
      <column type="int">BannerPWDays</column>
      <column type="int">ModifiedBy</column>
      <column type="datetime">DateDisabled</column>
      <column type="datetime">DateDeleted</column>
      <column type="nvarchar">BrandName</column>
      <column type="nvarchar">OU_Name</column>
      <column type="int">ObjectID</column>
      <column type="nvarchar">BillingID</column>
      <column type="int">ImpersonatingUserID</column>
      <column type="nvarchar">BrandType</column>
      <column type="int">ParentID</column>
      <data>
<![CDATA[
CustomerID   Name   Label  Location      ContactName   ContactEMail  Description
       Created       PhoneNumber   FaxNumber     MinPWLength   BannerPWDays
       ModifiedBy    DateDisabled  DateDeleted   BrandName     OU_Name       ObjectID
       BillingID     ImpersonatingUserID BrandType     ParentID
1      CSP    Cortex service Provider           CSP Admin     cspadmin@csp.local
       7/29/2012 8:36:22 PM                     0      0      0
       Default       customers     36543         0      DNS
]]>
      </data>
    </result>
  </report>
</response>
```

Examples

This section includes examples for how to query and set core services.

In these examples we will be querying the details of a service for the "CSP" customer.

Querying Services

When you query a service that has not yet been provisioned to a customer, the results show you what will happen by default when the service is provisioned to the customer, but no customer plans are specified. For instance, assume that a service has three customer plans and three user plans. If you query a customer that does not yet have the service provisioned, the query yields the following results:

- The first customer plan has a status of <enabled>True<enabled>.
- The other customer plans has a status of <enabled>False<enabled>.
- All of the user plans have a status of <enabled>True<enabled>.

CRM 2011 for a Customer

Assume that the CRM 2011 service provisioned to this customer has an instance named "Site01."

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0" trace="true">
  <customer>
    <name>CSP</name>
    <service>
      <name>CRM2011</name>
      <instances>
        <instance>
          <name>Site01</name>
        </instance>
      </instances>
    </service>
  </customer>
</request>
```

To query all of the instances for that customer's service:

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0" trace="true">
  <customer>
    <name>CSP</name>
    <service>
      <name>CRM2011</name>
      <instances />
    </service>
  </customer>
</request>
```

CRM 2011 for a User

Assume that we wish to query the CRM 2011 service that was provisioned to user01_CSP for the Site01 instance of CRM.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0" trace="true">
  <customer>
    <name>CSP</name>
    <user>
      <name>user01_CSP</name>
```

```
      <service>
        <name>CRM2011</name>
        <instances>
          <instance>
            <name>Site01</name>
          </instance>
        </instances>
      </service>
    </user>
  </customer>
</request>
```

File Sharing Service for a Customer

The File Sharing Service has both customer plans and user plans, and it is not a multi-instance service. However, it does have some specific service settings, so we need to use the <properties> tag.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name>FTC</name>
    <service>
      <name>FSS</name>
      <properties filter="all" />
    </service>
  </customer>
</request>
```

Hosted Apps & Desktops for a Customer

The Hosted Apps and Desktops service also has both customer plans and user plans:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name>XTC</name>
    <service>
      <name>HostedAppsAndDesktops</name>
    </service>
  </customer>
</request>
```

Hosted Apps & Desktops for a User

Users provisioned with the Hosted Apps and Desktops service can have multiple user plans.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" trace="false" version="1.0">
  <customer>
    <name>XTC</name>
    <user>
      <name>barryh_XTC</name>
        <service>
          <name>HostedAppsAndDesktops</name>
        </service>
    </user>
  </customer>
</request>
```

Mail Archiving for a Customer

Mail Archiving is fairly straightforward . To see all the properties, use a filter, as shown.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
  <customer>
    <name> TestCust01</name>
    <service>
      <name>MARCH</name>
      <properties filter="all" />
    </service>
  </customer>
</request>
```

MySQL for a Customer

The MySQL service provisions database resources to customers. You can query the available resources that are provisioned by including the <resourceconfiguration /> tag in the request.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0">
  <customer>
    <name>crc</name>
    <service>
      <name>MySQL</name>
      <resourceconfiguration />
    </service>
  </customer>
</request>
```

SharePoint 2010 for a Customer

The SharePoint 2010 service is a multi-instance service, similar to the CRM 2011 service.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
  <customer>
    <name>CSP</name>
    <service>
      <name>SharePoint2010</name>
      <resourceconfiguration />
    </service>
  </customer>
</request>
```

To get a list of all properties:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name>CSP</name>
    <service>
      <name>Sharepoint2010</name>
      <properties filter="all" />
      <instances />
    </service>
  </customer>
</request>
```

To get all the data in a single request, you can combine the above two queries, including both the <resourceconfiguration /> and <instances /> tags in the same request.

SharePoint 2010 Sites for a Customer

An example of how to query the SharePoint 2010 sites that were provisioned to a customer named "CSP."

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name>CSP</name>
    <service>
      <name>Sharepoint2010</name>
      <properties filter="all" />
      <instances />
    </service>
  </customer>
</request>
```

SharePoint Sites for a User

An example of how to query the SharePoint sites that were provisioned to user "FTC_u1_CSP" of a customer named "CSP."

```
<?xml version="1.0" encoding="utf-8"?>
<request action="FIND" version="1.0">
  <customer>
    <name>CSP</name>
    <user>
      <name>FTC_u1_CSP</name>
      <service>
        <name>Sharepoint2010</name>
        <instances />
      </service>
    </user>
  </customer>
</request>
```

Provisioning Services

In the following examples we provision services to a customer named "CSP."

If you provision a service to a customer and you don't specify the customer plan to use, the first customer plan is automatically provisioned by default. Likewise, if you don't specify any user plans, then all user plans are enabled. If you want some user plans to be disabled for the customer, you must set those user plans to <enabled>False<enabled> in the request.

File Sharing Service to a Customer

The File Sharing Service has one required setting: FileShareServer. This value must be the name of a file server that already exists in the environment.

The File Sharing Service has user plans, but they are built-in and cannot be changed. Omitting the <userplan> tag does not cause any problems, and all three user plans can be provisioned to a user afterwards.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>ZCSP</name>
    <service>
      <name>FSS</name>
      <status>Provisioned</status>
      <properties>
        <property>
          <name>FileShareServer</name>
          <value>CSPFILEHOST01</value>
        </property>
```

```
      </properties>
      <package>
        <name>Basic</name>
        <enabled>True</enabled>
      </package>
    </service>
  </customer>
</request>
```

Hosted Apps and Desktops to a Customer

The following example provisions the "Shared Site Web Interface" customer plan to the user and enables the "Desktops" user plan for that customer.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>XTC</name>
    <service>
      <name>HostedAppsandDesktops</name>
      <package>
        <name>SharedsiteWebInterfacemode</name>
        <enabled>True</enabled>
      </package>
      <userplan>
        <name>DESKTOPS</name>
        <enabled>True</enabled>
      </userplan>
    </service>
  </customer>
</request>
```

Hosted Apps and Desktops to a User

The Hosted Apps and Desktops service allows you to provision multiple user plans to a user.

The following example provisions the "Office Apps" and "Desktops" user plans to the user.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>XTC</name>
    <user>
      <name>user1_xtc</name>
      <service>
        <name>HostedAppsandDesktops</name>
        <userplan>
          <name>DESKTOPS</name>
        </userplan>
        <userplan>
          <name>OFFICEAPPS</name>
        </userplan>
      </service>
    </user>
  </customer>
</request>
```

Hosted Exchange

Get Distribution Groups

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0">
 <customer>
   <fullname>Justice League Unlimited</fullname>
   <service>
     <name>HE</name>
     <distributiongroups />
   </service>
 </customer>
</request>
```

Get Distribution Groups with a Filter

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0">
 <customer>
   <fullname>Justice League Unlimited</fullname>
   <service>
     <name>HE</name>
     <distributiongroups>
       <displaynamefilter>M</displaynamefilter>
       <emailfilter>M</emailfilter>
       <pageindex>1</pageindex>
       <pagesize>1000</pagesize>
     </distributiongroups>
   </service>
 </customer>
</request>
```

Get Distribution Group Members

Distribution group members are not returned by default to include members in the list, you can use the following syntax:

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <distributiongroups>
        <distributiongroup>
          <members />
          <sendrestrictionacceptedmembers />
          <sendrestrictionrejectedmembers />
          <sendasmembers />
        </distributiongroup>
      </distributiongroups>
    </service>
  </customer>
</request>
```

For synchronization purposes, it is possible to request distribution group details for a specific object identifier (objectSid or objectGuid) from a remote domain, and to query for the status of members from the remote domain using the following syntax:

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <distributiongroups>
        <distributiongroup>
```

```
          <syncobjectid>S-1-5-21-1816066181-3380177551-1806815814-
1235</syncobjectid>
          <members>
            <member>
              <syncobjectid>B5EF275A-D953-4FBC-9E89-B72AC0470278</syncobjectid>
            </member>
          </members>
          <sendrestrictionacceptedmembers />
          <sendrestrictionrejectedmembers />
          <sendasmembers />
        </distributiongroup>
      </distributiongroups>
    </service>
  </customer>
</request>
```

The returned distribution group contains member lists indicating whether or not each member exists in the managed domain and whether or not they are a group member in the managed domain.

Additional tags may be provided to filter by display name, email, and to retrieve a specified page size and index.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <distributiongroups>
        <displaynamefilter></displaynamefilter>
        <emailfilter></emailfilter>
        <pageindex></pageindex>
        <pagesize></pagesize>
      </distributiongroups>
    </service>
  </customer>
</request>
```

Get Distribution Groups of which the User is a Member

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" version="1.0">
 <customer>
   <fullname>Justice League Unlimited</fullname>
   <user>
     <name>Donald_E2010</name>
     <service>
       <name>HE</name>
       <distributiongroups />
     </service>
   </user>
 </customer>
</request>
```

Add a User to a Distribution Group

To add or modify a distribution group, you use a SET request. This request requires the customer and service name, as well as the <distributiongroups> tag with the <distributiongroup> tag containing the group details. The <syncobjectid> tag may be specified to indicate that the group has been synced from a remote domain.

You can set the <selected> tag on <members>, <sendasmembers>, <sendrestrictionacceptedmembers>, and <sendrestrictionrejectedmembers> to True or False and indicate whether the member should be added or removed from the group. It is not necessary to provide all members when submitting a request; if a member is not included in the request, it will not be modified. Use the <syncobjectid> tag for members to perform a search in the managed domain for users, contacts, or distribution groups.

Specify the full list of owners as existing owners will be replaced with the owners from the request when the <owners> tag is provided. The <syncobjectid> tag for owners is used to perform a search in the managed domain for users only.

Similarly, you must specify the full list of email addresses as existing email addresses will be replaced with the email addresses from the request when the <emailaddresses> tag is provided.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="SET" version="1.0">
 <customer>
   <fullname>Justice League Unlimited</fullname>
   <user>
     <name>Batman_JLU</name>
     <service>
       <name>HE</name>
       <distributiongroups>
         <distributiongroup>
           <path>CN=Heroes,OU=Distribution Groups,OU=Justice League
Unlimited(JLU),OU=Customers,DC=dev,DC=local</path>
           <selected>true</selected>
         </distributiongroup>
         <distributiongroup>
           <displayname>Enemies</displayname>
           <selected>false</selected>
         </distributiongroup>
       </distributiongroups>
     </service>
   </user>
 </customer>
</request>
```

A more complete request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <distributiongroups>
        <distributiongroup>
          <displayname></displayname>
          <syncobjectid></syncobjectid>
          <enabled></enabled>
          <path></path>
          <email></email>
          <emailalias></emailalias>
          <mailtip></mailtip>
          <grouptype></grouptype>
          <hidefromaddresslists></hidefromaddresslists>
          <requiresenderauthentication></requiresenderauthentication>
          <memberjoinrestriction></memberjoinrestriction>
          <memberdepartrestriction></memberdepartrestriction>
          <sendmoderationnotifications></sendmoderationnotifications>
          <emailaddresses>
            <emailaddress></emailaddress>
          </emailaddresses>
          <owners>
            <owners>
              <displayname></displayname>
              <syncobjectid></syncobjectid>
              <path></path>
              <email></email>
            </owners>
          </owners>
```

```
          <members>
            <member>
              <displayname></displayname>
              <syncobjectid></syncobjectid>
              <path></path>
              <email></email>
              <enabled></enabled>
            </member>
          </members>
          <sendasmembers>
            <sendasmember>
              <displayname></displayname>
              <syncobjectid></syncobjectid>
              <path></path>
              <email></email>
              <enabled></enabled>
            </sendasmember>
          </sendasmembers>
          <sendrestrictionacceptedmembers>
            <sendrestrictionacceptedmember>
              <displayname></displayname>
              <syncobjectid></syncobjectid>
              <path></path>
              <email></email>
              <enabled></enabled>
            </sendrestrictionacceptedmember>
          </sendrestrictionacceptedmembers>
          <sendrestrictionrejectedmembers>
            <sendrestrictionrejectedmember>
              <displayname></displayname>
              <syncobjectid></syncobjectid>
              <path></path>
              <email></email>
              <enabled></enabled>
            </sendrestrictionrejectedmember>
          </sendrestrictionrejectedmembers>
        </distributiongroup>
      </distributiongroups>
    </service>
  </customer>
</request>
```

Tag	Tag Options
<grouptypes>	• Universal_Distribution • Global_Distribution • Local_Distribution • Universal_Security • Global_Security • Local_Security
<memberjoinrestriction>	• Open • Closed • ApprovalRequired
<memberdepartrestriction>	• Open • Closed • ApprovalRequired
<sendmoderationnotification>	• True • False
<hidefromaddresslists>	• True • False
<requiresenderauthentication>	• True

	• False

Get Contacts

To retrieve a list of contacts, use a GET request. This request requires the customer name and service name, as well as the <contacts /> tag which includes the list of contacts in the response.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <contacts />
    </service>
  </customer>
</request>
```

You can provide additional tags to filter by display name or the sync object identifier, and to retrieve a specified page size and index.

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="GET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <contacts>
        <displaynamefilter></displaynamefilter>
        <syncfilter></syncfilter>
        <pageindex></pageindex>
        <pagesize></pagesize>
      </contacts>
    </service>
  </customer>
</request>
```

Set Contact

To add or modify a contact, a SET request is used. This request requires the customer and service name, as well as the <contacts> tag with the <contact> tag containing the contact details. You can specify the <syncobjectid> tag to indicate that the contact has been synced from a remote domain.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name></name>
    <service>
      <name>HE</name>
      <contacts>
        <contact>
          <displayname></displayname>
          <syncobjectid></syncobjectid>
          <enabled></enabled>
          <path></path>
          <email></email>
          <emailalias></emailalias>
          <firstname></firstname>
          <lastname></lastname>
          <initials></initials>
          <jobtitle></jobtitle>
          <companyname></companyname>
          <businessphone></businessphone>
          <businessfax></businessfax>
```

```
          <homephone></homephone>
          <mobile></mobile>
          <webpage></webpage>
          <street></street>
          <city></city>
          <state></state>
          <country></country>
          <postcode></postcode>
          <hidefromaddresslists></hidefromaddresslists>
        </contact>
      </contacts>
    </service>
  </customer>
</request>
```

Users

Provisioning Users

To create a new user with a specific password:

```
<?xml version="1.0" encoding="utf-8" ?>
<request action="SET" version="1.0">
   <customer>
      <name>CSP</name>
      <user>
         <name>User1_CSP</name>
         <password>
            <password>Hello123</password>
            <changeatnextlogon>true</changeatnextlogon>
            <expires>NEVER</expires>
         </password>
      </user>
   </customer>
</request>
```

If no password is provided, then a password is generated.

Note that the first user provisioned to a customer will always become the customer administrator user. To specify security roles for a user, use the <roles> XML element within the <user> XML element.

Customers

Deprovisioning Customers

To deprovision a customer named "ZCSP," send the following request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>ZCSP</name>
    <status>NotProvisioned</status>
  </customer>
</request>
```

Re-provisioning Customers

To re-provision a customer named "ZCSP," set the status value to "Provisioned."

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>ZCSP</name>
    <status>Provisioned</status>
  </customer>
</request>
```

Disabling Customers

To disable a customer named "ZCSP," send the following request:

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>ZCSP</name>
    <enabled>false</enabled>
  </customer>
</request>
```

Enabling Customers

To re-enable a customer named "ZCSP," set the <enabled> value to **True**.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" trace="true" version="1.0">
  <customer>
    <name>ZCSP</name>
    <enabled>true</enabled>
  </customer>
</request>
```

Workflow Approval

This section describes how to configure workflow approval subscription for a customer. Sample requests are included to manage the "Workflow Managers" and "Workflow Groups" providers.

Get Approval Provider Subscriptions

Customer subscriptions can only be managed for enabled approval providers via the API.

A user must have the Update role permission to view and manage approval providers for a customer.

Use the following request to retrieve a list of approval providers for a customer with the “CSP” customer code.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
 <customer>
   <name>CSP</name>
   <approvalproviders />
 </customer>
</request>
```

Set Approval Provider Subscriptions

Use the GET request from the example above to create a SET request to manage approval providers. It is recommended to configure subscriptions in CPSM control panel to evaluate how the requests should be formatted via the API.

Only approval providers that are specified will be managed.

The configuration settings are optional and only applied if the tag is specified.

Subscriptions are optional and are only applied to the tags that are specified.

Set Workflow Managers

Use the following request to manage the Workflow Managers approval provider for a customer with the “CSP” customer code.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
 <customer>
   <name>CSP</name>
   <approvalproviders>
     <approvalprovider>
       <name>Workflow Managers</name>
       <configuration>
         <managers>5</managers>
         <timeoutdays>4</timeoutdays>
         <reminderdays>1</reminderdays>
       </configuration>
       <subscriptions>
         <users>
           <subscription>
             <name>Provision</name>
             <access>Object</access>
           </subscription>
           <subscription>
             <name>Deprovision</name>
             <access>Object</access>
           </subscription>
         </users>
         <userservices>
           <service>
             <label>Hosted Exchange</label>
             <subscription>
               <name>Provision</name>
               <access>None</access>
             </subscription>
             <subscription>
               <name>Deprovision</name>
               <access>None</access>
             </subscription>
           </service>
         </userservices>
       </subscriptions>
     </approvalprovider>
   </approvalproviders>
 </customer>
</request>
```

Set Workflow Groups

Use the following request to manage the Workflow Groups approval provider for a customer with the “CSP” short code.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
 <customer>
   <name>CSP</name>
   <approvalproviders>
     <approvalprovider>
       <name>Workflow Groups</name>
       <description>Allow groups of people to subscribe to events that require
approval</description>
       <groups>
         <group>
           <name>Customer Service Approval</name>
```

```
            <description>This group will approve all customer services</description>
            <configuration>
              <memberstoapprove>1</memberstoapprove>
              <timeoutdays>5</timeoutdays>
              <reminderdays>2</reminderdays>
            </configuration>
            <members>
              <member>
                <displayname>Jocelyn Brittain</displayname>
                <username>jocelynb_CSP</username>
              </member>
              <member>
                <displayname>Roger Alborough</displayname>
                <username>RogerA_CSP</username>
              </member>
            </members>
            <subscriptions>
              <services>
                <service>
                  <label>All</label>
                  <subscription>
                    <name>Provision</name>
                    <access>Object</access>
                  </subscription>
                  <subscription>
                    <name>Deprovision</name>
                    <access>Object</access>
                  </subscription>
                </service>
              </services>
            </subscriptions>
          </group>
        </groups>
      </approvalprovider>
    </approvalproviders>
  </customer>
</request>
```

For the request shown above, the Customer Service Approvals group is either created (if it does not exist) or updated (if it already exists).

Members will only be managed when the <members> tag is specified, so it is possible to update a group without changing its membership. The <members> tag must contain a list of all the members. Members that are not included in the <members> tag will be removed from the group. Only the <username> tag is required when a member is added and the <displayname> tag will be ignored. Deprovisioned users cannot be added as a member of a group.

Delete Workflow Groups

Use the following request to delete Group A for the "CSP" customer.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
 <customer>
   <name>CSP</name>
   <approvalproviders>
     <approvalprovider>
       <name>Workflow Groups</name>
       <description>Allow groups of people to subscribe to events that require
approval</description>
       <groups>
         <group>
           <name>Group A</name>
           <action>delete</action>
```

```
        </group>
      </groups>
    </approvalprovider>
  </approvalproviders>
 </customer>
</request>
```

Approval management

Workflow approval is triggered at the object level, so nothing is required via the API to generate an approval request when an object is provisioned or deprovisioned.

Standard functionality such as viewing pending requests that is available in CPSM control panel is also supported via the XML API. This section describes how to manage approval requests for the API.

Approval Status

The approval status of an object such as a customer, user, customer service, or user service is included automatically when a GET request is made.

All of the objects listed above have an existing provisioning “status” tag which will remain the same. A new read-only “approvalpending” boolean will also be included in the results to state whether the object is currently waiting on approval.

Here is an example status.

```
<status>Provisioned</status>
<approvalpending>False</approvalpending>
```

Get Customer Approval Requests

Use the following request to retrieve a list of approval requests for a customer with a “CSP” short code.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
  <customer>
    <name>CSP</name>
    <approvalresponses />
  </customer>
</request>
```

If a customer has users with pending approval responses, then the following approvalresponses tag will be returned.

```
<approvalresponses>
  <approvalresponse>
    <description>Provision User - Minnie Mouse</description>
    <status>Pending</status>
    <requestdate>20121001</requestdate>
    <resolveddate />
    <customer>
      <name>CSP</name>
      <fullname>Cortex Service Provider</fullname>
    </customer>
    <requestedbyuser>
      <name>Steve_CSP</name>
      <fullname>Steve McQueen</fullname>
      <email>Steve@csp.local</email>
    </requestedbyuser>
    <subscription>
      <id>b17edee6-50a0-4706-bcd8-29f67d2d5ac1</id>
      <requestdate>20121001</requestdate>
      <resolveddate />
      <status>Pending</status>
      <reason />
      <user>
```

```
        <name>Donald_CSP</name>
        <fullname>Donald Duck</fullname>
        <email>Donald@csp.local</email>
      </user>
    </subscription>
    <subscription>
      <id>b360825a-d7b1-4685-a1cc-ceb94b81fa4a</id>
      <requestdate>20121001</requestdate>
      <resolveddate />
      <status>Pending</status>
      <reason />
      <user>
        <name>Mickey_CSP</name>
        <fullname>Mickey Mouse</fullname>
        <email>Mickey@csp.local</email>
      </user>
    </subscription>
  </approvalresponse>
</approvalresponses>
```

Each approvalresponse tag represents an approval request. The approvalresponse tag contains information such as a description, status, and request date. The following sub tags are also included:

Sub tag	Description
Requestedbyuser	Contains information about the user who requested the change.
Subscription	Each Subscription tag contains an approval response that a user needs to approve. The user's information and response status is included. As well, a unique subscription identifier (GUID) is included that can be used for responding to a request or retrieving detailed request information.
Customer	Contains the customer information to state which organization the request belongs to. The customer information is useful when sub-customer requests have to be approved.

Only pending approvals are returned by default. Change the <approvalresponses /> filter to <approvalresponses filter="all" /> to include all approval requests for the customer.

Set Customer Approval Request

Use the following request to accept an approval request for a subscription with ID b360825a-d7b1-4685-a1cc-ceb94b81fa4a.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="SET" version="1.0">
 <customer>
   <name>CSP</name>
   <approvalresponses>
     <approvalresponse>
       <subscription>
         <id>b360825a-d7b1-4685-a1cc-ceb94b81fa4a</id>
         <status>Accepted</status>
         <reason>Go ahead and perform the update - from the API :)</reason>
       </subscription>
     </approvalresponse>
   </approvalresponses>
 </customer>
</request>
```

Get User Approval Requests

Use the following request to retrieve a list of pending approval requests for a user with a username "Mickey_CSP" and customer short code "CSP":

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
```

```
 <customer>
   <name>CSP</name>
   <user>
     <name>Mickey_CSP</name>
     <approvalresponses />
   </user>
 </customer>
</request>
```

Similar to the customer results, each approvalresponse tag represents an approval request that the user has subscribed.

```
<approvalresponses>
  <approvalresponse>
    <description>Provision User - Minnie Mouse</description>
    <status>Pending</status>
    <requestdate>20121001</requestdate>
    <resolveddate />
    <customer>
      <name>CSP</name>
      <fullname>Cortex service Provider</fullname>
    </customer>
    <requestedbyuser>
      <name>Steve_CSP</name>
      <fullname>Steve McQueen</fullname>
      <email>Steve@csp.local</email>
    </requestedbyuser>
    <subscription>
      <id>b360825a-d7b1-4685-a1cc-ceb94b81fa4a</id>
      <requestdate>20121001</requestdate>
      <resolveddate />
      <status>Pending</status>
      <reason />
      <user>
        <name>Mickey_CSP</name>
        <fullname>Mickey Mouse</fullname>
        <email>Mickey@csp.local</email>
      </user>
    </subscription>
  </approvalresponse>
</approvalresponses>
```

Note that only pending approvals will be returned by default. Change the <approvalresponses /> filter to <approvalresponses filter="all" /> to include all approval requests for the user.

Set User Approval Request

Use the following request to accept an approval request for a subscription with ID b360825a-d7b1-4685-a1cc-ceb94b81fa4a.

```
<request action="SET" version="1.0">
 <customer>
   <name>CSP</name>
   <user>
     <name>Mickey_CSP</name>
       <approvalresponses>
         <approvalresponse>
           <subscription>
             <id>b360825a-d7b1-4685-a1cc-ceb94b81fa4a</id>
             <status>Accepted</status>
             <reason>Mickey said it's okay - from the API :)</reason>
           </subscription>
       </approvalresponse>
     </approvalresponses>
   </user>
```

```
 </customer>
</request>
```

Get Approval Request Information

Use the following request to retrieve the XML document of the approval request for a subscription with ID b360825a-d7b1-4685-a1cc-ceb94b81fa4a.

```
<?xml version="1.0" encoding="utf-8"?>
<request action="GET" version="1.0">
 <approval>
   <id>b360825a-d7b1-4685-a1cc-ceb94b81fa4a</id>
 </approval>
</request>
```

Appendix A – API Exception Codes

API Error Code/ Exception	Description
0 Unknown	Unclassified API error – details in error message
1 CustomerError	Error in a customer level request – invalid attribute or setting.
2 TemplateNotFound	The template specified in the request was not found on the server
3 InvalidXmlFormat	This will be raised if the XML in the request is incorrectly formatted – e.g. missing tags.
4 InvalidObject	Deprecated
5 InvalidAction	This will be raised if the ActionName attribute is missing or is not a valid action (find, get, set, delete)
6 InvalidNewCustomer	Unable to create customer because either the name, fullname, contactname, contactemail and primarydomain was not specified.
7 InvalidUnlimited	A non-numeric property of type limit was specified and the value was non numeric with a value other than 'unlimited'.
8 InvalidNumericDelta	A non-numeric amount was entered in a delta amount for increasing/decreasing a quantity.
9 InvalidNumericType	A non-numeric amount was entered in a delta amount for increasing/decreasing a quantity
10 InvalidService	Relates to a problem related to the <service> tag – this can apply when within a customer or user node. Errors will be specific to the service.
11 NotAuthenticated	The user authentication failed
12 NotAuthorized	The current user (connecting to the API) does not have the required permissions within Cortex to execute the API request.
13 ReportError	An error occurred while processing an API report.
14 ReportNotFound	The report specified by the Name attribute was not found on the server.
15 InvalidUser	Error related to a User API request
16 UserNotFound	The user specified in the request does not exist in the customer specified.
17 InvalidStatus	Error will be raised if the <status> tag is not set to one of the following: Provisioned, Failed, Pending, Requested, InProgress, NotProvisioned
18 InvalidReference	When an XML node is passed by reference, this error is raised if the referenced node does not exist
19 InvalidProperty	Raised when an invalid property is specified in a request.
20 LocationNotFound	The <name> specified for an active directory location does not exist.
21 ServiceNotFound	The <service> specified does not exist or has not been configured.
22 RoleNotFound	The role specified does not exist
23 CitrixItemNotFound	The Citrix application or resource specified does not exist
24 CitrixItemInUse	The item could not be removed because it is still in use
25 CitrixCollectionNotFound	The application group specified does not exist
26 CitrixCollectionInvalid	The Citrix Farm name is missing or does not match the farm that is configured against the customer account
27 LocationError	The <LocationName> can not be deleted as it is in use
28 TokenNotFound	If the <name> specified for a token does not exist this error will be raised.
29 BrandTypeNotFound	The brand type was not specified or was not one of “DNS”, Parent or Custom
30 BrandNameNotFound	The brand name specified does not exist
31 HMCNotValid	If HMC was specified in a request and the location is not an HMC location.
32 InvalidBoolean	Returned If a non-Boolean value was used (something other than “true”, “yes”, “1”,”-1”,”0”, “no”
33 InvalidKey	If the “secure” attribute of the action is specified and the key provided cannot be decrypted
34 CustomerNotFound	The customer specified by <name> <id> <fullname> or <billingid> could not be located
35 PasswordNeverExpiredNotAllowed	Returned when a password is set to never expire when this is not enabled for the specified customer
36 InvalidAccountExpiration	The date specified for the invalid account expiration is not a valid format that can be cast to YYYY-MM-dd
37 InvalidPasswordExpiration	The date specified for the invalid account expiration is not a valid format that can be cast to YYYY-MM-dd

38	InvalidInteger	The number specified is not an integer

Appendix B – Service Names

The following table lists all the internal service names for the services that CloudPortal Services Manager supports. This value should be used when using the <Name> tag in the API requests

Service Display Name	Name
AD Sync	ADSync
BlackBerry 5	BlackBerry5
Citrix	Citrix
Customer Relationship Management 2011	CRM2011
DNS	DNS
File Sharing	FSS
Hosted Apps and Desktops	HostedAppsandDesktops
Hosted Exchange	HE
Lync 2010 for Hosters	LyncHosted
Lync Enterprise	LyncEnterprise
Mail Archiving	MARCH
Microsoft SQL Server Hosting	SQL2005Hosting
MySQL	MySQL
Office Communication Server 2007	OCS
Reseller	Reseller
SharePoint 2010	SharePoint 2010
Virtual Machine	VirtualMachine
Windows Web-Hosting	WinWebHost