vCenter Chargeback API Programming Guide

vCenter Chargeback 1.0.1

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.

EN-000296-00

vmware®

You can find the most up-to-date technical documentation on the VMware Web site at:

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book

The *vCenter Chargeback API Programming Guide* provides information on how to use vCenter Chargeback APIs.

Intended Audience

This book is intended for anyone who develop applications to work with vCenter Chargeback.

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to docfeedback@vmware.com.

vCenter Chargeback Documentation

The vCenter Chargeback documentation comprises the following guides:

- vCenter Chargeback User's Guide
- vCenter Chargeback API Programming Guide
- vCenter Chargeback API Reference Guide

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current version of this book and other books, go to http://www.vmware.com/support/pubs.

Online and Telephone Support

To use online support to submit technical support requests, view your product and contract information, and register your products, go to http://www.vmware.com/support.

Customers with appropriate support contracts should use telephone support for the fastest response on priority 1 issues. Go to http://www.vmware.com/support/phone_support.

Support Offerings

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available onsite, in the classroom, and live online. For onsite pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to http://www.vmware.com/services.

vCenter Chargeback APIs

1

This chapter includes the following topics:

What Is vCenter Chargeback?

vCenter Chargeback is an end-to-end cost reporting solution for environments virtualized using vSphere. This Web-based application interacts with the vCenter Database to retrieve usage information, calculates the cost by using the defined Chargeback formulas, and generates reports.

vCenter Chargeback runs on an Apache Tomcat server instance. The user interacts with the vCenter Chargeback application through a load balancer (Apache HTTP Server). vCenter Chargeback connects to a Chargeback database that stores information such as the defined hierarchies, cost model, users, roles, and so on. The application also interacts with the vCenter Server and vCenter Database through a Data Collector. The Data Collector uses VIM APIs to communicate with the vCenter Server and JDBC to communicate with the vCenter Database.

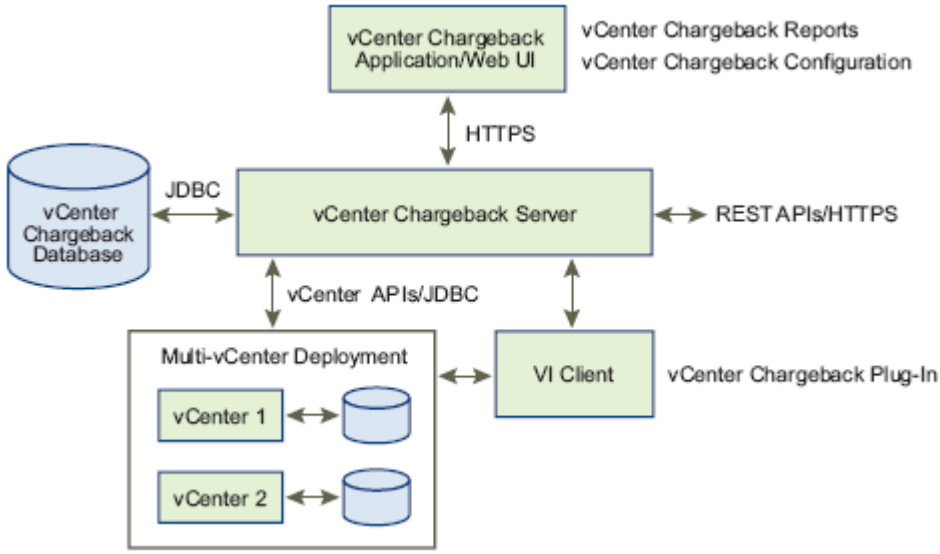
vCenter Chargeback APIs provide an interface for application developers to programmatically use the functionality of vCenter Chargeback.

For more information about Chargeback and its capabilities, see the *vCenter Chargeback User's Guide*.

REST Architecture

vCenter Chargeback APIs implement the Representational State Transfer (REST) architecture. REST-based APIs typically help you send HTTP requests for resources over the network and receive responses.

Figure 1-1. REST in vCenter Chargeback Architecture



Requests

An HTTP request sent by a Chargeback API can be one of the following: PUT, POST, GET, or DELETE. Each of these request types maps to a standard CRUD operation as shown in the following table.

Table 1-1. Request Type Mapping

Request Type	CRUD Operation
POST	CREATE
GET	READ
PUT	UPDATE/CREATE
DELETE	DELETE

Responses

When an API task is successful, the value of the `status` field in the response XML is set to `success` as shown in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response status="success" isValidLicense="true"
               xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Hierarchies>
        <Hierarchy id="1">
            <Name>Test_Hierarchy_Renamed</Name>
            <Description>Test_Hierarchy_Renamed
            </Description>
            <InSync>true</InSync>
            <LastUpdatedTime>1251893303708</LastUpdatedTime>
            <LastUpdatedUser>CBM Server</LastUpdatedUser>
            <Entities />
        </Hierarchy>
    </Hierarchies>
</Response>
```

If an API task is unsuccessful, the `status` field is set to `failure` and the Error element captures all the details as explained in the table provided after this code sample.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response status="failure" isValidLicense="true"
               xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Error majorErrorCode="500" minorErrorCode="2014"
message="Hierarchy with id '1' does not exist." />
</Response>
```

Table 1-2. Components of Error Elements

Element	Description
majorErrorCode	The class of the error. It represents the HTTP Status codes.
minorErrorCode	The specific API error code. For example, it can indicate that hierarchy creation failed.
vendorSpecificErrorCode (Optional)	A vendor/implementation specific error code that points to specific modules/parts of the code and can make problem diagnostics easier. For example, it can indicate if a code snippet is a vCenter error code or a Database error code.
message	A one-line, human-readable message that describes the error that occurred.
ErrorStackTrace	This element is present when the log level in Chargeback is set to `Debug`.

Common Elements in the Request and Response XMLs

The common XML elements used by the request and response XMLs are presented in the following table. Refer to the earlier sections for sample request and response XMLs.

Table 1-3. Common Elements

Element	Description
Request	An API request starts with this element.
Response	An API response starts with this element.
status	Denotes whether API is successfully executed.
Is Valid License	Indicates the status of the license. Value can be true or false.

Chargeback API Syntax

Each vCenter Chargeback API has this syntax:

```
<HTTP_request_method> <Base_Url>/<API_signature>?version=1.0.1
```

For example, `GET https://123.123.123.123/vCenter-CB/api/hierarchies?version=1.0.1`

Table 1-4. API Syntax Components

Syntax Component	Description
HTTP_request_method	PUT, POST, GET, or DELETE
Base_Url,	The URL of the Chargeback host. The base URL for vCenter Chargeback APIs is: `https://<Chargeback server IP>/vCenter-CB/api` For example, `https://123.123.123.123/vCenter-CB/api`
API_signature	The URL path for a Chargeback API. For example, `/hierarchies` retrieves the hierarchies added to the Chargeback server running on 123.123.123.123.
version	API version. For Chargeback server 1.0.1, it needs to be passed as 1.0.1.

API Versioning

Every vCenter Chargeback 1.0.1 API request/response includes targetnamespace to denote its version. For example, the following API request sends the version with which it is working.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <Hierarchies>
        <Hierarchy>
            <Name>Test_Hierarchy</Name>
            <Description>Test Hierarchy</Description>
        </Hierarchy>
    </Hierarchies>
</Request>
```

The request states that, vCenter Chargeback API version 1.0.1 needs to be called. If the request is for vCenter Chargeback server version 1.0, this call fails. If the request is for vCenter Chargeback server 1.0.1, the call succeeds and gets a response as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response status="success" isValidLicense="true"
               xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Hierarchies>
        <Hierarchy id="5">
            <Name>Test_Hierarchy</Name>
            <Description>Test Hierarchy</Description>
            <InSync>true</InSync>
            <LastUpdatedTime>1251892244029</LastUpdatedTime>
            <LastUpdatedUser>local\admin</LastUpdatedUser>
            <Entities>
                <Entity id="119">
                    <Name>Test_Hierarchy</Name>
                    <Description>Root</Description>
                    <Type>100</Type>
                    <Shares>
                        <Share>
                            <Percentage>100</Percentage>
                            <Parent id="-1" />
                        </Share>
                    </Shares>
                </Entity>
            </Entities>
        </Hierarchy>
    </Hierarchies>
</Response>
```

This response indicates that vCenter Chargeback API 1.0.1 is called. All the APIs starting from version 1.0.1 will be backward compatible. That is, a request for vCenter Chargeback API 1.0.1 to vCenter Chargeback 1.2 server, will be valid.

The API version, if being specified, should be present in both the request XML (wherever applicable) and in the URL. If it is specified only in the request XML or the URL and not both, then the API displays an error. If the API version is absent in both the URL and the request XML (wherever applicable), then the API returns the output corresponding to the latest version.

URI Versioning

In the URI of an API, you can pass an optional parameter to indicate the output version you expect as shown in the following example:

```
HTTP_request_method> <Base_Url>/<API_signature>?[version=1.0.1]
```

If you do not provide the version, the API returns the output corresponding to the latest version.

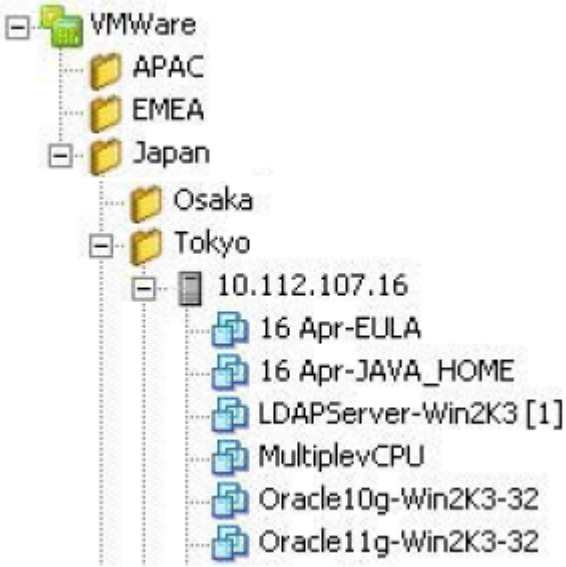
Understanding the Workflow 2

This chapter presents a case study of a fictitious company to explain how to perform some of the basic vCenter Chargeback tasks using the APIs. The chapter includes the following topics:

Introduction to the Case Study

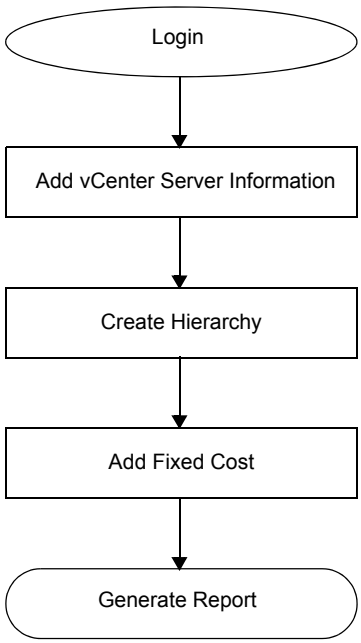
ABC has a presence across geographic regions such as Asia Pacific (APAC), Europe, the Middle East and Africa (EMEA), and Japan. In each of these regions, their offices are located in multiple cities. For example, in Japan, among other cities, ABC also has an office in Tokyo. This office has a number of ESX hosts that run many virtual machines. The company wants to create a hierarchy of resources and measure the cost of resources for each office and for each region as represented in Figure 2-1.

Figure 2-1. Sample Hierarchy for ABC



To accomplish the objective stated earlier, the vCenter Chargeback Administrator needs to do the following tasks. The sections that follow this diagram explain how to perform these tasks using the vCenter Chargeback APIs.

Figure 2-2. Task Flow



Login to vCenter Chargeback Server

Use the Login API to login to vCenter Chargeback server.

- The syntax for this API is `<HTTP_request_method> <Base_URL>/login`. For example, you can define a call as follows:

  `POST https://123.123.123.123/vCenter-CB/api/login`.

- Use the URL parameter **`version`** to specify the API version to be called.
- This API takes a request XML in which you can specify the following login details:
  - LDAP server ID
  - User type
  - User name
  - Password
- The following is a sample request XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <Users>
        <User>
        <!-- <LdapServerId>1</LdapServerId> -->
            <Type>local</Type>
            <Name>TestUser</Name>
            <Password>SomePassword</Password>
        </User>
    </Users>
</Request>
```

- The following is a sample program that calls the Login API. This program assumes that the request XML is populated with the required information.

```
/**
* This method is for Login to vCenter-ChargeBack
*
* @param requestFilePath
* @param baseURL
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleLoginMethod(String requestFilePath, String baseURL, String
               clientVersion) throws IOException, JDOMException,
NoSuchAlgorithmException, KeyManagementException, HttpException {
PostMethod post = null;
// String uri=baseURL+"/login";

NameValuePair[] parameters = {new NameValuePair("version", clientVersion)};

Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https", (ProtocolSocketFactory) new
               FakeSSLCertificateSocketFactory(), 443));
String uri = "https://" + baseURL + "/vCenter-CB/api/login";

try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
        }
finally {
    if (post != null) {
    post.releaseConnection();
    }
}
}
}
```

- If the login succeeds, the API returns a response XML that indicates the status of the operation.

Add vCenter Server Information

Use the Add vCenter Server API to add vCenter servers instances in your virtualized environment to vCenter Chargeback. This helps determine the computing resources utilization for the virtual machines and calculate the total costs.

- The URL for this API is `<HTTP_request_method> <Base_URL>/vCenterServer`. You can define a sample call as follows:

  `POST https://123.123.123.123/vCenter-CB/api/vCenterServer`.

- The Add vCenter Server API takes a request XML that captures the following information:
  - vCenter Server Hostname/IP: FQDN or IP address of the vCenter Server.
  - vCenter Server Name: A display name for the vCenter Server.
  - vCenter Server Description: A description of the vCenter Server. This is optional.

- vCenter Server Username: User name to access the vCenter Server.
- vCenter Server Password: Password for the user name entered.
- Database URL: The IP address of the system on which the vCenter Chargeback database is installed along with either the port at which the database listener service is running or the vCenter Chargeback database instance name.

  For Oracle database, the database URL can be in one of the following formats:

  ```
  <IP Address>:<TNS Listener Port>
  <Host Name>:<TNS Listener Port>
  ```

  For SQL Server, the database URL can be in one of the following formats:

  ```
  <IP Address>\<Database Instance Name>
  <Host Name>\<Database Instance Name>
  ```

- Database Name: The name of the vCenter Server database. For example, vim_vcdb, which is the default name given by vCenter Server.
- Database Username: A database user name to access the vCenter Database.
- Database Password: A password for the database user name entered.
- Database Type: The database type can be either SQL Server (default) or Oracle. An ID value of 1 means SQL server and 2 means Oracle.
- DataSourceAuthType: The authentication type ID, which for SQL server can be Windows authentication or database authentication. The ID 1 means Credential-based Authentication and the ID 2 indicates Windows authentication.
- vCenter Server View Type: The type of view to be used to display the entities in the vCenter Server hierarchy. This can be either Hosts and Clusters or Virtual Machines and Templates. An id value of 1 indicates Hosts and Clusters and a value of 2 indicates Virtual Machines and Templates. These view types are the same as the corresponding vCenter Server view types, and the entities in the vCenter Server hierarchy will be displayed in a manner similar to the vCenter Server views. Once set, this option cannot be edited for the vCenter Server.
- Register As VI Client Plugin: Select this option if you would like to register vCenter Chargeback as a plug-in to VI Client. If you select this option, each time you log in to this vCenter Server using the VI Client, the vCenter Chargeback plug-in is displayed on the VI Client. You can access the vCenter Chargeback application from the VI Client.
- Enable Stats Replication: Select this option if you want the resource usage statistics from the vCenter Server Database to be replicated in the vCenter Chargeback Database.

- The following is a sample request XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <VCenterServers>
        <VCenterServer>
            <Url>10.112.107.21:443</Url>
            <Name>vc1</Name>
            <Description>vc1</Description>
            <UserName>administrator</UserName>
            <Password>xxxx</Password>
            <PluginRegistered>true</PluginRegistered>
            <StatsSync>true</StatsSync>
            <DataSourceUrl>
                10.112.107.21\sqlexp_vim
            </DataSourceUrl>
            <DataSourceName>vim_vcdb</DataSourceName>
            <DataSourceUserName>sa</DataSourceUserName>
            <DataSourcePassword>xxxx</DataSourcePassword>
            <DataSourceType id="1" />
            <DataSourceAuthType id="1" />
            <VCenterServerView id="1" />
```

```
            <ForceUpdate>false</ForceUpdate>
        </VCenterServer>
    </VCenterServers>
</Request>
```

- The following is a sample program that calls the Add vCenter Server Information API. This program assumes that the request XML is populated with the required information.

```
/**
* This method to add the vCenter-Server in vCenter-Chargeback application
*
* @param requestFilePath
* @param baseURL
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/

@SuppressWarnings("deprecation")
public static void sampleAddVCenterServerMethod(String requestFilePath, String baseURL,
               String clientVersion) throws IOException, JDOMException,
               NoSuchAlgorithmException,
KeyManagementException, HttpException {
PostMethod post = null;
NameValuePair[] parameters = {new NameValuePair("version", clientVersion)};
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https", (ProtocolSocketFactory) new
               FakeSSLCertificateSocketFactory(), 443));
String uri = "https://" + baseURL + "/vCenter-CB/api/vCenterServer";
try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
}
finally {
    if (post != null) {
    post.releaseConnection();
}
}
}
}
```

- If the vCenter Server is successfully added, the API returns an XML that provides the vCenter Server identifier.

Create a Custom Chargeback Hierarchy

Use the Add a Chargeback Hierarchy API to create a hierarchy with the given name and description.

- The URL for this API is `<HTTP_request_method> <Base_URL>/hierarchy`. You can define a sample call like this:

  `POST https://123.123.123.123/vCenter-CB/api/hierarchy`

- The API takes a request XML in which you can specify the name and description for the hierarchy. A sample input XML is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <Hierarchies>
        <Hierarchy>
```

```
            <Name>Test_Hierarchy</Name>
            <Description>Test Hierarchy</Description>
        </Hierarchy>
    </Hierarchies>
</Request>
```

- The following is a sample program that calls the Add a Chargeback Hierarchy API. This program assumes that the request XML is populated with the required information.

```
/**
* This method is to add a vCenter-ChargeBack hierarchy in
* vCenter-ChargeBack
*
* @param requestFilePath
* @param baseURL
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleAddHierarchyMethod(String requestFilePath, String baseURL,String
               clientVersion) throws IOException, JDOMException, NoSuchAlgorithmException,
KeyManagementException, HttpException {
PostMethod post = null;
NameValuePair[] parameters = {new NameValuePair("version", clientVersion)};
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https", (ProtocolSocketFactory) new
               FakeSSLCertificateSocketFactory(), 443));
String uri = "https://" + baseURL + "/vCenter-CB/api/hierarchy";
System.out.println(uri);
System.out.println(bodyString);
try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
}
finally {
    if (post != null) {
    post.releaseConnection();
}
}
}

}
```

- If successful, the API returns the new hierarchy.

Add a vCenter Server Entity to the Chargeback Hierarchy

This API helps you create a vCenter Server entity under a specified parent entity in a Chargeback hierarchy.

- The URL for this API is `<HTTP_request_method> <Base_URL>/hierarchy/{hierarchyId}/entity/{parentEntityId}`. You can define a call like this:

  `POST https://123.123.123.123/vCenter-CB/api/hierarchy/11/entity/101`.

- The API takes a request XML file in which you can specify the hierarchy identifier, entity name and description for the hierarchy. A sample request XML is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <Hierarchies>
```

```
        <Hierarchy id="38">
            <Entities>
                <Entity id="171">
                    <VCenterServers>
                        <VCenterServer id="145">
                            <Entities>
                                <!-- Either id or MoId needs to be there. id takes
                                   precedance over MoId-->
                                <Entity id="630">
                                    <!-- <MoId>host-2460</MoId> -->
                                </Entity>
                            </Entities>
                        </VCenterServer>
                    </VCenterServers>
                </Entity>
            </Entities>
        </Hierarchy>
    </Hierarchies>
</Request>
```

- The following is a sample program that calls the Add an Entity API. Make sure that you:
  - Update the request XML for Login API with required values and run the `login.java` sample code.
  - Run `AddVCenterServer.java` and read the vcId from the response
  - Read the vcEntity Id which you need to add under vCenter-ChargeBack hierarchy entity, from the response of `sampleGetVcHierarchyMethod()`
  - Add a vCenter-ChargeBack Hierarchy and from the response, read the hierarchy Id and the CBEntity Id under which the vcEntity is to be added
  - Update the Hierarchy id, cbEntity id, VCenterServer id and vcEntity id in the request XML file.

```
/**
* This method is to add a new vCenter-Server entity under
* vCenter-Chargeback Hierarchy entity
*
* @param requestFilePath
* @param baseURL
* @param hierachyId
* @param CBEntityId
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleAddNewVCenterServerEntity(String requestFilePath, String
               baseURL, int hierachyId, int CBEntityId,String clientVersion) throws
               IOException,
JDOMException, NoSuchAlgorithmException, KeyManagementException, HttpException {
PostMethod post = null;
NameValuePair[] parameters = {new NameValuePair("version", clientVersion)};
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https", (ProtocolSocketFactory) new
               FakeSSLCertificateSocketFactory(), 443));
String uri = "https://" + baseURL + "/vCenter-CB/api/hierarchy/" + hierachyId +
               "/entity/" + CBEntityId;
try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
```

```
}
finally {
    if (post != null) {
    post.releaseConnection();
}
}
}
}
```

- If successful, the API returns an XML file that indicates the status.

Add a Fixed Cost

A fixed cost is a definite cost that can be charged on an entity. Using vCenter Chargeback API, you can create fixed costs for entities.

- The URL for this API is `<HTTP_request_method> <Base_URL>/fixedCost`. You can define a call like this:

  `POST https://123.123.123.123/vCenter-CB/api/fixedCost`

- The API takes a request XML file in which you can specify the fixed cost name and its description. The following is a sample request XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <FixedCosts>
        <FixedCost>
            <Name>Fixed Cost 1</Name>
            <Description>Fixed Cost 1 description</Description>
        </FixedCost>
    </FixedCosts>
</Request>
```

- The following is a sample program that calls the Add an Fixed Cost API. Make sure that you:
  - Update the `Login-request.xml` with required values and run the `login.java` sample code.
  - Update the `addFixedCost-request.xml` with fixed cost name and its description.

```
/**
* This method is to add a fixed cost in vCenter-ChargeBack
*
* @param requestFilePath
* @param baseURL
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleAddFixedCost(String requestFilePath,
String baseURL, String clientVersion) throws IOException,
JDOMException, NoSuchAlgorithmException, KeyManagementException,
HttpException {
PostMethod post = null;
NameValuePair[] parameters = { new NameValuePair("version",
clientVersion) };
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https",
(ProtocolSocketFactory) new FakeSSLCertificateSocketFactory(),
443));
String uri = "https://" + baseURL + "/vCenter-CB/api/fixedCost";
```

```
try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
    }
finally {
    if (post != null) {
    post.releaseConnection();
    }
}
}
}
```

- If the task is successful, the API returns an XML file that returns the identifier of the new fixed cost.

Modify a Fixed Cost Value

Using the Modify Fixed Cost API, you can update fixed cost id, value and duration for a fixed cost.

- The URL for this API is `<HTTP_request_method> <Base_URL>/fixedCost/{fixedCostId}/values`. You can define a call like this:

  `PUT https://123.123.123.123/vCenter-CB/api/fixedCost/{fixedCostId}/values`

- The API takes a request XML file in which you can specify the fixed cost name and its description. The following is a sample request XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <FixedCosts>
        <FixedCost id="1">
            <Values>
                <Value>
                    <Cost>3.1415</Cost>
                    <Duration id="1"/>
                </Value>
            </Values>
        </FixedCost>
    </FixedCosts>
</Request>
```

- The following is a sample program that calls the Add an Fixed Cost API. Make sure that you:
  - Update the `Login-request.xml` with required values and run the `login.java` sample code.
  - Run the Add Fixed Cost API and read the fixedCostId from the response.
  - Update the fixed Cost Id, the cost and duration id in `modifyFixedCostValues-request.xml` file.

```
/**
* This method is to modify the values of an existing fixedCost
*
* @param requestFilePath
* @param baseURL
* @param startTime
* @param endTime
* @param clientVersion
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleModifyFixedCostValues(String requestFilePath,
    String baseURL, int fixedCostId, long startTime, long endTime,
    String clientVersion) throws IOException, JDOMException,
```

```
    NoSuchAlgorithmException, KeyManagementException, HttpException {
PutMethod put = null;
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);
Protocol.registerProtocol("https", new Protocol("https",
    (ProtocolSocketFactory) new FakeSSLCertificateSocketFactory(),
    443));
String uri = "https://" + baseURL + "/vCenter-CB/api/fixedCost/"
    + fixedCostId + "/values";
System.out.println(uri);
NameValuePair[] parameters = {
    new NameValuePair("version", clientVersion),
    new NameValuePair("startTime", String.valueOf(startTime)),
    new NameValuePair("endTime", String.valueOf(endTime)) };
try {
    put = new PutMethod(uri);
    put.setRequestBody(bodyString);
    put.setQueryString(parameters);
    client.executeMethod(put);
    System.out.println(put.getResponseBodyAsString());
}
finally {
    if (put != null) {
    put.releaseConnection();
}
}
}
}
```

Generate a Report

Using the Generate a Raw Report API, you can generate a report that provides information on cost details for the given entity.

- The URL for this API is `<HTTP_request_method> <Base_URL>/rawReport`. You can define a call like this:

  `POST https://123.123.123.123/vCenter-CB/api/rawReport`

- The API takes a request XML file in which you can specify details about the hierarchy, entity, resource counters and the cost model. The following is a sample request XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.vmware.com/vcenter/chargeback/1.0.1">
    <ReportTemplates>
        <ReportTemplate>
            <Name>Name</Name>
            <Description>Description</Description>
            <ReportType>COST_REPORT</ReportType>
            <Hierarchies>
                <Hierarchy id="1">
                    <Entities>
                        <Entity id="2" />
                    </Entities>
                </Hierarchy>
            </Hierarchies>
            <CostModels>
                <CostModel id="2" />
            </CostModels>
            <ComputingResources>
                <ComputingResource id="2" />
            </ComputingResources>
```

```
            <BillingScheduleType>NOW</BillingScheduleType>
            <BillingPeriod>
                <Now>
                    <StartDate>1211871400000</StartDate>
                    <EndDate>1232399800000</EndDate>
                </Now>
            </BillingPeriod>
            <DataFilter>
                <DataDepth>1</DataDepth>
                <ExcludeFilters>
                    <Filter id="CategorizedCosts" />
                    <Filter id="CategorizedUsage" />
                    <Filter id="InformationMessages" />
                    <Filter id="CostDetails" />
                </ExcludeFilters>
            </DataFilter>
        </ReportTemplate>
    </ReportTemplates>
</Request>
```

- The following is a sample program that calls the Generate Raw Report API. Make sure that you:
  - Update the request XML for Login API with required values and run the `login.java` sample code.
  - Run `AddVCenterServer.java` and read the vcId from the response.
  - Read the vcEntity Id, which you need to add under vCenter-ChargeBack hierarchy entity, from the response of sampleGetVcHierarchyMethod()
  - Add a vCenter-ChargeBack Hierarchy and from the response, read the hierarchy Id and CBEntity Id under which the vcEntity is to be added.
  - Update the Hierarchy id, cbEntity id, VCenterServer id and vcEntity id in `AddVCenterEntity-request.xml` file.
  - Add the vcEntity under vCenter-ChargeBack hierarchy entity by calling method addNewVCenterServerEntity()
  - Update the Hierarchy id, Entity id (on which you want to generate report), StartDate and EndDate in `GenerateRawReport-request.xml`.

```
/**
* This method is to generate report for a vCenter-ChargeBack hierarchy
* entity in vCenter-ChargeBack
*
* @param requestFilePath
* @param baseURL
* @throws IOException
* @throws JDOMException
* @throws NoSuchAlgorithmException
* @throws KeyManagementException
* @throws HttpException
*/
@SuppressWarnings("deprecation")
public static void sampleGenerateReportMethod(String requestFilePath, String
                baseURL,String clientVersion) throws IOException, JDOMException,
                NoSuchAlgorithmException,
KeyManagementException, HttpException {
PostMethod post = null;
NameValuePair[] parameters = {new NameValuePair("version", clientVersion)};
Document requestDocument = CommonUtil.getXMLDocument(requestFilePath);
String bodyString = CommonUtil.xmlAsString(requestDocument);

Protocol.registerProtocol("https", new Protocol("https", (ProtocolSocketFactory) new
                FakeSSLCertificateSocketFactory(), 443));
String uri = "https://" + baseURL + "/vCenter-CB/api/rawReport";
System.out.println(uri);
System.out.println(bodyString);
```

```
try {
    post = new PostMethod(uri);
    post.setQueryString(parameters);
    post.setRequestBody(bodyString);
    client.executeMethod(post);
    System.out.println(post.getResponseBodyAsString());
}
finally {
    if (post != null) {
    post.releaseConnection();
}
}
}

}

}
```