Citrix® CloudPortal™ Services Manager v10.0 API User’s Guide

Document Version 1.0
Copyright and Trademarks

Use of the product documented herein is subject to your prior acceptance of the End User License Agreement. A printable copy of the End User License Agreement is included with your installation media.

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Citrix Systems, Inc.

© 2012 Citrix Systems, Inc. All rights reserved.

The following are trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries:
Citrix®, XenApp™, XenDesktop®, HDX™

All other trademarks and registered trademarks are the property of their respective owners.
<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Edits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Aug. 10, 2012</td>
<td>Initial v10.0 Release</td>
</tr>
</tbody>
</table>
Contents

Chapter 1 .......................................................................................................................... 1

Introduction ....................................................................................................................... 1
  Why use the API ............................................................................................................... 2
  Who should use the API ................................................................................................. 2
  How it works .................................................................................................................... 2
  CPSM Entities – Customers, Users and Services ............................................................ 2
    Provisioning ..................................................................................................................... 2
    Services .......................................................................................................................... 2
    Package Templates (Packages) ..................................................................................... 3
    User Plans ..................................................................................................................... 3

Chapter 2 .......................................................................................................................... 4

API Usage ......................................................................................................................... 4

Accessing the API ............................................................................................................ 5
  Where is the Citrix API Web service located? ............................................................... 5
  SSL ................................................................................................................................... 5
  Granting users access to the API ................................................................................... 5

Request Basics ................................................................................................................ 7
  Authentication .................................................................................................................. 7
  Request Structure .......................................................................................................... 7
  Request Actions .............................................................................................................. 7
  Response Status ............................................................................................................. 8
  Response Examples ....................................................................................................... 8
    “Set” / “Delete” Action Responses ............................................................................. 8
    “Find” Action Response .............................................................................................. 8
Basic API Customer Demo................................................................. 10
Find Customer(s) ........................................................................... 10
Create Customer........................................................................... 10
Request XML ................................................................................ 11
Response XML ............................................................................. 11
Provision/Find Users................................................................. 11
Provision Users ............................................................................. 11
Find Users .................................................................................. 11
Provision a Customer Service – Hosted Exchange ...................... 12
Provision User Service – Hosted Exchange .................................. 12
De-provision User/Customer Service and Customer itself .......... 13
De-provision User Service – Hosted Exchange............................ 13
De-provision Customer Service – Hosted Exchange .................. 13
De-provision Customer ................................................................ 13
Delete User/Customer .............................................................. 13
Delete User .................................................................................. 14
Delete Customer ........................................................................... 14

Making an API request.................................................................. 15
PowerShell Example ..................................................................... 15

Useful Techniques ........................................................................ 17
Multi-Item Provision ..................................................................... 17
Multi-User Provision .................................................................... 17
Multi-Userplan (userpackage) Provision ...................................... 17
Create Customer and provision HE service ................................. 18
Get Customer Information ............................................................ 18
Using the Get command to view hidden properties ...................... 18
Request ....................................................................................... 18
Response ..................................................................................... 19
Set property ................................................................................ 19

Advanced Techniques .................................................................. 20
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get all User plans and Packages</td>
<td>20</td>
</tr>
<tr>
<td>Get all Instances for Instance-able Services</td>
<td>20</td>
</tr>
<tr>
<td>Find all Services Provisioned to Customer</td>
<td>20</td>
</tr>
<tr>
<td>Find all Services a Reseller can Resell</td>
<td>20</td>
</tr>
<tr>
<td>Producing Reports via the API</td>
<td>21</td>
</tr>
<tr>
<td>Find Reports</td>
<td>21</td>
</tr>
<tr>
<td>Get Reports</td>
<td>21</td>
</tr>
</tbody>
</table>

**Appendix A – API Exception Codes** .......................................................... 24

**Appendix B – Customer Properties** ........................................................... 26

**Appendix C – Service Names** .................................................................... 28
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 the 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) format requests.
Why use the API

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

An example of this is interfacing CPSM with a billing application to automate the billing every month, further automating what might be a manual process.

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

The API can be useful in migration tasks – correctly formatted data can be 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 be able to leverage the power of the API.

The API test tool will allow you to construct XML requests and run them without having to complete 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 Provisioning Status can be one of:

<table>
<thead>
<tr>
<th>Status</th>
<th>CPSM UI Indicator (colour)</th>
<th>CPSM API Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Not provisioned&quot;</td>
<td>Grey</td>
<td>NotProvisioned</td>
</tr>
<tr>
<td>&quot;Provisioning requested&quot;</td>
<td>Yellow</td>
<td>Requested</td>
</tr>
<tr>
<td>&quot;Provisioning in progress&quot;</td>
<td>Orange</td>
<td>InProgress</td>
</tr>
<tr>
<td>&quot;Provisioned&quot;</td>
<td>Green</td>
<td>Provisioned</td>
</tr>
<tr>
<td>&quot;Provisioning failed&quot;</td>
<td>Red</td>
<td>Failed</td>
</tr>
<tr>
<td>&quot;Pending Changes&quot;</td>
<td>Blue</td>
<td>Pending</td>
</tr>
</tbody>
</table>

Services

Services can be provisioned to a Customer. These are called Customer Services. Once 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 Microsoft Exchange Service to a Customer you will need to set whether to create a public folder, and also the domain names for which mail will be accepted. When provisioning the Microsoft Exchange Service to a User, you would need to set the mail aliases for the User.

**Package Templates (Packages)**

Package Templates group together common Customer Service properties. Some Customer Services don’t have Package Templates. For those that do, when a Customer is provisioned with a Service you can only select one Package Template. A Package Template may also be called a Customer Plan.

**User Plans**

User Plans group together common User Service properties. Some User Services don’t have a User Plan. User Plans are also referred to as User Packages when dealing with Hosted Exchange.

When provisioning a Customer Service, you can specify which User Plans are going 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 could allow 10 users to be created with the ‘Gold’ User Plan, and 100 users with the ‘Silver’ User Plan.

In version 9.1/9.2, User Plans were referred to as Service Access Levels.
Chapter 2
API Usage

Topics:

- Accessing the API
- Request Basics
- Basic API Customer Demo
- Making an API request
- Useful Techniques
- Advanced Techniques
- Appendix A – API Exception Codes
- Appendix B – Customer Properties
- Appendix C – Service Names

This chapter will discuss 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 Webserver. If you are a reseller accessing the CPSM API web service you will need to request the CPSM API URL from your service provider.

For the service provider:

- Connect to the CPSM Webserver and open Internet Information Services (IIS) Manager in your Administrator Tools or via Start > Run > inetmgr.exe
- Expand the Sites folder, Cortex Management and then CortexAPI.
- Browse to the Default.aspx page.
- You will be prompted for a username and password.
- Input your Citrix Username and Password.
- If the API is working correctly you will see the following:

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

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

The URL in the Internet Explorer (IE) page should be similar to the following:

https://{CPSM Site Domain}/CortexAPI/Default.aspx

E.g. https://www.cspmdemo.com/cortexapi might be the external URL that you provide to your clients so they can use the API.

SSL

We strongly recommend that this API is only ever accessed over SSL if the API is being accessed over the Internet.

There is nothing to stop the API from working without SSL, it is your responsibility to configure and secure the CPSM Web application site.

Granting users access to the API

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

We recommend making a new role that grants a user access to the API. This allows you to keep the current CPSM roles you have provided a user. You can create a new role by

- Navigating to Configuration → Security → Security Roles
  - Click “New Role” and name the Role “API Access”
  - Within the “Role Permissions” section click on the “Customers” tab
  - Click the API Access to choose the level of access
  - Click “Save”

Next you must provision that role to the customer, in order for that customer to provide the role to a specific user. Find the customer you want to provide the role to.
• **Navigate to Customer** \(\rightarrow\) **“your customer”** \(\rightarrow\) **Edit Customer**
  - **Advanced Properties** \(\rightarrow\) **Allowed Roles** \(\rightarrow\) **Check the “API Access” role that you’ve created**
  - **Click “Provision” for the customer**
    - **This provides the customer the ability to assign the role to a user or set of users**
  - **Login as the customer admin, or go to the customers “Users” list**
  - **Pick a user and click “Edit User”** \(\rightarrow\) **Account Settings** \(\rightarrow\) **Advanced Options**
  - **Check the “API Access” role for the user and the click “provision” for the user**

You will need to ensure that you provide the user with sufficient privileges that allows them to fully carry out the tasks they require. If they can only access their current customer’s users they will not be able to create any sub customers, so granting them the Reseller Administrator 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 identity must be in a CPSM Role that has both administrative permissions and the “API Access” permission.

Request Structure

The following XML defines the request structure:

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

The action attribute is mandatory and defines what operation the request performs. It can be one of 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 typically 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. It can be one of customer, location or template. The customer is the most common entity and it contains service and user entities.

Request Actions

There are four actions that you can perform when passing in information to the API. These actions will dictate how the information will be handled and what responses you will get.

**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 will be returned in the response.

Users can be searched for by using these filters:

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

User Service can be searched for by using these filters:

- Name
- Fullname

**GET Action**

The GET action retrieves details for the entities you request.
**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.

**DELETE Action**

The DELETE action uses the same format as the GET request, only it deletes the entity. For example, if you specify a customer, then that customer is deleted. The DELETE action can also be applied to users. Services cannot be deleted as being de-provisioned removes all rights from the user or customer, so they are already removed from the entity.

**Response Status**

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

If there was a problem processing the request, the response will have an error tag with a numeric id and a textual message explaining the reason for the error. It will have a status of one of the following:

- **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 responses types, Set / Delete, Find, Get

**“Set” / “Delete” Action Responses**

A successful Set or Delete command will result in the following.

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

If you see the response below 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>
```

**“Find” Action Response**

The Find action will return data according to the parameters you specified.

For example if you specify a service that does not exist, it will still return the customer information, but not the service details.

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

If it only returns the following, it means that it did not find the requested customer information. The 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 will return all information it can about the entities you specified in the request.

If all the specified entities could not be found, then it will return with an error message stating what could not be found.

Request Layout
The layout of requests follows a similar pattern for every action.

By keeping the format, you will be able to choose the correct syntax to produce the results to achieve your requirements.

Request:

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

The attribute “action” on the “request” node will determine the action executed. Since we want detailed information, we set it to Get. The node “name” in the “customer” node expects the customer short name.
Basic API Customer Demo

The best way to learn how to effectively leverage the API is to first test on a lab environment, so you can use trial and error.

For most instances we provide the minimum parameter requirements for a provision to occur, we sometimes 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)

First step is to check if the customer name you want to use is already taken by another customer. For this, use the “find” command.

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

The attribute “action” on the “request” node is set to Find. Since we want to find if the customer “fullname” exists, we put it inside a “customer” node. If the customer fullname doesn’t exist you will get the following response.

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

You will see the above response often in the API, as it represents either no information was returned in a Find request or a successful transaction for a Set or Delete request.

However if a customer with the fullname of “Basic API Customer” does exist the response will appear as follows.

```xml
<?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 the above response, it returned information contained within a “customer” node. All nodes 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’s to use a new customer name, or you could de-provision and delete it first.

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

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

Create Customer

When you create a customer in the CPSM UI there are required fields, these fields are also a requirement for the CPSM API.

(Full Name, Contact Name, Email Address, Domain)
Request XML

```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 attribute “action” on the “request” node is set to SET as we want to provision a customer. All nodes in between the Customer tags are the specified requirements for creating a customer as listed above. The “parent” node specifies the “fullname” of the reseller this customer will be created under. This is not required but recommended if you are dealing with multiple tiers of resellers within the CPSM Hierarchy.

Response XML

An empty response will be returned for a successful request.

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

If you leave one of the required fields an error similar to the one below will be returned.

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

The reason why we don’t have to specify a `<name>` node is due to CPSM automatically generating the short code (short name) for the customer. If you want a specific code, you may specify it as well.

Provision/Find Users

Provision Users

```xml
<?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 attribute “action” on the “request” node is SET to set as we want to create a user. The node “name” in the “customer” node is for the customer short name where you want to provision the user to. In the “user” node we specify the “name” of the user. The user’s upn, firstname, lastname and externalemail will default to your environment settings.

Find Users

If you want to Find all the users for a customer, use this request:
Provision a Customer Service – Hosted Exchange

The attribute “action” on the “request” node is set to SET as we want to provision Hosted Exchange to a Customer. The node “name” in the “customer” node expects the customer short name. The “service” node specifies that we want to alter details about a service. (In our case, Provision it) For this we need to specify the “name” of the service. Also, for HE, specify the “version” of exchange users will receive.

The “userpackage” node is the User Plan that users can be provisioned to. For backward compatibility reasons HE is the only service where this node is named “userpackage”; in all other services, the node is named “userplan”.

It must include an “enabled” node set to true or else it will be disabled by default. The “package” node is the package the customer will be provisioned with. Commonly, packages are called “Basic” and “Public Folders”.

Provision User Service – Hosted Exchange

You will notice that the syntax to provision a user with a service is very similar to provisioning a customer with a service.
De-provision User/Customer Service and 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 User Service – Hosted Exchange

```xml
<?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 attribute “action” on the “request” node is set to SET as we want to alter the status of a service to the user. We add the “status” node as it specifies what we want to set the status to. The status must be one of: Provisioned, Failed, Pending, Requested, InProgress, NotProvisioned.

De-provision Customer Service – Hosted Exchange

```xml
<?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 will in turn de-provision all users with the HE service. The “status” node is contained within the “service” node.

De-provision Customer

```xml
<?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” node is contained within the “customer” node.

Delete User/Customer

**WARNING:** Please take care when using the Delete action as unlike the CPSM UI, you may specify a Delete action to a User or Customer that is yet to be de-provisioned, this will result in the User or Customer losing their information if you make a mistake.
Delete User

```xml
<?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 attribute “action” on the “request” node is set to “DELETE” as we want delete the user. As you can see, you require all the same details as if you are going to de-provision the user, except you do not need to specify the “status” node.

**Error Message**

If you have been following the Document exactly, you will get an error message.

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

CPSM will not allow you to delete the last user in a customer as we always require at least one to exist on a customer. If you would like to test this Delete User script, create another user.

Delete Customer

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

The attribute “action” on the “request” node is set to DELETE as we want delete the customer. As you can see, the only node required is the customer “name” node.
Making an API request

PowerShell Example

Here’s a sample of PowerShell script that uses the .Net WebClient to make 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://10.8.168.96/cortexapi/default.aspx"

# Initialize the client
$client.Credentials = New-Object System.Net.NetworkCredential("cspadmin@csp.local","<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 /></customer><primarydomain>csp.local</primarydomain><status>Provisioned</status></response>
```
Useful Techniques

Multi-Item Provision

Instead of sending the same request multiple times, to provision users, you can chain the items into one request, and 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 2 users. By closing the “user” node and opening it, both users will be provisioned under customer ‘BAC’

Multi-Userplan (userpackage) Provision

This 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 Customer and provision HE service

```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>
    <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. By using this on a customer, this will return all the information you require in a format that you can cut and paste into a Set action and use again. All that is required is for you to alter the "response" node to "request" and put it as action='SET' and any other settings you want different.

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

Using the Get command to view hidden properties

Currently all properties used in the default CPSM interface are exposed, but there may be other properties associated with an entity that are hidden by default. These are typically more advanced settings that are infrequently changed. To view any properties that are hidden you will need to use the "properties" node.

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

As you can see from the response size, it contains many different properties that you would normally not see. The "<properties />" node can be added under a customer, user, userplan, or package node as well as the service node.
**Response**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<response version="1.0">
  <customer>
    <service>
      <name>HE</name>
      <properties>
        <property>
          <name>MailBoxAllocated</name>
          <fullname>MailBox Allocated in Mb</fullname>
          <value>1300</value>
          <access>Modify</access>
          <category>Customer</category>
        </property>
      </properties>
    </service>
  </customer>
</response>
```

This is a snippet of the response with it showing the MailboxAllocated value for the customer HE service. The value is set to 1300 Megabytes.

To see even more properties, change `<properties />` to `<properties filter="all" />`.

**Set property**

You can set these values using the API by following the same syntax from the response.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<request action='SET' version='1.0'>
  <customer>
    <name>BAC</name>
    <service>
      <name>HE</name>
      <properties>
        <property>
          <name>MailBoxAllocated</name>
          <value>1400</value>
        </property>
      </properties>
    </service>
  </customer>
</request>
```

Here we specify the "value" node as 1400 MB for the "MailBoxAllocated" property.
Advanced Techniques

The following sections show how to Get and Set 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
<?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 Instance-able Services

If you wish to see information about the instances of one of the four instance-able services (CRM2011, SharePoint2010, SQLHosting, or WinWebHost), use this request:

```xml
<?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>
```

Find all Services Provisioned to Customer

If you wish to see some information about all the services a customer was provisioned, use this request:

```xml
<?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

If you wish to see all the services a reseller can resell, use this request:

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

The CPSM API can be used to run simple report queries on the CPSM database and displaying the results in XML or tab delimited (TSV). There is some configuration required in the CPSM database.

- Log on to the CPSM web server and open the web.config file for the API. This is located under:
  ```bash
  %Program Files (x86)%/Citrix/Cortex/CortexWeb/CortexAPI/
  ```
- Copy the connection string value that has been populated for key “APIConnectionString”
- Log on to the CPSM SQL server and the OLM database, Open table dbo.ReportQueries and update all reports with the connection string that you have copied from the web.config. Dbo.ReportsQueries defines the report queries that can be executed by the CPSM API. Additional reports can be added to this table.

**WARNING**: If you add your own ReportQueries they can potentially expose information about customers that the API caller would not normally have permissions to view. To help with this situation the customer ID of the API caller is automatically added to GET report query parameters. The query or stored procedure should use this ID to limit the results returned. See the example 'Customers with Query' for an example 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 callers customer:

```sql
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 will return all reports that are configured in the ReportQueries table. For each report, a list of parameter names will be displayed, these can be used to specify the context of the report that will be produced.

```xml
<request action="FIND" version="1.0">
  <report />
</request>
```

Get Reports

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

The following nodes can be used:

- `<name>`  
  Name of the Report
- `<parameter name="[ParameterName]">`  
  The report’s parameter name. A value should be entered for the parameter node
- `<timeout>`  
  Sets a timeout period in milliseconds. The value 0 will wait forever
- `<column>`  
  Using the column node will only return the data for the specified column

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

- `<dataformat columnnames="[True]">xml</dataformat>`  
  Report will be generated in xml. If columnnames is set to true, the Column Name will appear as the returned nodes
- `<dataformat header="[True]">tsv</dataformat>`  
  Report will be generated in TSV. If header is set to true, the column names will appear at the top of the report

The following API query will return all customers where the customer Name begins with “C”. The report will be in xml.format with the column names appearing in the returned nodes.
The API returns the data as:

```xml
<?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 will return the details for customer code “CSP”. The report will be in tsv format with the column names appearing. The timeout for the report query is set to wait forever.

```xml
<?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
<?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[
    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>
# Appendix A – API Exception Codes

<table>
<thead>
<tr>
<th>API Error Code/ Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
</tr>
<tr>
<td>1</td>
<td>CustomerError</td>
</tr>
<tr>
<td>2</td>
<td>TemplateNotFound</td>
</tr>
<tr>
<td>3</td>
<td>InvalidXmlFormat</td>
</tr>
<tr>
<td>4</td>
<td>InvalidObject</td>
</tr>
<tr>
<td>5</td>
<td>InvalidAction</td>
</tr>
<tr>
<td>6</td>
<td>InvalidNewCustomer</td>
</tr>
<tr>
<td>7</td>
<td>InvalidUnlimited</td>
</tr>
<tr>
<td>8</td>
<td>InvalidNumericDelta</td>
</tr>
<tr>
<td>9</td>
<td>InvalidnumericType</td>
</tr>
<tr>
<td>10</td>
<td>NotAuthenticated</td>
</tr>
<tr>
<td>11</td>
<td>NotAuthorized</td>
</tr>
<tr>
<td>12</td>
<td>ReportError</td>
</tr>
<tr>
<td>13</td>
<td>ReportNotFound</td>
</tr>
<tr>
<td>14</td>
<td>InvalidUser</td>
</tr>
<tr>
<td>15</td>
<td>UserNotFound</td>
</tr>
<tr>
<td>16</td>
<td>InvalidStatus</td>
</tr>
<tr>
<td>17</td>
<td>InvalidReference</td>
</tr>
<tr>
<td>18</td>
<td>InvalidProperty</td>
</tr>
<tr>
<td>19</td>
<td>LocationNotFound</td>
</tr>
<tr>
<td>20</td>
<td>ServiceNotFound</td>
</tr>
<tr>
<td>21</td>
<td>RoleNotFound</td>
</tr>
<tr>
<td>22</td>
<td>CitrixItemNotFound</td>
</tr>
<tr>
<td>23</td>
<td>CitrixItemInUse</td>
</tr>
<tr>
<td>24</td>
<td>CitrixCollectionNotFound</td>
</tr>
<tr>
<td>25</td>
<td>CitrixCollectionInvalid</td>
</tr>
<tr>
<td>26</td>
<td>LocationError</td>
</tr>
<tr>
<td>27</td>
<td>TokenNotFound</td>
</tr>
<tr>
<td>28</td>
<td>BrandTypeNotFound</td>
</tr>
<tr>
<td>29</td>
<td>BrandNameNotFound</td>
</tr>
<tr>
<td>30</td>
<td>HMCNotValid</td>
</tr>
<tr>
<td>31</td>
<td>InvalidBoolean</td>
</tr>
<tr>
<td>32</td>
<td>InvalidKey</td>
</tr>
<tr>
<td>33</td>
<td>CustomerNotFound</td>
</tr>
<tr>
<td>34</td>
<td>PasswordNeverExpiredNotAllowed</td>
</tr>
<tr>
<td>35</td>
<td>InvalidAccountExpiration</td>
</tr>
<tr>
<td>36</td>
<td>InvalidPasswordExpiration</td>
</tr>
<tr>
<td>API Error Code/ Exception</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>0</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Uncategorized error</td>
</tr>
</tbody>
</table>
# Appendix B – Customer Properties

The following table lists all the default customer properties within the `<properties>` node. These properties are all alterable.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value Type</th>
<th>Description</th>
<th>Read / Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>HeadOffice</td>
<td>text</td>
<td>Can be used to save the name and address of the Head Office</td>
<td>Read / Write</td>
</tr>
<tr>
<td>isPrepay</td>
<td>0 or 1</td>
<td>Determines whether the customer pays for its services a) in advance (defined units) or b) at the end of the month (actual usage) * The property can be changed from a value 1 to a value of 0. Once the value is set to 0, it cannot be changed</td>
<td>Read / Write</td>
</tr>
<tr>
<td>CurrencySymbol</td>
<td>any text</td>
<td>Configures the currency symbol that is applied customer account. The default is $</td>
<td>Read / Write</td>
</tr>
<tr>
<td>LanguageCode</td>
<td>1 for English</td>
<td>The default language that will be configured to all new users</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxExpired</td>
<td>True or False</td>
<td>Configuration setting for Email Expiry Report. Filters on accounts where the password has expired</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxNeverExpire</td>
<td>True or False</td>
<td>Configuration setting for Email Expiry Report. Filters on accounts that never expire</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxDisabled</td>
<td>True or False</td>
<td>Configuration setting for Email Expiry Report. Filters on accounts that are disabled</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxEmail</td>
<td>True or False</td>
<td>Enable the Email Expiry Report to be sent to the customer’s users. Settings for this report are configured under Users &gt; Configuration &gt; Email Configuration</td>
<td>Read / Write</td>
</tr>
<tr>
<td>CPNL</td>
<td>0 or 1</td>
<td>All new users will be required to change their password at next logon</td>
<td>Read / Write</td>
</tr>
<tr>
<td>APwdNEExpire</td>
<td>0 or 1</td>
<td>Provide the ability for Customer Administrators to set their users’ passwords to never expire</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxAboutToExpire</td>
<td>True or False</td>
<td>Configuration setting for Email Expiry Report and Email Notification Report to include users that are about to expire. If this is set to True, property “pwxDays” should be set with a numeric value</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxDays</td>
<td>Numeric</td>
<td>Configuration setting for Email Expiry Report and Email Notification Report. Includes users where passwords will expire in x days</td>
<td>Read / Write</td>
</tr>
<tr>
<td>OrganisationalStructure</td>
<td>ungrouped department location department location_department location_department user_specified</td>
<td>Specifies the structure on how new users will be saved in the Customer’s OU in Active Directory</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxSendNotificationReport</td>
<td>True or False</td>
<td>Enable the Email Notification Report to be sent to the Customer Administrator. Settings for this report are configured under Users &gt; Configuration &gt; Email Configuration</td>
<td>Read / Write</td>
</tr>
<tr>
<td>pwxLocked</td>
<td>True or False</td>
<td>Configuration setting for the Email Notification Report. Includes user accounts that are locked</td>
<td>Read / Write</td>
</tr>
<tr>
<td>DisabledUsers</td>
<td>Text</td>
<td>This is an internal system property and will list all users that were in a disabled state before a customer is disabled. When the customer is reactivated, these users will remain in an disabled state</td>
<td>Read</td>
</tr>
<tr>
<td>Property Name</td>
<td>Value Type</td>
<td>Description</td>
<td>Read / Write</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Enabled</td>
<td>True or False</td>
<td>Internal system property to determine if the customer is enabled or disabled</td>
<td>Read</td>
</tr>
<tr>
<td>isDemoCustomer</td>
<td>0 or 1</td>
<td>Determines if the customer is a trial customer account. * The property can be changed from a value 1 to a value of 0. Once the value is set to 0, it cannot be changed</td>
<td>Read / Write</td>
</tr>
<tr>
<td>DemoExpiryDate</td>
<td>text</td>
<td>The expiry date for the trial customer account. The customer account will automatically change to a deprovisioned state Mandatory if property “isDemoCustomer” is set to 1</td>
<td>Read / Write</td>
</tr>
<tr>
<td>DemoStartDate</td>
<td>text</td>
<td>The start date of the trial customer account. Mandatory if property “isDemoCustomer” is set to 1</td>
<td>Read / Write</td>
</tr>
<tr>
<td>UserDisplayNamePattern</td>
<td>text</td>
<td>The pattern that is used to generate a new user’s display name. The text value must equal a value that is specified in the Validation Control “UserDisplayPatterns”. The display name should not be entered.</td>
<td>Read / Write</td>
</tr>
<tr>
<td>UserDefaultEmailPattern</td>
<td>text</td>
<td>The pattern that is used to generate a new user’s default email address. The text value must equal a value that is specified in the Validation Control “UserDefaultEmailPatterns”. The display name should not be entered.</td>
<td>Read / Write</td>
</tr>
<tr>
<td>CustomerFullPath</td>
<td>text</td>
<td>Displays the OU path for the customer OU</td>
<td>Read</td>
</tr>
</tbody>
</table>
## Appendix C – 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>` node in the API requests.

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