Main page: Cisco Unity Express -- Programmatic Interface Service Programming Guide

Previous page: Cisco Unity Express -- Client Software Choices

Next page: Cisco Unity Express -- REST PI Service Operations

Search the Cisco Unity Express documentation on the DocWiki:

Loading

Contents

- 1 Resource URI
- 2 HTTP Methods
  - 2.1 GET Operation
  - 2.2 POST Operation
  - 2.3 PUT Operation
  - 2.4 DELETE Operation
- 3 HTTPS Support
- 4 Authentication
- 5 MIME Types
- 6 Error Handling and Response Codes
- 7 API Versioning and Backward Compatibility

Resource URI

The CUE PI service defines a URI for each of its resources (for example, Users, Groups, Mailboxes, and so on). Typically, resource URI represents nouns instead of verbs; in other words, resources are things and not actions.

Clients can determine the resource URI for various resources in the following possible ways:

1. By referring to this document to determine how to construct the resource URI for a given resource.
2. When a resource created by invoking POST operation on the CUE PI service, the resource URI of the resource created is provided as part of the HTTP response header (Refer to section 5.2.2 for details).
3. Resource URI is included with each resource returned when a read operation is performed by calling HTTP GET on the CUE PI service.

The resource URI for individual resources published in this document, as well as those returned by GET operations on the CUE PI service, does not include protocol, hostname, and port. Clients can create the fully qualified URL for the resource by prepending the CUE hostname and PI service application name "rest". For example, the published resource URI for a voicemail subscriber profile is:

```
/voicemail/users/{userId}
```

Thus, the fully qualified resource URI for a subscriber with userId "johndoe" on a CUE system host named "cue_host" would be:

```
http://cue_host/rest/voicemail/users/johndoe
```

The above resource URI represents a single entity, a user with id "johndoe". A resource URI can also represent a collection of entities in the system. For example, a collection of users in the system is represented by the resource:

```
http://myhost/rest/voicemail/users
```

Association relationships between resources is also represented through resource URI structure. For example a collection, of subscribers belonging to a group named "admin" is represented by the resource:

```
http://myhost/rest/voicemail/groups/admin/users
```

Some resource URIs support HTTP query parameters that clients could append to the resource URI to fine tune the result set or choose available update options. Query parameters where supported are documented in this guide.

HTTP Methods

This sections describes the list of HTTP methods supported by the CUE PI service. The following table provides the list of supported HTTP methods with mapping to the actual operations performed on the resource addressed in the request.

Table 1: HTTP to CRUD Operation Mapping

HTTP Method	CRUD Operation
Post	Create Resource
Get	Read Resource
Put	Update Resource
Delete	Delete Resource

GET Operation

GET method is used for retrieving a resource. A GET request is safe, that is, it does not cause a change in system state besides generation of logging and audit information. While processing the GET request, the

HTTP message entity is not checked by the CUE PI service and all the required information to read the resource is obtained from the resource URI and query parameters (if supported) only.

GET using Query parameter "fields" with Resource URI -

When a GET operation is performed on a resource, the CUE PI service by default includes all the fields defined for the resource. However, for most of the resources defined in the CUE PI service, clients can customize the set of fields returned by specifying a query parameter named "fields" with value as comma separated list of fields to be included in the result. The client can also specify a blank field set to retrieve the resource URI.

For example, to retrieve only the "firstName" and "lastName" fields for a subscriber having userId "johnDoe", the client needs to perform GET on URI:

```
http://rest/voicemail/users/johnDoe?fields=firstName,lastName
```

To retrieve the resource URI of all users, the client can specify an explicit blank field set:

```
http://rest/voicemail/users?fields=%
```

A field name in the list would be ignored if it is not applicable for the resource being queried. The result set will contain only those fields which are part of the resource definition. If no valid field name is provided, the result will only include the resource URI for the resource.

Any additional query parameter supported for a resource is documented along with the resource definition. An unrecognized query parameter will be ignored and query will perform as if no query parameter is specified in the URI.

POST Operation

HTTP POST operation is used for creating new instances of a resource. The CUE PI service obtains the representation of the resource to be created from the HTTP message entity. If the POST operation results in a successful creation of a resource, the CUE PI service responds with HTTP status code 201 (CREATED) and provides the resource URI of the newly created resource as value for HTTP response header "location".

PUT Operation

PUT operation is used for updating information about an existing resource. While processing the PUT request for a resource, the CUE PI service will update only those fields specified explicitly in the resource payload. To set a field to blank value, it needs to be included in the payload with an empty field value.

DELETE Operation

HTTP DELETE operation is used to delete a resource. While processing the delete request, the HTTP message entity is not checked by the CUE PI service and all the required information to delete the resource is obtained from the resource URI and query parameters (if supported) only.

HTTPS Support

HTTP is plain text-based protocol so sensitive data sent over HTTP are not secured. For added security, the CUE PI service supports HTTPS as a communication protocol.

The client can configure the system to enable HTTPS for all web applications, including PI, by using existing CUE CLI. Please refer to Cisco Unity Express Administrator Guide for details on steps for enabling HTTPS in a CUE system.

If CUE system is enabled for HTTPS mode and request is made using HTTP protocol, request will be redirected to the client suggesting use of "HTTPS" in the request.

Authentication

For authorizing client operations, the CUE PI service expects clients to provide valid credentials with every HTTP request made to the service.

- Username and Password must be set in request header "Authorization", as per the basic access authentication scheme. For basic access authentication, the user name is appended with a colon and concatenated with the password. The resulting string is encoded with the Base64 algorithm and passed in the HTTP request as "Authorization" header.

This example shows generation of authorization string for user name Aladdin and password open sesame:

- User-name and password are concatenated with a colon in the middle to get the string

```
Aladdin:open sesame
```

- The string **Aladdin:open sesame** is Base64 encoded resulting in authorization string **QWxhZGRpbjpvcGVuIHNlc2FtZQ==**

If using Linux CLI:

```
$ echo -n "Aladdin:open sesame" | openssl enc -base64
QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

- The Base64 encoded string is passed with HTTP requests using **Authorization** header:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

- CUE PI service requires that the user credentials passed are of "superuser" privilege. The "superuser" privilege will be verified by the implementation as part of the authentication process. If the user does not have the " superuser " privilege, the request will fail. Please refer to Cisco Unity Express Administrator Guide for information on configuring a user with "superuser" privilege.

MIME Types

Most of the resources defined by the CUE PI service use XML representation. Some resource also support binary content like prompts, greetings, and binary script files. While making POST and PUT requests, clients must set the "Content-type" header to the appropriate mime type value. While making GET calls, the client can choose to specify the MIME type in the "Accept" header.

The mime to be used for all resource definitions provided in this document, if not specified otherwise, is "application/xml".

Error Handling and Response Codes

The CUE PI service uses standard HTTP status codes (1xx, 2xx, 4xx, 5xx) for indicating the status of the client action requests. In general, 2xx status codes are success status code and 4xx/5xx is used for non-success status.

When CU PI service fails to process the client request, either because of the error in client supplied data or internal failure in the server, the service will choose an appropriate HTTP status code to return to the client.

In most error cases, in addition to the HTTP status code, the service will also provide detailed error response as XML document. The XML error response document (section 6.2.4) will contain a CUE PI service defined error code, error message, and some additional information about the error condition.

API Versioning and Backward Compatibility

The CUE PI service provides product version information in the HTTP response header for every response it sends to the client. Clients can examine the response header "PI-Version" to extract the string value specifying the product version information.

This is the first release of the CUE PI service and any future versions of the service are expected to be backward compatible with clients that are still using previous versions of XML schema.

To achieve backward compatibility in the future versions, both client and the CUE PI service needs to follow certain implementation guidelines that are explained as follows:

- API Elements Are Forever (Applies to Server)

That is, once an element of an API (URI, Representation, Method, Fields) is published, it cannot be removed in subsequent versions. This rule is crucial for ensuring compatibility between an older client and a newer service.

- Ignore Unrecognized Elements (Applies to Client)

If a client receives elements within a representation that it does not know how to process, then it must silently ignore it. This rule ensures compatibility between an older client communicating with a newer version of service with new elements defined in the schema.

- Apply Defaults For Missing Elements (Applies to Server)

If the service does not receive an element within a representation that it needs, it must silently assume a default value for that element. This rule provides backward compatibility, where a newer service is able to interface with an older client.