VOSS-4-UC
API Guide

Release 20.1.1 **Early Field Trial**

Jan 28, 2021

Legal Information

Please take careful note of the following legal notices:

- Copyright © 2021 VisionOSS Limited.
  All rights reserved.

- VOSS, VisionOSS and VOSS-4-UC are trademarks of VisionOSS Limited.

- No part of this document may be reproduced or transmitted in any form without the prior written permission of VOSS.

- VOSS does not guarantee that this document is technically correct, complete, or that the product is free from minor flaws. VOSS endeavors to ensure that the information contained in this document is correct, whilst every effort is made to ensure the accuracy of such information, VOSS accepts no liability for any loss (however caused) sustained as a result of any error or omission in the same.

- This document is used entirely at the users own risk. VOSS cannot be held responsible or liable for any damage to property, loss of income, and or business disruption arising from the use of this document.

- The product capabilities described in this document and the actual capabilities of the product provided by VOSS are subject to change without notice.

- VOSS reserves the right to publish corrections to this document whenever VOSS deems it necessary.

- All vendor/product names mentioned in this document are registered trademarks and belong to their respective owners. VOSS does not own, nor is related to, these products and vendors. These terms have been included to showcase the potential of the VOSS solution and to simplify the deployment of these products with VOSS should you select to utilize them.

Security Information

This product may contain cryptographic features that may be subject to state and local country laws that govern the import, export, transfer and use of such features. The provision of this software does not imply that third-party authorization to import, export, distribute or use encryption in your particular region has been obtained. By using this product, you agree to comply with all applicable laws and regulations within your region of operation. If you require further assistance, please contact your dedicated VOSS support person.

Contents

1 Overview

1.1. Introduction

1.1.1. What's New

API Guide: Release 20.1.1

- EKB-3608: Support a mechanism to clear resource tags. See: *API Parameters*

- VOSS-670: Small Enhancements 20.1.1 (EKB-3608: Support a mechanism to clear resource tags) . See: *Meta data*

- VOSS-670: Small Enhancements 20.1.1 (EKB-3785: List view summary attributes must honor FDP title overrides) . See: *Summary Attributes*

- VOSS-695: Investigate and Optimise Memory Utilisation and Performance 20.1.1 (EKB-3868: Disable support for limit=0 queries on the API) . See: *API Parameters*

1.1.2. API Introduction

The secure and comprehensive API provides a single point of integration with multiple business systems that require information and need use functionality exposed by the product, the underlying managed network and related products that are enabled by the core.

The REST-based API covers all functionality provided by the product and includes a comprehensive JSON-based schema with schema rules, meta data and data that simplifies integration.

Refer to the API Guide for more information on integrating with the VOSS-4-UC API.

For a reference of the schema and the operations applicable for each resource in the system, refer to the relevant API Reference. Resources are classified by the type of model in the system (data, device, domain, relation or view), for example `data/AccessProfile`, `device/cucm/Phone`, and so on. Depending on the installed modules and their feature packages, the API of feature package models may be available, for example `relation/Subscriber`, `view/QuickSubscriber`, and so on.

The product is fully integrated with external LDAP directories and SAML identity providers, allowing users to utilize existing identity management system to provide seamless access to portals developed using the product.

Note: References to HCM-F and Shared Data Repository (SDR) are only relevant if installed.

1.1.3. API System Concepts

In order to understand the API, an understanding of two basic concepts is required:

- Models
- Hierarchy

The term "model" is used to describe the types of JSON objects fulfilling purposes such as defining data structures, containing data, defining GUI forms, mapping data from devices or other models. The system employs the following types of models:

- Data Models
- Device Models
- Domain Models
- Relations
- Views

Data in the system is represented using Data and Device models.

Device models are generated from the application API of entities that are provisioned on devices.

Domain models, relations and views wrap the Data or Device models by means of references to them.

Data models can be created and are stored in the database. Data models contain a JSON schema/metadata for the entities exposed by the underlying database. The schemas for the data models are stored in the database and represent the structure that instances of the data model conforms to.

Device models interface with devices and services on the system. For example:

- Unified CM device models interface with the Call Manager's AXL SOAP API.
- CUC device models interface with Unity Connection's RESTful API.

The ability to rapidly develop and deploy new device interfaces provides an extensible mechanism to add support for additional provisioning tasks or additional southbound integration into other business systems. Domain models act as "containers" of other data-, device- and domain models along with provisioning workflows to represent the management of a created feature.

Relations do not store data on the system. Instead, they relate groups of resource types such as device models, data models or other domain models.

Views provide a mechanism to define an arbitrary schema which can be used to define a user input screen.

1.1.4. Hierarchy

A system hierarchy node is present at first startup of the system. Each entity that is attached to the hierarchy has an address represented by a `pkid`, that is defined as a standard URI. Hierarchies can be created under the system hierarchy node, because the hierarchy is exposed as a RESTful API. API calls are made with reference to the hierarchy.

1.1.5. Basic REST

The system uses a REST (Representational State Transfer) API. For details on this type of API, see for example:

- http://en.wikipedia.org/wiki/Representational_state_transfer

1.1.6. API Traversal

The system represent the reference of an entity in the system as Hypermedia as the Engine of Application State (HATEOS). Each reference position is represented by an object pair pkid and href.

A client integrates with VOSS-4-UC entirely through hypermedia dynamically provided by the VOSS-4-UC application and does not need any prior knowledge of how to interact with the system other than a generic understanding of hypermedia. This means that no WADL is provided. This also means that the client and VOSS-4-UC can be decoupled in a way that allows VOSS-4-UC to evolve independently.

A client enters the VOSS-4-UC through a simple fixed URL. All future actions the client may take are discovered within resource representations returned from the server

The detailed URL tree endpoint information is available in the relevant API Reference Guides for the core and features.

This response emulates the HierarchyNode list response and utilizes the parent and children in the meta references section of the response as discussed in Meta Data References.

1.1.7. Request and Response Patterns

The request and response patterns between service requester and VOSS-4-UC is summarized below. For details, refer to the topics in the chapter called *Anatomy of an API Response*.

For synchronous operations:

1. Service Requestor sends a accessor (e.g. Get, List) request with request parameters to VOSS-4-UC.

2. Either:

    a. VOSS-4-UC responds synchronously with a Get/List response.

    b. VOSS-4-UC responds synchronously with a fault response.

For asynchronous operations:

1. Service Requestor sends a mutator (e.g. Add, Modify, Delete) request with parameters.

2. The Add/Update/Delete transaction is scheduled on the VOSS-4-UC transaction queue with a transactionID.

3. VOSS-4-UC responds synchronously with either:

    a. An Add/Update/Delete response and a transactionID.

    b. A fault response.

4. The external system either:

    a. Polls the system to retrieve the status of the transaction as needed, or

    b. Specifies a callback URL (with an optional username and password if the interface is secured (recommended)) and waits for a asynchronous transaction status callback (recommended).

        When the transaction completes, VOSS-4-UC sends an async transaction status callback message to the callback URL specified in the request.

1.2. Anatomy of an API Request

1.2.1. General Structure of the API

The VOSS-4-UC API accesses system resources or tools.

- **Resources**

The general structure of an API URL for accessing a system resource (an endpoint) is:

Method https://servername/api/Resource/Action/?Parameters

Where:

Method

```
[GET|POST|DELETE|PUT|PATCH]
```

Servername

The installation server determines the base URL, e.g. `https://servername`. In a cluster environment, this is the address of the web proxy node. Refer to the Install Guide for cluster deployment information.

api

A static string in the URL that is a part of the endpoint.

Resource

```
(str:modeltype/str:modelname)[/pkid]
```

Refer to the relevant API Reference guides for a list of supported resources.

Action

For a complete list of actions supported for resources in the system and for a list of custom actions, refer to the relevant API Reference Guides.

Parameters

```
[(str:api parameter)[&(str:api parameter)...]]
```

The HTTP methods and parameters are described in relevant sections. The different resources supported in the system are described in the API Reference Guides.

- **Tools**

For tools, the general structure of the URL structure is for example:

[GET|POST] /api/tool/(str:tool_name)/

1.2.2. Format

The system API supports the following format HTTP headers when handling and responding to requests.

Field Name	Description	Value
Content-Type	The format type of the body of the request (used with POST and PUT requests)	application/json
Content-Type	The format type of the body of the request (used with PATCH requests)	application/json-patch+json
Accept	Content-Types that are acceptable in response	application/json

1.2.3. Authentication

The system controls access to its service through HTTP basic authentication. The technique is defined in section 11.1 of RFC1945 which is simple to implement, uses standard HTTP headers.

The HTTP Basic Access Authentication requires authorization credentials in the form of a user name and password before granting access to resources in the system. The username and password are passed as Base64 encoded text in the header of API requests.

The HTTP header format for authentication is defined in the table below.

Field Name	Description	Value
Authorization	Basic authentication is supported.	Basic [Base64 encoded credentials]

For example:

The Base64 encoded credentials for user name of joe and a password of bloggs.

For example, from a command line (note the removal of the new line in the `echo` command):

```
$ echo -n "joe:bloggs" | base64
am9lOmJsb2dncw==
```

the header will be:

```
Authorization:   Basic am9lOmJsb2dncw==
```

For example, using **curl**:

```
curl -k -H "Authorization: Basic am9lOmJsb2dncw=="
  'https://hostname/api/data/MyModel/'
```

It is required that all requests be conducted over a secure session, such as HTTPS or SSL.

A VOSS-4-UC self-signed certificate needs to be installed into a local trust store of the client application.

1.2.4. Authorization

The access profile of a user determines whether he or she can perform a given operation on a model. The user can also only access items below the position they are defined in the hierarchy.

1.2.5. HTTP Methods

The API supports the following HTTP methods:

GET

- Used to query a resource or a list of resource.

POST

- Used to create a new resource.

- The data is submitted as a JSON object.

- The return value is the pkid of the resource.

PUT

- Used to update the data of a resource.

- The resource URL includes the resource pkid.

- The data to be updated is submitted as a JSON object.

PATCH

- Used to update the data of a resource.

- PATCH request body in JSON Patch format

- Content-Type is "application/json-patch+json"

- JSON Patch: http://tools.ietf.org/html/rfc6902

DELETE

- Used to delete a resource.

- The resource URL includes the resource pkid.

- The DELETE method can also be used to delete multiple resources on one request as a "bulk delete".

1.2.6. PUT Versus PATCH

For PUT methods the resource data is replaced with the data specified in the request. All fields of the resource are replaced with the fields in the request.

This means that:

- Fields not present in the request that are present in the resource will be dropped from the resource.

- Fields present in the request that are not present in the resource will be appended to the resource.

- The data of fields present in the request is used to update fields that already exist in the resource.

PATCH methods operate in two modes depending on the content type:

- Content type: application/json

- The values of data fields present in the request is used to update the corresponding resource fields. This means that:

  – Fields present in the request but not in the resource is appended to the resource.

  – The value of each field that is already present in the resource is updated from the request data.

   – Field values that are set to null in the request is dropped from the resource.

   – Fields that are present in the resource but not in the request are left untouched.

 • Content type: application/json-patch+json

   – Existing resource data is patched according to RFC6902.

Modifying data fields:

 • To drop the field from a data model, specify null as the parameter value (i.e. {"field": null}).

 • To blank out a string value set the parameter value to an empty string (i.e. {"field":""}).

1.2.7. API Parameters

The hierarchy parameter is required for each API request and can be specified as any of the following:

 • the pkid of the hierarchy node in the form of a UUID, for example 1c055772c0deab00da595101

 • in dot notation, for example ProviderName.CustomerName.LocationName

To obtain the pkid of a hierarchy node, refer to the path element in the metadata of data/HierarchyNode resource.

Note: For the purposes of simplifying the documentation, the hierarchy API parameter &hierarchy=[hierarchy] is not included in all examples in this document. Specifying the hierarchy is however required in all API requests where the instance pkid is not referenced. In the examples, [hierarchy] is substituted with the caller's hierarchy id.

Format

The system API supports the following request parameters for data format when handling requests.

Key	Description	Value
format	The format type of the body of the request	json

A request of the following format returns HTML:

```
GET /api/(str:model_type)/(str:model_name)/help/
```

A parameter &format=json is not displayed in all examples, but it is required for all requests unless a different format is specifically stated.

Configuration Template and Template Name

The Configuration Template can be specified in the POST request parameters for a resource as follows:

```
POST /api/(str:model_type)/(str:model_name)/&template_name=[CFG name]
```

Key	Description	Value
template	Apply the Configuration Template with pkid [CFG pkid] to the payload of the POST request.	[CFG pkid]
template_name	Apply the Configuration Template with name [CFG name] to the payload of the POST request.	[CFG name]

Field Display Policy

Field Display Policy can be specified in the GET request parameters for a resource as follows:

```
GET /api/(str:model_type)/(str:model_name)/add/
```

Key	Description	Value
policy	Return a model form schema where the Field Display Policy with pkid [FDP pkid] is applied to it. Use `policy` with the parameters `schema` and `format=json`.	[FDP pkid]
policy_name	Return a model form schema where the Field Display Policy with name [FDP name] is applied to it. Use `policy` with the parameters `schema` and `format=json`.	[FDP name]

Cached

The API can return cached data from the system or data from devices, using the following format:

```
GET /api/(str:model_type)/(str:model_name)/[pkid]/
```

Key	Description	Value	Default
cached	System will respond with resource information where the data was obtained from cache. (Functionally only applicable to device models and domain models containing device models)	true, false	true

Note: From 11.5.2 onwards, the API URL `cached` parameter on the Subscriber list (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will always display cached information and will not refresh the information from the device during a list query with `cached=false`.

Resource instance

To identify a single resource, the API call contains the single resource (pkid) using the following format:

```
GET /api/(str:model_type)/(str:model_name)/(pkid)/
```

Schema and Schema Rules

To obtain the schema or schema rules of a resource, use the following parameters to an API request:

```
GET /api/(str:model_type)/(str:model_name)/?
    hierarchy=[hierarchy]&schema=true&schema_rules=true
```

Key	Description	Value
schema	Return the schema of the resource. Use with the parameter `format=json`	true, false
schema_rules	Return the GUI Rules and Field Display Policies of the resource if available. Use with the parameters `format=json` and `schema` to see `schema_rules` in the response.	true, false

List pagination

The system API supports the following two tables of API request parameters when specifying the format of and structure of the resources to list.

- Pagination parameters

Key	Description	Value	Default
skip	The list resource offset. If the `Range` request header is used, it will override this parameter.		0
limit	The maximum number of resources returned. The maximum value is 2000. If the `Range` request header is used, it will override this parameter.	1-2000	50
count	Specify if the number of resources should be counted. If false, the `pagination` object in the response shows the `total` as 0, so no total is calculated and the API performance is improved.	true, false	true

List format

- List format parameters

Key	Description	Value	Default
order_by	The summary attribute field to sort on		First summary attribute
direction	The direction of the summary attribute field sort (asc:ascending, desc: descending)	asc, desc	asc
summary	Only summary data is returned in the data object	true, false	true
traversal	The direction of the resource lookup of resources tied to the hierarchy tree from the hierarchy node provided as parameter	up, down, local	down

Note: From 11.5.2 onwards for `api/relation/Subscriber`:

- The API URL `summary` parameter on the Subscriber list (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will always display summarized information and will not display full CUCM User data with `summary=false`.

- The API parameter `traversal=up` on the Subscriber list (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will default to display resources down the hierarchy tree with `traversal=up`.

Filter

Models that have the `list` action defined in their schema can also be filtered by using a number of URL filter parameters in parameter sets of four key-value pairs.

Filters also apply to the `api/tool/Transaction/` endpoint, which has additional filter functionality to filter by transaction ID. Refer to the topic on Filter Transactions.

These parameters can be added in addition to the parameters available to list resources as in the topic on API Parameters.

A single filter query can contains one or more sets of the following four parameters:

Key	Description	Value	Default
filter_field	The model attribute name to filter.	The name of the attribute in the list of `summary_attrs` in the model schema.	
filter_condition	The matching operator for the `filter_field`. If `equals` is used in a condition, then other filter sets are ignored.	One of the conditions below, applied to a `filter_text` string value. • `startswith` • `endswith` • `contains` • `notcontain` • `equals` • `notequal`	`contains`
filter_text	A text string applied to the `filter_field` by a `filter_condition`.	Plain text	
ignore_case	Additional specifier applied to the case of the `filter_text`.	Either `true` or `false`.	`true`

Example showing a single filter set:

```
GET /api/(str:model_type)/(str:model_name)/?
  hierarchy=[hierarchy]
  &filter_field=[attribute_name]
  &filter_condition=startswith
  &filter_text=John
  &ignore_case=false
```

Note: For `relation/Subscriber`, the list of `filter_field` values are restricted to:

- `userid`

- `firstname`

- `lastname`

- `mailid`

- `hierarchy_friendly_name`

- `device`

- `extension_mobility`

- `phone`

If more than one filter set is used, all similar keys are grouped, so that the key position indicates the filter set. For example:

```
GET /api/(str:model_type)/(str:model_name)/?
  hierarchy=[hierarchy]
  &filter_field=[attribute_name]
  &filter_field=[attribute_name2]
  &filter_condition=startswith
  &filter_condition=endswith
  &filter_text=John
  &filter_text=an
  &ignore_case=false
  &ignore_case=false
```

The two filter sets in this example, are:

- `&filter_field=[attribute_name]`

- `&filter_condition=startswith`

- `&filter_text=John`

- `&ignore_case=false`

and

- `&filter_field=[attribute_name2]`

- `&filter_condition=endswith`

- `&filter_text=th`

- `&ignore_case=false`

Synchronous and Asynchronous

It is possible to submit mutator type operations with API parameters to complete synchronously, in which case the synchronous response to the transaction either includes the status of the transaction or a fault response. This is not recommended as long-running transactions or a busy system may exceed the HTTP timeout.

This is only available for models where the actions in the meta data contains `support_async`.

Key	Description	Value	Default
nowait	Controls the API synchronous or asynchronous behavior for requests resulting in transactions. Please refer to the `support_async` property in the model schema under **meta** -> **actions**, for an indication of support per action.	true, false	false

API version

To specify a specific API version of a resource, use the following parameter to an API request:

```
GET /api/(str:model_type)/(str:model_name)/?
  hierarchy=[hierarchy]&api_version=<version_number>
```

Key	Description	Value
api_version	Return the the resource with `api_version`. Use with the parameter `format=json`	supported version no.

Tags

To manage (add, remove) tags of a resource instance where the resource operations permissions allows tag management.

Key	Description	Value
tag	• Applies to resource instance (`<instance_pkid>`) • Uses `+tag` in URL • Resource operation enables Tag • API call is PATCH on resource instance	`<tag_value>` See below.

`<tag_value>` can be:

1. a tag name (no capital letters if tag should be searchable).

2. `__CLEAR_TAG__<tag_name>` to remove a tag `<tag_name>`.

3. `__CLEAR_ALL_TAGS__` to remove all tags.

```
PATCH /api/(str:model_type)/(str:model_name)/<instance_pkid>/+tag/?
  hierarchy=[hierarchy]
  &tag=<tag_value>
```

Note: More than one `tag` parameter may be used, for example `&tag=tag_one&tag=tag_two...`

Example JSON export of `meta` object of an instance showing:

- `tags`: "mytag", "another_tag"

- `version_tag`: "1.2"

```
"meta": {
          "tags": [
              "mytag",
              "another_tag"
          ],
          "pkid": "5ad5e4e3affa9343e4d9b140",
          "schema_version": "0.2.2",
          "hierarchy": "sys",
          "version_tag": "1.2",
          "model_type": "data/GeneralHelp"
      }
```

Version Tags

To manage (add, remove) tags of a resource instance where the resource operations permissions allows tag management.

Key	Description	Value
version_tag	• Applies to resource instance (`<instance_pkid>`) • Uses `+tag_version` in URL • Resource operation enables Version Tag • API call is PATCH on resource instance	`<version_tag_value>` for example `1.1`, `1.2` ..

```
PATCH /api/(str:model_type)/(str:model_name)/<instance_pkid>/+tag_version/?
  hierarchy=[hierarchy]
  &version_tag=<version_tag_value>
```

Example JSON export of `meta` object of an instance showing:

- `tags`: "mytag", "another_tag"

- `version_tag`: "1.2"

```
"meta": {
          "tags": [
              "mytag",
              "another_tag"
          ],
          "pkid": "5ad5e4e3affa9343e4d9b140",
          "schema_version": "0.2.2",
          "hierarchy": "sys",
          "version_tag": "1.2",
          "model_type": "data/GeneralHelp"
      }
```

Schema version, API version

Where the parameters below are added to the GET call:

```
&schema=true
&format=json
&schema_rules=true
```

then the JSON schema `meta` property will contain `schema_version` in accordance with `api_version=<version_number>`.

For Unified CM device models, the `schema_version` will match the version of the Unified CM that corresponds with the `api_version`, for example:

```
GET /api/device/cucm/(str:model_name)/?
  hierarchy=[hierarchy]&
  schema=true&
  format=json&
  schema_rules=true&
  api_version=10.1.2
```

Returns:

```
"meta": {
        "tags": [],
        "cached": true,
        "title": "",
        "business_key": {},
        "schema_version": 10.0,
```

The following table shows the current mapping for `/device/cucm` models:

api_version	schema_version
10.1.2	10.0
10.6.1	10.5
10.6.2	10.5
10.6.3	10.5
11.5.1	11.5
11.5.2	11.5
11.5.3	11.5

If no `api_version` is specified in the GET call, then the default `schema_version` is determined by the latest entry in the mapping table.

- For more details on API versioning, refer to the topic on API Backwards compatibility.

- For API performance best practices, refer to the Performance Best Practices Guide.

1.2.8. Filter Parameters for Choices

For the context in which the filter parameter are used, refer to the Choices topic on the `/choices/` endpoint.

Format:

```
GET http://<server_address>/api/<resource_type>/<resource_name>/choices/
  ?hierarchy=[hierarchy]
```

```
&format=json
&<filter_parameters>
```

Response data of the `/choices/` endpoint *without* filter parameters is a list of value-title pairs of the business keys. This can be modified with filter parameters.

Example without `<filter_parameters>`:

- Request

```
GET http://<server_address>/api/data/Countries/choices/
 ?hierarchy=[hierarchy]
 &format=json
```

- Response

```
HTTP 200 OK
Vary: Accept
X-Request-ID: 9bcd77b4cd27dccd0f18a1d8d22e7ddab85aa848
Content-Type: text/html; charset=utf-8
Allow: GET, HEAD, OPTIONS
Response-Content:
{

    pagination : {
        direction :  asc,
        maximum_limit :  2000,
        skip :  0,
        limit :  0,
        total_limit :  ,
        total :  37
    },
    meta : {
        query :  /api/data/Countries/choices/,
        references : [
            {
                pkid :  5a16c3c68963f91b84baf357,
                href :  /api/data/Countries/5a16c3c68963f91b84baf357/
            },
            ...
        ]
    },
    choices : [
        {
            value :  ["Australia", "AUS", "hcs"],
            title :  ["Australia", "AUS", "hcs"]
        },
        ...
```

Filter parameters available to modify the response:

- `field`: specifies the field in the business key to return as title and value, for example adding the parameter below

```
&field=iso_country_code
```

would return:

```
choices : [
    {
        value :  ["AUS"],
        title :  ["AUS"]
    },
    ...
```

- `choice_title`: specifies the field of the business key to be the `title` value, for example adding the parameter below

```
&field=iso_country_code
&choice_title=country_name
```

would return:

```
choices : [
    {
        value :  ["AUS"],
        title :  ["Australia"]
    },
    ...
```

- `title`: specifies the value of the `field` parameter to filter on, for example adding the parameter below

```
&field=iso_country_code
&title=BHR
&choice_title=country_name
```

would return:

```
choices : [
    {
        value :  ["BHR"],
        title :  ["Bahrain"]
    },
    ...
```

Note that the `title` parameter matches on the *start* of the value.

For an *exact* match, the `&filter_condition=equals` parameter can be added, for example:

```
&filter_condition=equals
&field=iso_country_code
&choice_title=country_name
&title=N
```

returns no value:

```
choices  []
```

Without `filter_condition=equals`, in other words, with just:

```
&field=iso_country_code
&choice_title=country_name
&title=N
```

returns:

```
choices": [
{"value": "NLD", "title": "Netherlands"},
{"value": "NZL", "title": "New Zealand"},]
```

1.2.9. API Request Headers

API Headers are available for:

1. Pagination of choices and macro results in an API call.

   The headers are `X-range` and `Range`, with the starting value as 0. These override and can be used instead of the `skip` and `limit` API parameters.

   For example, the following examples return the same results:

   ```
   GET /api/tool/Macro/?method=evaluate
        &hierarchy=[hierarchy]
        &input={{fn.lines}}
        &skip=0
        &limit=6



   GET /api/tool/Macro/?method=evaluate
        &hierarchy=[hierarchy]
        &input={{fn.lines}}

        Request headers:

        X-Range: items=0-5
        Range: items=0-5
   ```

   If the request is `items=0-199` (for 200 items) and there are more results, the response will show:

   ```
   Content-Range:items 0-199/999999999
   ```

   Since it is undetermined how many items there are, the value `999999999` represents the total.

   In this example, we have a total of 298 items. if a subsequent request is for the next 200 items (200-399), this includes the total. The response will then also show the total number of items (298) returned by the macro:

   ```
   Content-Range:items 200-399/298
   ```

2. Backward compatibility. The `X-Version` header is available to take an API version as value.

   For example:

   ```
   GET /api/data/Countries/?hierarchy=[hierarchy]
        &schema=true
        &format=json

        Request headers
        X-Version: 10.1.2
   ```

   Refer to the topics on API backwards compatibility.

1.2.10. Login and Authorization Tokens

The API includes as part of responses a `X-CSRFToken` response header that is set to the CSRF token, for example to `KEMzraBRygy2ZJ7fLuvbfKhAEIPK9D4s`. API clients should source the CSRF token from this header.

For background on CSRF, see:

- Cross-site request forgery
- Cross-Site Request Forgery (CSRF)

The API also includes as a part of responses a `csrftoken` cookie containing the CSRF token. This cookie is marked `httponly` and as such is not readable by browser-based client scripts. API clients should not try to source the CSRF token from this cookie.

The `X-CSRFToken` response header and `csrftoken` cookie values are identical.

When performing requests that require CSRF token validation, API clients should follow the general procedure:

1. Prior to performing the principal request, perform a request to the API and retrieve a CSRF token from the resulting response's `X-CSRFToken` response header. The CSRF token remains constant for the duration of a session, so clients could perform this request once per session (post authentication), storing the CSRF token and using it for subsequent requests.

   Clients should also retrieve the `csrftoken` cookie from the response.

2. For the primary request, include a `X-CSRFToken` request header containing the CSRF token as sourced from the response header, as well as the unchanged `csrftoken` cookie.

Note: Cookies must conform to https://tools.ietf.org/html/rfc6265

Concrete example for login:

```
GET http://localhost:8000/login/

Raw response headers:
Cache-Control: max-age=0
Connection: keep-alive
Content-Encoding: gzip
Content-Language: en-us
Content-Type: text/html; charset=utf-8
Date: Mon, 20 Apr 2015 09:18:47 GMT
Expires: Mon, 20 Apr 2015 09:18:47 GMT
Last-Modified: Mon, 20 Apr 2015 09:18:47 GMT
Server: nginx/1.4.6 (Ubuntu)
Set-Cookie: csrftoken=KEMzraBRygy2ZJ7fLuvbfKhAEIPK9D4s; httponly; Path=/
sessionid=5d1ccc96cbd7e7f290020aaedd64c1b3; httponly; Path=/
sso_login_url=; Path=/
Transfer-Encoding: chunked
Vary: Accept-Encoding, Cookie, Accept-Language, X-CSRFToken
X-CSRFToken: KEMzraBRygy2ZJ7fLuvbfKhAEIPK9D4s
```

1. Source the CSRF token from response's `X-CSRFToken` header.

2. Retain the CSRF cookie from response's `csrftoken` cookie.

3. Now perform the primary POST /login/ request to login, including the CSRF token as a `X-CSRFToken` request header as well as the unchanged `csrftoken` cookie:

```
POST http://localhost:8000/login/

Raw request headers:
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:37.0) Gecko/20100101␣
→Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8000/login/
Cookie: sessionid=5d1ccc96cbd7e7f290020aaedd64c1b3;␣
→csrftoken=KEMzraBRygy2ZJ7fLuvbfKhAEIPK9D4s; sso_login_url=
Connection: keep-alive
X-CSRFToken: KEMzraBRygy2ZJ7fLuvbfKhAEIPK9D4s
```

With for example payload as parameters:

```
&username=joe
&password=bloggs
&next=%2F
```

1.2.11. Non-interactive Login

The following request, parameter and endpoint is available on the API:

- Request:

```
POST <hostname>/noninteractivelogin/
```

- Payload:

    - `Content-Type: application/json`

    - JSON containing user credentials, for example:

```
{
 "username":"joebloggs@email.com",
 "password":"mysecret"
}
```

- Parameter:

    A request parameter to expose hierarchy and role related data is available: `rbacinfo`

With the user credentials payload as above, the following calls result in the same response:

```
POST <hostname>/noninteractivelogin/
POST <hostname>/noninteractivelogin/?rbacinfo=false
POST <hostname>/noninteractivelogin/?rbacinfo=False
```

If the request is successful:

- the HTTP response is 200

- the JSON body is for example:

```
{
 "is_externally_authenticated": false,
 "last_successful_login_time": "2017-06-12T13:28:55.785Z",
 "num_of_failed_login_attempts": 0
}
```

- the X-CSRFToken value

When enabling the `rbacinfo` parameter and with the same user credentials payload as above, the following calls result in the same response:

```
POST <hostname>/noninteractivelogin/?rbacinfo
POST <hostname>/noninteractivelogin/?rbacinfo=true
POST <hostname>/noninteractivelogin/?rbacinfo=True
POST <hostname>/noninteractivelogin/?rbacinfo=
```

If the request is successful:

- the HTTP response is 200

- the JSON body is for example:

```
{
 "hierarchy_path": "sys.Prov",
 "language": "en-us",
 "is_externally_authenticated": false,
 "hierarchy_name": "Prov",
 "hierarchy_href": "/api/data/HierarchyNode/593e8fa28719cf00060a7011/",
 "role_name": "ProvRole",
 "role_href": "/api/data/Role/593e91098719cf00060a7029/",
 "role_pkid": "593e91098719cf00060a7029",
 "last_successful_login_time": "2017-06-12T13:28:38.390Z",
 "hierarchy_type": "TestHierarchyNodeType",
 "hierarchy_pkid": "593e8fa28719cf00060a7011",
 "num_of_failed_login_attempts": 0
}
```

If a `data/PrivacyPolicy` instance is found at or above the logged in user's hierarchy, the data for the instance closest to that hierarchy will be included in the response JSON body:

```
{
 "privacy_policy": {
     "url": "<URL from data/PrivacyPolicy>",
     "name": "<Name from data/PrivacyPolicy>"
 },
 "hierarchy_path": "sys.Prov",
 "language": "en-us",
 ...
```

Note:

- Upon the first successful login, the `last_successful_login_time` is an empty string.

- Upon a subsequent successful login, the `last_successful_login_time` is the login time *prior* to current session.

- The `num_of_failed_login_attempts` value is reset to 0 after a successful login.

-

- If the requests above fail:

  - the HTTP response is 403

  - the JSON body is:

```
{
 "error_message": "Please enter a valid username and password.",
 "error_code": 27009
}
```

  - the X-CSRFToken value

1.2.12. Access Profiles

A logged in user is associated with an Access Profile that specifies access permissions to operations and models.

A user's Access Profile may not apply to models that are included or referenced in for example GUI Rules, Wizards or models that provide choices.

For example, when API calls are made to models that contain choices, such as:

```
GET api/data/DataSync/add/?schema_rules=&schema=&format=json
```

then any model GET calls that are carried out to provide the list of choices are shown with a generated `auth_token` that is required to provide access to these GET calls. This can be seen in the returned schema, for example, for the `target` call to show the choices available for `sync_order` in `data/DataSync` (`[hierarchy]` is substituted with the GET caller hierarchy ID.):

```
sync_order: {
target: "/api/data/ModelTypeList/choices/?hierarchy=[hierarchy]&
 field=name&format=json&
 auth_token=[auth_token]"
title: "Synchronization Order"
description: "The selected 'ordered' model type list that was created
as a model instance of the Model Type List. This list dictates the
order in which models will be synchronized. See: Model Type List."
format: "uri"
choices: [ ]
target_attr: "name"
target_model_type: "data/ModelTypeList"
type: "string"
```

This `auth_token` parameter is required to provide authorization to access the `data/ModelTypeList`, which may not be available in a user's Access Profile.

1.2.13. Time to Live (TTL)

For client applications that use session-based authentication upon initial login, an API endpoint that extends (if possible) and reports the session lifetime is available. This endpoint is typically used for client-side session management, for example to display a pop up to warn the user to extend the session before it expires (as in the case of for example self-service).

```
POST <hostname>/api/session/keep_alive/
```

The request returns a payload in JSON format with details:

- max_age: The number of seconds remaining for the session.

- expiry: The date at which the session will expire.

- extendable: Boolean indicating if the client can extend the length of the session by triggering an API request.

An example response:

```
{
  max_age: 86296
  extendable: false
  expiry: "2015-03-18T10:24:53.059Z"
}
```

1.2.14. Account Endpoint

The `<hostname>/account` endpoint provides addiotnal endpoints:

- `<hostname>/account/me/` : returns user details

- `<hostname>/account/password/` : allows for password management

Logged-in User Details

The following request and endpoint is available on the API to return logged-in user details:

- Request:

```
GET <hostname>/account/me/?format=json
```

If the request is successful:

- the HTTP response is 200

- the JSON body contains user account details, as shown in the example snippet below:

```
{
  "username": "CS-PAdmin",
  "menu_layout": {
    "pkid": "5c7daa2a7579050013878f83",
    "href": "/api/data/MenuLayout/5c7daa2a7579050013878f83/",
    "name": "HcsProviderMenu"
  },
  "language": "en-us",
  "landing_page": {
    "pkid": "5c7daa157579050013878d88",
    "href": "/api/data/LandingPage/5c7daa157579050013878d88/",
    "name": "HcsProviderLP"
  },
  "pkid": "5c7db7c5757905001387e6a1",
  "account_information": {
    "password_last_change_time": "2019-03-05T00:54:27.277Z",
```

```
    "last_login_time": "2019-03-05T08:01:11.184Z"
  },
  "hierarchy": {
    "pkid": "5c7db7b5757905001387e2d6",
    "node_type": "Provider",
    "href": "/api/data/HierarchyNode/5c7db7b5757905001387e2d6/",
    "name": "CS-P",
    "hierarchy_path": "sys.hcs.CS-P"
  },
  "theme": {
    "pkid": "5c7db13d757905001387c33b",
    "href": "/api/data/Theme/5c7db13d757905001387c33b/",
    "name": "default"
  },
  "role": {

...
```

Password Change

An API endpoint is available to request the details needed for a user password change and to submit a password change.

To get details of the POST request and the JSON schema of the payload to change the password, use the request:

```
GET https://hostname/account/password/change?hierarchy=[hierarchy]&format=json
```

To change a user password, the request will then be of the format:

```
POST https://hostname/account/password/change?hierarchy=[hierarchy]&format=json
```

The payload is in JSON format and contains user details, old password and new password.

A successful password change request returns a response of the format:

```
{"meta": {
    "uri": "/account/password/change/"
    },
    "success": true
}
```

The request format if a user changes their own password on the GUI, payload parameters include the token, for example:

```
csrfmiddlewaretoken=am9lOmJsb2dncw==
```

In this instance, the `user_pkid` a part of the payload, as it is hidden in the GUI.

For a successful password change from the GUI, the user's browser client is redirected to the endpoint:

```
https://hostname/account/password/change/done/
```

This presents the user with a message and request to log in with the new password.

1.3. Anatomy of an API Response

1.3.1. API Response Overview

Below are the typical elements of an API response:

- header - API header.

- meta - Meta data.

- data - Actual data contained in the model as name:value pairs.

- schema - Schema describing the structure of the data of the resource, in particular the data types of the names in the name:value pairs in the data.

- resources - An object grouping a list of single resource's meta and data objects in an API list response

- pagination - an object containing pagination data in an API list response

Not all the elements above exist in each response. These differ depending on request parameters and whether response contains a list of resource or a single resource.

1.3.2. API Response Header

The following is a header data example of an API response from an API request not using Basic Auth:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 09 Oct 2017 09:57:54 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
X-CSRFToken: MnPzYbeIVKcSleCsmHWNUapz3igZ79iy
X-Request-ID: 137fcf465ed6f77880f2739e0c50ad88d8e45073
Content-Language: en-us
Vary: Accept, Cookie, Accept-Language, X-CSRFToken
Allow: GET, POST, HEAD, OPTIONS
X-Session: {"max_age": 1200,
 "extendable": true,
 "expiry": "2017-10-09T10:17:54.542300+00:00"}
Set-Cookie: csrftoken=MnPzYbeItKcSyyysmHWyyypz3igZ79iy;
 httponly;
 Path=/
Set-Cookie: sessionid=q150dg1ctpgc1sza3ktggyguo4nsbg5u;
 httponly;
 Path=/
Content-Security-Policy:
   style-src 'unsafe-inline' 'self';
   script-src 'unsafe-eval' 'self';
```

- The `X-Session` header entry has the following properties:

  - `max_age`: The number of seconds remaining for the session.

  - `expiry`: The date at which the session will expire.

– `extendable`: Boolean indicating if the client can extend the length of the session by triggering an API request.

This information is also available from a POST call to the following endpoint:

```
POST <hostname>/api/session/keep_alive/
```

An example response JSON payload:

```
{
  max_age: 86296
  extendable: false
  expiry: "2015-03-18T10:24:53.059Z"
}
```

Refer to the section: Time to Live (TTL)

Note: This header is not present in responses from Basic Auth API requests.

1.3.3. Single Resource Response

A single resource response outline is as follows:

```
{
  "meta": {
    ...
  },
  "data": {
    ...
  },
  "schema": {
    ...
  }
}
```

The *schema* object is only returned for a single resource request when the *schema* request parameter is added to the request. Please see Response Elements

1.3.4. Resource List Response

The response object outline is as follows:

```
{
    "pagination": {
        ...
    },
    "meta": {
        ...
    },
    "resources": [{
        "meta": {
            ...
```

(continues on next page)

```
        },
        "data": {
            ...
        }
    },
    {
        "meta": {
            ...
        },
        "data": {
            ...
        }
    }]
}
```

1.3.5. POST/PUT/DELETE/PATCH Response

Support for synchronous and asynchronous request resulting in transactions, is controlled by the `nowait` parameter in the request URL. The support for asynchronous request handling is indicated in the API schema structure **actions** with the `support_async` property.

The outline of the default synchronous transaction response of mutator transactions when the API parameter `nowait` is set to be false, is as follows:

```
{
  "pkid": "51f7e09bd0278d4b28e981da",
  "model_type": "data/CallManager",
  "meta": {
    "parent_id": {
      "pkid": "51f7d06ad0278d4b34e98134",
      "uri": "/api/data/HierarchyNode/51f7d06ad0278d4b34e98134"
    },
    "summary_attrs": [
      {
        "name": "description",
        "title": "Description"
      },
      {
        "name": "host",
        "title": "Host Name"
      },
      {
        "name": "port",
        "title": "Port"
      }
    ],
    "uri": "/api/data/CallManager/51f7e09bd0278d4b28e981da"
  },
  "success": true
}
```

The outline of the synchronous response to asynchronous mutator transactions when the API parameter `nowait` is set to be true, is as follows:

```
{
  "href": "/api/tool/Transaction/cfe8a8fd-98e6-4290-b0c3-2dfa2224b808",
  "success": true,
  "transaction_id": "cfe8a8fd-98e6-4290-b0c3-2dfa2224b808"
}
```

To retrieve (for example by polling) the transaction status of any mutator transactions, use the *transaction_id* in the synchronous response to the asynchronous mutator transaction as follows:

```
GET /api/tool/Transaction/cfe8a8fd-98e6-4290-b0c3-2dfa2224b808
```

The response contains the status ad replay action URL, for example:

```
{
 "meta": {
        "model_type": "tool/Transaction",
        "summary_attrs":
        {
         "name": "name",
         "title": "Name"
        },
        "references": {}
        "actions":
          {
           "replay": {
             "class": "execute",
             "href": "/api/tool/Transaction/cfe8a8fd-98e6-4290-b0c3-2dfa2224b808/
               replay?format=json",
             "method": "GET",
             "title": "Replay"
           }
          }
        }
 "data": {
        "status": "Completed",
        "username": "sysadmin",
        "resource": {
         "hierarchy: "sys",
         "after_transaction": "/api/data/GeneralHelp/5268c7d3a616540a766b91f5/?
           cached=5268f2eba616540a736b926c Entity",
         "current_state": "/api/data/GeneralHelp/5268c7d3a616540a766b91f5/ Entity",
         "before_transaction": "/api/data/GeneralHelp/5268c7d3a616540a766b91f5/
           ?cached=5268c7d3a616540a766b91f7 Entity",
         "pkid": "5268c7d3a616540a766b91f5",
         "model_type": "data/GeneralHelp",
        }
  [...]
```

This mechanism can be used to retrieve the transaction status of any transaction or its sub-transaction, using the pkid of the (sub) transaction.

For the View model, the GET call to `tool/Transaction/[trans pkid]` shows the View resource has no instance pkid, because a view model stores no instances.

1.3.6. Asynchronous Mutator Transaction Status Callback

When using the API parameter `nowait=true`, the service requester can submit optional request meta data - containing a callback URL - with any mutator request by appending the `request_meta` tag to the normal payload of the request.

In order to receive asynchronous transaction status notifications, the requesting system needs to publish an HTTP service to service requests made by the callback URL. An example of a simple http service is provided in a separate section.

The callback operation supports an optional username and password that VOSS-4-UC uses to perform HTTP basic authentication on requests made to the callback service. The optional elements `external_id` and `external_reference` are explained in the section on correlation identifiers.

```
{
    <Actual request data goes here>,
    "request_meta": {
        "external_id": "3x4mpl3-3xtern4l-FF",
        "external_reference": "Example External Reference-FF",
        "callback_url": "http://my.callbackservice:8080",
        "callback_username": "username",
        "callback_password": "password"
    }
}
```

The following details should be noted here:

- The schema of system resources or system tools do not include reference to the request meta data in the schema definition of each resource in the system.

- The `<Actual request data goes here>` request data needed to for example add a `country_name` instance for data/Countries would be similar to: `"country_name": "South Africa"`.

- The request data for deleting two countries for example would be

```
"hrefs":[
    "/api/data/Countries/534fdf190dd19012066433ce",
    "/api/data/Countries/534fda1d0dd1901206643397"
]
```

- For the callback service to function, the callback service needs to be accessible from the fulfillment server.

Upon completion of the asynchronous mutator transaction posted with a callback URL, VOSS-4-UC POSTs an HTTP request (asynchronous transaction status callback) to the callback service specified by the callback URL. The callback service needs to respond with a HTTP 200 ACK *before* internal processing of the callback. The callback includes the transaction ID sent to the requesting system as part of the synchronous response. To correlate the asynchronous transaction status callbacks with the original request, the requesting system would need to record the `transaction_id` returned in the synchronous response.

The HTTP headers and the payload of the asynchronous transaction status callback includes the following information:

HTTP headers:

```
{
    'accept-encoding': 'identity',
```

(continues on next page)

```
        'authorization': 'BasicdXNlcm5hbWU6cGFzc3dvcmQ=',
        'content-length': '275',
        'content-type': 'application/json',
        'host': 'localhost: 8080'
}
```

Payload:

```
{
    "external_id": "3x4mpl3-3xtern4l-FF",
    "external_reference": "ExampleExternalReference",
    "status": "Success",
    "transaction": {
        "href":
          "http: //my.fulfillmentserver/api/tool/
             Transaction/e6ac7c1e-c63a-11e3-9af5-08002791605b/",
        "id": "e6ac7c1e-c63a-11e3-9af5-08002791605b"
    }
}
```

The following details should be noted here:

- Correlation identifiers (see correlation identifiers) are included in the payload if they are present.

- The status of the transaction is as in the transaction log: Fail or Success.

The transaction status in VOSS-4-UC is not affected by the response of the HTTP service published by the requesting system. The transaction log information includes the callback request and the response returned by the callback service published by the external system.

For transactions with multiple sub-transactions, a single transaction status callback request is made upon the completion of the parent transaction. Transaction status callbacks are not supported for the parent transactions tool/BulkLoad and tool/DataImport.

In the event that the transaction status callback is not received by the external system due to for example a network outage, the external system can poll to retrieve the transaction status. For example:

```
GET /api/tool/Transaction/e6ac7c1e-c63a-11e3-9af5-08002791605b
```

Callbacks for failing transactions include error data as part of the callback body/payload. For example:

```
{u'authorization': 'Basic dXNlcm5hbWU6cGFzc3dvcmQ=',
 u'error': {u'code': 4001,
            u'http_code': 400,
            u'message': u'Error, Duplicate Resource Found.
             data/CallbackDataModel already exists with the following
               unique data - (name = "CallbackDataModel Name 2")'},
 u'external_id': u'3x4mpl3-3xt3rn4l-7d',
 u'external_reference': u'External Ref',
 u'resource': {u'hierarchy': u'542a7347c952703e3646a4c5',
               u'model_type': u'data/CallbackDataModel',
               u'pkid': u'542a7357c952703e3646a4da'},
 u'status': u'Fail',
 u'transaction': {u'href':
  u'http://django.testserver/api/tool/Transaction/844344ee-4881-11e4-a4f9-0800279e955b/
↪',
                   u'id': u'844344ee-4881-11e4-a4f9-0800279e955b'}}
```

The error data as show in the example, includes:

- the exception code: 4001

- http error code: 400

- error message:

```
'Error, Duplicate Resource Found. data/CallbackDataModel already
exists with the following unique data – (name = "CallbackDataModel Name 2")'
```

This is the same error message structure as returned by the API for failing requests.

1.3.7. Example of an Asynchronous Mutator Transaction with nowait=true

Request:

```
POST http://172.29.232.238/api/data/Countries/
 ?hierarchy=[hierarchy]
 &nowait=true
Payload of the request:

{'country_name': 'Callback Created Example Country Name',
 'request_meta': {'callback_password': 'password',
                  'callback_url': 'http://localhost:9365',
                  'callback_username': 'username',
                  'external_id': '3x4mpl3-3xt3rn4l-7d',
                  'external_reference': 'External Ref'}}
```

Synchronous response:

```
{
href: "/api/tool/Transaction/e6ac7c1e-c63a-11e3-9af5-08002791605b"
success: true
transaction_id: "e6ac7c1e-c63a-11e3-9af5-08002791605b"
}
HTTP 202 ACCEPTED
```

Asynchronous transaction status callback (console output of the simple http service provided in the separate example section):

```
POST – 2014-04-17 16:16:43.737509


Headers:

{'accept-encoding': 'identity',
 'authorization': 'Basic dXNlcm5hbWU6cGFzc3dvcmQ=',
 'content-length': '275',
 'content-type': 'application/json',
 'host': 'localhost:8080'}


Raw Callback Body:

'{"status": "Fail", "transaction":
{"href":
   "http://django.testserver/api/tool/Transaction/
```

```
     34866060-fd47-11e3-88dd-080027880ca6/",
 "id": "34866060-fd47-11e3-88dd-080027880ca6"},
 "resource": {"hierarchy": "1c0efge2c0deab10da595101",
              "model_type": "data/Countries",
              "pkid": "53ac3d41c9527062809c0021"},
              "external_reference": "External Ref",
              "external_id": "3x4mpl3-3xt3rn4l-7d"}'

Pretty Callback Body:

{u'external_id': u'3x4mpl3-3xt3rn4l-7d',
 u'external_reference': u'External Ref',
 u'resource': {u'hierarchy': u'1c0efge2c0deab10da595101',
               u'model_type': u'data/Countries',
               u'pkid': u'53ac3d41c9527062809c0021'},
 u'status': u'Fail',
 u'transaction': {u'href':
   u'http://django.testserver/api/tool/Transaction/
    34866060-fd47-11e3-88dd-080027880ca6/',
                  u'id': u'34866060-fd47-11e3-88dd-080027880ca6'}}

localhost - - [17/Apr/2014 16:16:43] "POST / HTTP/1.1" 200 -
```

1.3.8. Correlation Identifiers

In order to allow an external system use its own identifiers to cross-reference transactions in the system, the API supports two external identifiers for all transactions. This allows the external system to:

1. Tie together multiple transactions in the system (using for example an order number)

2. Track individual requests in the system using the external IDs.

External identifiers are not supported for the parent transactions tool/BulkLoad and tool/DataImport.

The transaction log will include these two IDs and the transaction log, as shown below.

You can obtain the details of the parent transaction with a given ID by using the following API call:

```
GET http://my.fulfillmentserver/api/tool/Transaction/
    ?hierarchy=[hierarchy]&
     filter_condition=contains&
     format=json&
     filter_text=3x4mpl3-3xtern4l-FF&
     filter_field=external.id
```

You can obtain the details of transactions tied together using an external reference number using the following API call:

```
GET http://my.fulfillmentserver/api/tool/Transaction/
 ?hierarchy=[hierarchy]&
    filter_condition=contains&
    format=json&
    filter_text=Example%20External%20Reference-FF&
    filter_field=external.reference
```

Transaction ID	1013
Detail	Delete Multiple Resources
Username	sysadmin
Action	Delete Bulk Delete
Status	Success
Rolled Back	No
External Id	3x4mpl3-3xtern4l-FF
External Reference	Example External Reference-FF
Submitted Time	Apr 17, 2014 16:16:43 South Africa Standard Time
Started Time	Apr 17, 2014 16:16:43 South Africa Standard Time
Completed Time	Apr 17, 2014 16:16:43 South Africa Standard Time
Duration	0.521 sec

.

Sub Transactions

Action	Status	Transaction	Submitted Time	Detail	
Delete Resource	Success	Link	Apr 17, 2014 16:16:43 South Africa Standard Time	Resource Delete	
Delete Resource	Success	Link	Apr 17, 2014 16:16:43 South Africa Standard Time	Resource Delete	

1 - 2 of 2.   Items/Page: 10 ▾

Log

Time	Severity	Message	Duration
Apr 17, 2014 16:16:43 South Africa Standard Time	info	[send] [Request] API Call Success [200] [http://localhost:808	0.003362

[send] [Request] API Call Success [200] [http://localhost:8080] CaseInsensitiveDict({u'Content-Type': u'application/json'})

REQUEST:

{"status": " Success ",   "transaction": {"href": "http://172.29.232.238/v0/tool/Transaction/e6ac7c1e-c63a-11e3-9af5-08002791605b/",

            "id": "e6ac7c1e-c63a-11e3-9af5-08002791605b"}, "external_id": "3x4mpl3-3xtern4l-FF",

        "external_reference": "Example External Reference-FF"}

RESPONSE:

1.3.9. Example Of A Simple HTTP Server

The following code is an example of a simple HTTP server that can be used to test basic async transaction status callback operations. The code is not intended for actual use.

Note that the HTTP 200 ACK is sent asyncronously *before* internal processing of the callback.

```python
#!/usr/bin/env python
from datetime import datetime
import SimpleHTTPServer
import SocketServer
import logging
import cgi
import json
from pprint import pprint
PORT = 8080


class ServerHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):

    def do_GET(self):
        SimpleHTTPServer.SimpleHTTPRequestHandler.do_GET(self)

    def do_POST(self):
        self.send_response(200)
        self.wfile.write("ACK")

        # Insert internal processing here.
        # Below is an example of internal processing that simply prints out the
        # callback request.
        print "\nPOST - {}".format(datetime.now())
        print "Headers:"
        pprint(dict(self.headers))
        print "\nRaw Body:"
        body = self.rfile.read(int(self.headers['Content-Length'])).decode('utf-8')
        pprint(body)
        print "\nPretty Body:"
        pprint(json.loads(body))


Handler = ServerHandler

httpd = SocketServer.TCPServer(("", PORT), Handler)

print "Serving at port", PORT
httpd.serve_forever()
```

2 Using the API

2.1. Developer Guidelines

The following practices are recommended to all developers. The aim is to reduce the number and extent of any updates that may be necessary.

1. The order of elements within the interface data and messages may change, within the constraints of the interface specification. Developers should avoid unnecessary dependence on the order of elements to interpret information exchanged with VOSS-4-UC.

2. New interface methods, operations, actions, requests, responses, headers, parameters, attributes, other elements, or new values of existing elements, may be introduced into the VOSS-4-UC interfaces. Developers should disregard or provide generic treatments where necessary for any unknown elements or unknown values of known elements encountered.

3. Notifications, operations, methods, actions, requests, responses, headers, parameters, attributes, and other elements from previous versions of VOSS-4-UC interfaces, will remain, and will maintain their previous meaning and behavior to the extent possible and consistent with the need to correct defects.

4. Applications should not be dependent on interface behavior resulting from defects (behavior not consistent with published interface specifications), since the behavior can change when defects are fixed.

5. The use of deprecated methods, operations, actions, handlers, requests, responses, headers, parameters, attributes, or other elements should be removed from applications as soon as possible to avoid issues when those deprecated items are removed from VOSS-4-UC or its interfaces.

6. Application Developers should be aware that not all new features and new supported devices (for example, phones) will be forward compatible. New features and devices may require application modifications to be compatible or to make use of the new features or devices.

2.2. Workflow Tasks

2.2.1. Procedure

1. Log in with the username "hcsadmin@sys.hcs", using the password that was set during the installation.

2. Get the Provider Name & Provider PKID using the data/HierarchyNode API in the url, where "hierarchy" = "sys.hcs".

3. Note:

For all POST/PUT/DELETE operations to be asynchronous transactions, set the query param "nowait=true" in the URI.

To create a provider Admin use the relation/User API in the url, with the hierarchy value that you receive in the get call of Step-1 (PKID or the dot notation).

4. Note:

   Creating a Reseller is not mandatory, and it depends on the structure of provisioning. A Reseller must be created if the tree structure of the provisioning is: Provider -> Reseller -> Customer -> Site.

   To create a Reseller use the relation/HcsResellerREL API, with the Provider hierarchy of the API of Step-1.

5. To create a Reseller Admin use the relation/User API in the url, with the hierarchy value of the Reseller (PKID or the dot notation).

6. To create a Shared Network Device (Cisco Unified Communications Manager or Unity), it needs to be done either at either the Provider Hierarchy or the Reseller Hierarchy Level. Use the following APIs for each of the devices listed below:

Device	API
Cisco Unified Communications Manager	relation/HcsCallManagerREL
Unity	relation/HcsUnityConnectionREL
Presence	relation/HcsPresenceREL
WebEx	relation/HcsWebExREL

7. Note:

   The Customer is directly under Provider if the deployment structure is Provider -> Customer -> Site or under the Reseller if the deployment structure is Provider ->Reseller ->Customer -> Site.

   To create a Customer use the relation/HcsCustomerREL API, with the hierarchy of provider/reseller that can be retrieved using the respective API.

8. To create a Customer Admin use the relation/User API in the url, with the hierarchy value of the Customer(PKID or the dot notation).

9. Note:

   If the Customer is using "shared_uc_apps" then, you cannot add dedicated devices for that customer.

   Adding a Dedicated Network Device for a Customer(Cisco Unified Communications Manager or Unity) needs to be done at the Customer hierarchy Level. Use the following APIs for each of the devices listed below:

Device	API
Cisco Unified Communications Manager	relation/HcsCallManagerREL
Unity	relation/HcsUnityConnectionREL
Presence	relation/HcsPresenceREL
WebEx	relation/HcsWebExREL

10. Once the devices are configured a Network Device List (NDL) needs to be configured for the Customer. To create an NDL use the relation/HcsNetworkDeviceListREL API. At least one device is required to add an NDL.

11. Note:

    It is not mandatory to have a Network Device List (NDL) to create a Site. However, an NDL is needed to add a Subscriber or Phone or Lines to a Site. Sites created without an NDL can later be able associated to one.

    To create a site use the relation/HcsSiteREL API.

12. Note:

    Using the relation/User API will only be local Cisco Unified Communications Domain Manager admin. To add an Admin who is also a Subscriber, use the relation/User API, which can later be moved to any Cisco Unified Communications Manager that is associated with the Site.

    To create a Site Admin use the relation/User API in the url, with the hierarchy value of the Site (PKID or the dot notation).

13. Complete the following activities at the Site level:

    a. To create a Subscriber use the relation/Subscriber API.

    b. To create a Line use the view/HcsDNMgmtVIEW API.

    c. To create a Phone use the relation/SubscriberPhone API.

    d. To create a Voicemail use the relation/Voicemail API.

2.3. Developer Tools

The Developer tools that are available in Mozilla Firefox and Google Chrome allow all the actions exposed by each API to be displayed as they are being called in the GUI. This gives us the opportunity to view the request and response actions as they occur, and provide the details of what each API provides and its relationship to the GUI.

With Developer tools enabled, the network tab of the Developer tools show the information that is contained in each request and response as you navigate and use the GUI. This allows service providers a direct view as to what data each API requires.

3 Handling API Fault Responses

3.1. Fault Responses

To interpret the HTTP fault responses codes and the response_code within the data element of a API response for a faulty request, refer to the list of possible error codes.

3.2. Error Messages

The tables below provide:

- an error code range reference
- message details of the error codes

To inspect application log messages from the command line, set the debug level on and view the app log. Refer to the Platform Guide for more details.

```
voss set_debug 1
log view voss-deviceapi/app.log
```

The message strings are shown in their template format: references to specific properties are shown as placeholders that are represented by {} .

Note:

- For AuthError codes, the following rules apply:
  - For API version 11.5.3 and below, only the **AuthError_11_5_3** table messages apply.
  - For API greater than 11.5.3, **AuthError** table messages override the corresponding **AuthError_11_5_3** table messages, while the unchanged **AuthError_11_5_3** table messages still apply.

RuleError	Message	HTTP Code
15000	Invalid hierarchy for this operation. Please select new hierarchy.	449
15001	Multiple devices found at this Hierarchy level. Please select device.	449
15002	Multiple network device lists (NDL) found at this Hierarchy. Please select a NDL.	449
15003	Network device list reference (NDLR) not found at this Hierarchy.	449
15004	Network device list (NDL) with pkid [{}] not found in available list. Please check NDL rule at the Hierarchy	400
15005	No network device lists (NDL) found at this Hierarchy.	449
15999	Error, (UNHANDLED_ERROR)	400

TransactionError	Message	HTTP Code
23000	Unable to determine Transaction ID.	400
23001	Transaction must be registered with valid user details.	400
23002	Transaction not found.	404
23003	Transaction must be viewed with valid user details.	400
23004	{} (MAX_INSTANCES_EXCEEDED)	400
23005	Invalid Transaction State: {}	400
23006	Transaction canceled.	400
23007	Transaction must be registered with the hierarchy in which it is executing.	400
23008	Transaction must be registered with model_type if pkid is provided.	400
23010	The current filter caused a long running request. Please add more filter fields, use Case Sensitive or change the criteria types to one of {}.	400
23011	Invalid choices field [{}].	400
23012	The [{0}] condition on field [{1}], is not allowed.	400
23013	Invalid start and end date range provided in filter.	400
23014	Invalid start and end ID range provided in filter.	400
23015	Invalid ID value in filter	400
23999	Error, {} (UNHANDLED_ERROR)	400

ListUtilError	Message	HTTP Code
20000	Invalid query dictionary, expected 1 key!	400
20999	Error, (UNHANDLED_ERROR)	400

AllError	Message	HTTP Code
999999	All Error	400

ForeignKeyError	Message	HTTP Code
24000	Could not resolve foreign key to {model_type} with "{attr_name}: {attr_value}".	400
24999	Error, {} (UNHANDLED_ERROR)	400

ChoicesError	Message	HTTP Code
26000	Instance context for choices not valid, instance: {instance}	400
26999	Error, {} (UNHANDLED_ERROR)	400

CnfError	Message	HTTP Code
40000	Device change notifications are not supported for device {}.	400
40001	Device change notification data for device {} has been lost. Tracking data has been repaired and collector process will continue. Some changes may have been lost, please run a full sync on the device.	400
40002	Device change notification tracking data for device {} has become corrupted. Tracking data has been repaired and collector process will continue. Some changes may have been lost, please run a full sync on the device.	400
40003	Device change notification tracking DB write for device {} failed. The collector process will continue to attempt DB writes. Please investigate the database write failure. {}	400
40004	Device change notification data DB write for device {} failed. The collector process will continue to attempt DB writes. Please investigate the database write failure. {}	400
40005	Unable to repair device change notification tracking data for device {}. {}	400
40006	Too many unprocessed changes recorded for device {}. No new changes will be recorded until at least {} changes are processed. Please configure and run the necessary data syncs.	400
40008	Could not update pending changes data for device {}. {}.	400
40010	Unable to clear device change notifications for device {}. {}.	400

PackageError	Message	HTTP Code
17000	Unable to load package. Package ({}) depends on ({}) but it does not exist.	400
17001	Unable to load package. Package ({}) requires ({} {}) but {} is currently loaded.	400
17999	Unable to load package. {}	400

CascadeDeleteError	Message	HTTP Code
13000	Hierarchy path or pkid required	400
13001	Could not delete {} out of {} resources.	400
13002	Could not move the following resources that failed to delete: {}.	400
13999	Error, (UNHANDLED_ERROR)	400

WebExError	Message	HTTP Code
31000	[{}] Site Name or Site ID must at least be specified	400

CertificateError	Message	HTTP Code
25001	Certificate request cannot be exported while "Generate Certificate Signing Request" is not set.	400
25002	Certificate can only be imported when "Generate Certificate Signing Request" is set.	400
25003	Certificate upload failed.	400
25004	Uploaded file is not a certificate in .pem format.	400
25999	Error, {} (UNHANDLED_ERROR)	400

FileUploadError	Message	HTTP Code
39000	Can not determine supported file extensions.	400
39001	'{}' does not have a valid file extension.	400
39002	File is too large. Maximum permitted file size is {} bytes.	400

BulkLoadError	Message	HTTP Code
10000	File Upload Error for File Name : ({})	400
10001	File Encoding Error : ({})	400
10002	Only valid Excel xlsx files are accepted	400
10003	General Error; ({})	400
10004	{success} out of {total} items loaded successfully.	400
10005	Resource data was not found in worksheet '{worksheet}'.	400
10006	Both parallel and serial are not allowed in '{worksheet}'.	400
10007	Differing parallel_transaction_limit values are not allowed in '{worksheet}'.	400
10008	Invalid value of '{limit}' for parallel_transaction_limit header in '{worksheet}', should be left blank or a number between 1 and 100(inclusive).	400

Continued on next page

Table 1 -- continued from previous page

BulkLoadError	Message	HTTP Code
10010	Data does not conform to schema; ({})	400
10011	Hierarchy not specified for row with data; ({})	400
10012	'{user}' is not permitted access to resources at '{hierarchy}'.	403
10020	Hierarchy '{hierarchy}' was not found.	400
10021	Action '{action}' not allowed.	400
10022	Action '{action}' not allowed for model '{model}'.	400
10030	User '{username}' is not allowed to {operation} {model_type}.	403
10040	Fields do not exist in {model}: {fields}.	400
10041	No search fields specified in row.	400
10042	More than one resource found. Search fields '{search}'.	400
10043	Resource not found. Search fields '{search}'.	400
10044	Malformed search fields: {fields}.	400
10045	Malformed fields{message}: {fields}.	400
10046	Can not find meta actions for specified resource instance.	400
10047	Malformed entity header '{header}' in cell '{cell}' worksheet '{sheet}'.	400
10050	Can not enforce data type '{data_type}' on '{data}'. Row data: {row_data}	400
10051	An internal error occurred while processing workbook '{filename}'{note}	400
10052	The specified meta_prefix '{meta_prefix}' in sheet '{sheet_name}' is invalid.	400
10053	The specified meta_prefix '{meta_prefix}' in sheet '{sheet_name}' was not found in base headers.	400
10054	The following base headers '{headers}' in '{sheet_name}' are prefixed, but meta_prefix is not specified.	400
10061	No match for device '{device}'.	400
10062	XLSX File Error: ({})	400

CnfWarning	Message	HTTP Code
45000	Unprocessed changes at 75%% of limit for device {}. Please configure and run the necessary data syncs.	400

DataSyncError	Message	HTTP Code
29000	Could not find user executing data sync operation.	500
29001	User [{}] does not have {} {} permissions.	403
29002	Could not establish a test connection to the device. Verify that your device connection details are correct.	400
29003	Aborting operation. Reason: {}	400
29004	{} (CRITICAL_SUBTRANSCATION_ERROR)	400
29005	Auth Error while testing connection to device	400
29999	Error, {} (UNHANDLED_ERROR)	500

WorkflowError	Message	HTTP Code
7000	Workflow not found	400
7001	Maximum workflow recursion depth exceeded	400
7002	Invalid workflow script identifier {}	400
7003	Specified workflow script name {} not found	400
7004	Error looking up workflow script names against API	400
7005	Invalid workflow action	400
7006	{} (FAILED)	400
7007	Advanced Find Options invalid - Resource not found with options {}	400
7008	{} (CONDITION_CONSTRAINT)	400
7009	Advanced Find Options invalid - More than one resource found with options {}	400
7010	Network Device List {} does not contain an entry for type {}	400
7011	Workflow operation Sync not supported for type {}	400
7012	No target device found for Workflow Sync operation	400
7999	Unexpected error occurred.	400

ExpectError	Message	HTTP Code
35000	The expect binary is not present in the path on the server	500
35001	There was an error executing the expect script : {}	500

ResourceError	Message	HTTP Code
4000	Error, Cannot delete Hierarchy until all resources under it are removed	400
4001	Error, Duplicate Resource Found. {}	400
4002	Resource Not Found {}	404

Continued on next page

Table 2 -- continued from previous page

ResourceError	Message	HTTP Code
4003	Failed to save {}. {}	400
4004	Failed to save {}. {}	400
4005	Model Type cannot be None when adding a new Resource	400
4006	Resource Parent {} not found	400
4007	Resource Meta structure corrupt for {}	400
4008	Cannot create a Resource without a Parent Hierarchy	400
4009	Failed to save {}. {}	400
4010	Cannot find Resource relation {}	400
4011	Cannot find target device for model type {} in current hierarchy context	400
4012	Cannot find summary attr [{}] in schema root	400
4013	Cannot perform operation, model {} already has one or more instances	400
4014	Cannot perform operation, resource is part of domain model {}	400
4015	Resource Meta structure corrupt. {}	400
4016	Badly-formed schema; "properties" missing for data type "object"	400
4017	Cannot perform operation, model {} is already referenced by one or more resources: {}	400
4018	Failed to execute {}. {}	400
4019	One or more errors occurred during import	400
4020	Transaction resource failed with errors {}	400
4021	Resources are not of the same type	400
4022	Model type for Resources not found	400
4023	Cannot move Hierarchy Node {} to {}	400
4024	Resource move failed with error {}	400
4025	Invalid business key {}, expected {}	400
4026	Cascade delete failed with error {}	400
4027	Invalid business key for import. Did not expect path, found {}.	400
4028	Resource move failed, Device at source hierarchy [{}] is different from the target hierarchy [{}]	400
4029	Resource [{}] cannot be accessed by user [{}]	403
4030	Cannot perform operation. Hierarchy Node Type [{}] is reserved.	400
4031	Search index is not up to date. Please notify your administrator before proceeding	400
4032	Attempting to create hierarchy node '{}' is not permitted.	403
4033	Could not update reference cache, from: {}, reference: {}, error: {}	403
4034	Resource move failed, hierarchy [{}] of type [{}] does not contain an NDLR	400
4999	Unhandled Resource Error	400

MacroError	Message	HTTP Code
6000	Template must be a dictionary - got {}	400
6001	No hierarchy supplied	400
6002	Invalid macro specified: {}	400
6003	Macro lookup of {} failed at hierarchy {}	400
6004	Macro lookup of {} returned multiple values {} at hierarchy {}	400
6005	Macro lookup of {} failed when fetching from {} at hierarchy {}	400
6006	Macro lookup failed for field {} in context {}	400
6007	Macro lookup failed for field {} in context {}, type str or int expected not type dict {}	400
6008	Macro function {} not found	400
6009	Macro function arguments error - {}	400
6010	Macro function error - {}	400
6011	Unexpected business key format - {}	400
6012	Conditional Logic error occurred - {}	400
6013	Custom Macro function {} not found	400
6014	Custom Macro function {} not secure or contains invalid strings	400
6015	Could not parse the WhereClause Error:{} WhereClause:{} Please check quotation	400
6016	Lookup field {} not supported/permitted.	400
6017	Filter field: {} not in fields: {}.	400
6018	Incorrect hierarchy direction, {}. Allowed: {}.	400
6019	Error in macro function '{}' - {}	400
6999	Error, (UNHANDLED_ERROR)	400

InternalError	Message	HTTP Code
1000	Cannot import Python model name {}	404
1001	Python Type error	400
1002	{} must be an integer	400
1003	Improperly configured settings, {}	400

GraphLookupError	Message	HTTP Code
37000	Cannot perform operation, Resource with pkid [{}] cannot be accessed.	403

AuthError	Message	HTTP Code
27000	{} (INCORRECT_PASSWORD_ERROR)	401
27001	{} (PASSWORD_VERIFICATION_ERROR)	401
27009	Please enter a valid username and password.	401
27013	External (SSO or LDAP) authentication is required.	401
27014	Please enter valid answers to security questions.	401

ModelError	Message	HTTP Code
5000	[{}] Child model exists; ({})	400
5001	[{}] Model already exists; ({})	400
5002	One or more data sync errors occurred; ({})	400
5003	[{}] The helper cannot instantiate a model it does not recognize; ({})	400
5004	[{}] The specified resource could not be found; ({})	404
5005	[{}] A single model instance was expected but more than one was found; ({})	404
5006	[{}] Attempt to modify a read-only model failed; ({})	400
5007	[{}] Attempt to modify a read-only model field failed; ({})	400
5008	[{}] Data does not conform to schema; {}	400
5009	[{}] Validation failed; {}	400
5010	[{}] Error manipulating schema; ({})	400
5011	[{}] Error generating schema; ({})	400
5012	[{}] Invalid foreign key to {} for business keys {}	400
5013	[{}] Badly-formed schema; ({})	400
5014	[{}] Error deriving field value; {}	400
5015	Singleton constraint violated: Only one instance of [{}] is allowed per {}.	400
5016	The existing device in [{}] model cannot be modified, it is referenced by other resources.	400
5017	[{}] Invalid foreign key to {} for value {}	400
5018	[{}] Operation not supported for model instance; ({})	405
5019	[{}] Operation not supported; ({})	405
5020	Unable to determine workflow for operation "{}"	400
5021	Workflow "{}" not found	400
5022	Workflow operation "{}" clashes with an existing model attribute/method	400
5023	Unable to execute {} workflow. {}	400
5024	Unable to compile data for provisioning workflow for {}, error {}	400
5025	[{}] Connection timeout error after ({}) seconds	400
5026	[{}] Connection error; ({})	400

Continued on next page

Table 3 -- continued from previous page

ModelError	Message	HTTP Code
5027	[{}] API retry error; ({})	503
5028	[{}] Authentication error; ({})	400
5029	[{}] Attempt to add a contradicting rule; ({})	400
5030	[{}] Phones of this type must be added as gateway endpoints	400
5031	[{}] Unable to add NDLR to hierarchy node containing device models belonging to devices not referenced by NDLR	400
5032	[{}] Unable to query API with available data [{}]	400
5033	Retries exhausted; ({})	400
5050	Password cannot be reused.	400
5051	New password must have {} characters different from old password.	400
5052	User cannot change their password more than once within {} day(s). Please contact your administrator.	400
5053	Password does not meet minimum length required.	400
5054	Password {}.	400
5200	Invalid connection parameters for {}. Username and Password must specified for BASIC authentication method.	400
5201	Invalid connection parameters for {}. Token must specified for OAUTH authentication method.	400
5202	[{} {}] Unable to render model template [{}]. TEMPLATE: {} CONTEXT: {}	400
5203	[{} {}] Unable to parse API response. RESPONSE: {}	400
5204	Invalid connection parameters for {}. Hierarchy must be specified.	400
5205	[{}] Invalid paging parameters: page_size {} page_offset {}	400
5206	[{}] Paging required: page_size {} page_offset {}	400
5207	[{}] External response exceeded memory limit [{}] [{} {}]	400
5208	[{}] Template output exceeded memory limit [{}] [{}]	400
5209	[{}] Bad override for [{}]	400
5210	[{}] Session expired. The session cache has been cleared and the next request will go through successfully.	400
5211	[{}] Unable to authenticate using session based auth. {}	400
5212	[{}] Cannot add device {}	400
5215	[{}] Disallowed input [{}]	400
5998	[{0}] {1}	400
5999	Error, {}. (UNHANDLED_ERROR)	400

ApiError	Message	HTTP Code
3000	Hierarchy context may not be None, please select Hierarchy	400
3001	Error, Incorrect request format	400

Continued on next page

Table 4 -- continued from previous page

ApiError	Message	HTTP Code
3002	Error, Unhandled method for URL	400
3003	Invalid import file specified. {}	400
3004	Invalid export URL specified. {}	400
3005	Error, Invalid list view sort key [{}]. Valid options are {}	400
3006	Error, Invalid list direction [{}]. Valid options are {}	400
3007	Error, No schema available during list view	400
3008	Provisioning Workflow error [{}]	400
3009	Nothing to export	400
3010	List delete failed, error [{}]	400
3011	List size not allowed, requested [{}], maximum [{}]	400
3012	List sort by hierarchy path not allowed	400
3013	Function not implemented	400
3014	Attribute field name required	400
3015	Hierarchy path [{}] not found.	400
3016	Model type list [{}] not found at or above the current hierarchy.	400
3017	Bulk update failed, error [{}].	400
3018	Bulk operation {} failed, error [{}].	400
3019	Schemas of data being imported have cyclic foreign keys {}.	400
3020	Imported {} out of {} items successfully.	400
3021	{} is a required GET parameter.	400
3022	Invalid Range HTTP header: {}	400
3023	{} is an invalid GET parameter.	400
3024	Resource pkid(s) must be specified	400
3025	Request was throttled.	429
3026	Invalid UTC date format given: {0}, requires: {1} or {2}	400
3027	The current filter caused a long running request. Please add more filter fields, use Case Sensitive or change the criteria types to one of {}.	400
3028	Model Instance Filter [{}] not found at or above the current hierarchy.	400
3029	Purge failed, error [{}]	400
3030	Model Type List of [{}] type not valid for [{}] sync.	400
3031	Model Instance Filter of [{}] type not valid for [{}] sync.	400
3032	{} GET parameter has an invalid value.	400
3999	Unhandled API Error	400

AuthError_11_5_3	Message	HTTP Code
27000	{} (INCORRECT_PASSWORD_ERROR)	403
27001	{} (PASSWORD_VERIFICATION_ERROR)	403
27002	{} (USER_NOT_FOUND_ERROR)	404
27003	{} (LOGIN_NOT_ALLOWED_ERROR)	403
27004	Account locked. Please contact your administrator.	403
27005	Too many failed login attempts for this user account. Try again later.	403
27006	Too many failed login attempts from this computer. Try again later.	403
27007	Your Web browser doesn't appear to have cookies enabled. Cookies are required for logging in.	400
27008	User is not allowed to log in.	403
27009	Please enter a valid username and password.	403
27010	This account is inactive.	403
27011	User account password must be changed before any API requests are authorized.	403
27012	{} (ACCOUNT_DISABLED)	403
27013	External (SSO or LDAP) authentication is required.	403
27014	Please enter valid answers to security questions.	403
27015	Password reset is not available for user.	403
27016	Security questions and answers not set up.	403
27017	User can not log in to this interface.	403
27018	User is disabled due to inactivity	403
27019	User is not allowed to login. Please contact your administrator.	403
27020	Login is currently disabled due to a temporary overload. Please try again later.	503
27021	User is not allowed to log in. Maximum user login sessions has been reached.	403

DatabaseError	Message	HTTP Code
2000	Cannot setup Mongo DB collection {}	400
2001	Find failed with spec={}, fields={}, skip={}, limit={}, sort_by={}, err={}	400
2002	Find one failed with spec={}, fields={}, err={}	400
2003	Get archive history failed with spec={}, fields={}, skip={}, limit={}, err={}	400
2004	Remove failed with spec={}, err={}	400
2005	Find and modify failed with spec={}, modify={}, err={}	400
2006	Save failed with spec={}, modify={}, err={}	400
2007	Count failed for {}	400
2008	Find failed with spec={}, fields={}, err={}	400
2009	Duplicate error with spec={}, modify={}, err={}	400
2010	Found more than one record with spec={}	400
2100	Error, Cannot connect to RESOURCE database collection	400
2101	Error, Cannot connect to DATA database collection	400
2102	Error, Cannot connect to ARCHIVE database collection	400
2103	Aggregate failed with group_by={}, match={}, aggregations={}, sort={}, err={}	400
2104	Bulk insert failed, err={}	400
2106	Bulk write failed, err={}	400
2999	Unhandled Database Error	400

AuthenticationProxyError	Message	HTTP Code
32000	Cannot decode target user from authentication proxy. Error: {}	400
32001	Insufficient target user details specified by authentication proxy. Target user details must be contained in a JSON-formatted object with an email attribute.	400
32002	User [{}] is not a valid authentication proxy.	400
32003	Proxy user must be at a hierarchy above that of the target user.	400
32004	Error, {} (UNHANDLED_ERROR)	500

LibSchemaError	Message	HTTP Code
9000	Unhandled schema property error: [{}]	400
9001	Unhandled schema and data processing error: [{}]	400
9002	Data type incorrect, property: {}, not of type: {}	400
9999	Error, (UNHANDLED_ERROR)	400

RbacError	Message	HTTP Code
16000	Permission denied: {}.	400
16001	User not found.	400
16002	Role not specified; User [{}]	400
16003	Access profile not specified; User [{}], Role [{}]	400
16004	Role not found; User [{}], Role [{}]	400
16005	Access profile not found; User [{}], Role [{}], Access Profile [{}]	400
16006	User [{username}] is not allowed to {operation} attribute(s) of {model_type} resource [{pkid}]. Attribute(s) in breach: {breach_attrs}. This operation must be performed by the user's administrator.	403
16007	User [{username}] is not allowed to {operation} {model_type} resource [{pkid}]. This operation must be performed by the user's administrator.	403
16008	Invalid authorization token detected.	403
16009	Role not found; Hierarchy [{}], Role [{}]	400
16010	Access profile [{}] not found for Role [{}] in or above Hierarchy [{}]	400
16011	Access profile of role [{}] is not a subset of the request user's.	400
16012	SelfService Access Profile [{}] for Role [{}] at Hierarchy [{}] must not be created outside 'sys' hierarchy.	400
16999	Error, (UNHANDLED_ERROR)	400

SsoSettingsError	Message	HTTP Code
30000	Invalid certificate file found.	400
30001	Invalid key file found.	400
30002	Validity must not be negative or larger than {} hours ({} years).	400

ApiVersionError	Message	HTTP Code
38000	Invalid API header version specified: {}.	400
38001	No API version mapping defined.	400
38002	API header version: {} and API parameter version: {} mismatch	400

ExportError	Message	HTTP Code
36000	The export format is not specified in request.	400
36001	The specified export format is not supported.	415
36002	The worksheet was not initialized and can not be exported.	500

DataImportError	Message	HTTP Code
11000	Multiple json files {} found in zip archive root; only 1 expected	400
11001	Import file validation failed with: {}	400
11999	Error, (UNHANDLED_ERROR)	400

InterfaceError	Message	HTTP Code
50000	Invalid interface value [{}] for header 'X_INTERFACE'	403
50001	No access profile associated with Interface [{}]	403

BulkLoadMacroError	Message	HTTP Code
60000	Data type must be {}	400
60001	Invalid bulk load macro format {}. Supported format: {}	400

MigrationError	Message	HTTP Code
21000	Post condition failed. {}	400
21999	Error, {} (UNHANDLED_ERROR)	400

CryptoError	Message	HTTP Code
19000	Cryptography validation failed; {}.	400
19999	Error, (UNHANDLED_ERROR)	400

Saml2SsoError	Message	HTTP Code
14000	Could not find SSO settings; Hierarchy: {}.	400
14001	Found multiple SSO settings, only one expected; Hierarchy: {}.	400
14002	Could not find SSO Identity Provider; Hierarchy: {}, IDP uri: {}.	400
14003	Could not resolve SSO Identity Provider; Hierarchy: {}, IDP uri: {}.	400
14004	System generated certificate expected but not specified in data/SsoSettings.	400
14005	System generated certificate has an invalid private key.	400
14006	System generated certificate has an invalid certificate.	400
14007	Unknown principal: {}.	400
14008	Unsupported binding: {}.	400
14009	Verification error: {}.	400
14010	SubjectConfirmation is used but there is no NotOnOrAfter attribute	400
14012	NotBefore and NotOnOrAfter should be present when using either in Condition	400
14013	OneTimeUse element should be present when neither NotBefore nor NotOnOrAfter attributes in Condition	400
14014	Only one OneTimeUse element should be present in Condition	400
14015	Unencrypted assertions are not allowed	400
14016	The session cannot be used yet	400
14999	Error: {}. (UNHANDLED_ERROR)	400

ScriptError	Message	HTTP Code
8000	Script not found	400
8002	Syntax error on line {}	400
8003	Could not connect to {}	400
8004	Authentication failed {}	400
8999	Error, (UNHANDLED_ERROR)	400

HierarchyBasedAccessError	Message	HTTP Code
22000	Invalid traversal argument: '{}'; Traversal must be one of {}.	400
22001	{model_type} with {attr_name} "{attr_value}" is only permitted at the following hierarchy type(s): {hierarchy_types}.	403
22999	Error, {} (UNHANDLED_ERROR)	400

TestConnectionError	Message	HTTP Code
12000	Please specify the model type of the device connection parameters	400
12999	Error, (UNHANDLED_ERROR)	400

SysError	Message	HTTP Code
0	Error, Mongo service not started	400
1	Error, Server too busy	400
2	Error, Celery service not started	400

PlatformError	Message	HTTP Code
28000	Could not execute command: {}; Exit code: {}	500
28999	Error, {} (UNHANDLED_ERROR)	500

InternalApiUserError	Message	HTTP Code
18000	Authorization user [{}] not found.	400
18999	Error, (UNHANDLED_ERROR)	400

SystemMonitoringError	Message	HTTP Code
70000	Aggregate {} is not supported by {}	400

RisApiError	Message	HTTP Code
80000	RIS API data collection failed for {}	400

ThemeError	Message	HTTP Code
90000	Theme name {} is reserved for system use. Please choose another name. RIS API data collection failed for {}	400

4   Tool APIs

4.1.   Introduction to Tool APIs

This section describes API calls that are not related to a specific model. The full URL would include the host name: *http://[hostname]*.

The calls described here all contain `/tool/` in the URL and the full list can be obtained by the GET call for `choices`:

Task	Call	URL	Parameters	Response
List tools	GET	/api/tool/choices/	format=json hierarchy=[hierarchy]	JSON format of the list of tools as `title`-`value` pairs.

Variables are enclosed in square brackets, e.g.:

- [hierarchy] is the hierarchy UUID
- [filename] refers to a file

Other parameters are described with the relevant API call.

4.2.   Search and Search Result Export

For an API call that carries out a search, a POST payload in JSON format is added.

Task	Call	URL	Parameters	Payload
Search	POST	/api/tool/Search/	format=json hierarchy=[hierarchy]	{"query":"[query]"}

The value of `[query]` follows Search syntax, for example:

```
{"query":"data/Countries with country_name contains King"}
```

While the default search direction is down, a second parameter can be added to `[query]` to indicate the hierarchy direction to search. These are enabled by adding a value `true`:

- `hierarchy_paths` - up the hierarchy
- `hierarchy_shallow` - at the local hierarchy
- `hierarchy_all` - up and down the hierarchy

For example, if the user making the API call is at `sys.hcs`, then a call payload like:

```
{"query":"relation/Bundle with name is 'HcsBase' ", "hierarchy_paths":true}
```

will also search up the hierarchy path.

The parameter `filter_hierarchy` can also be used to filter the hierarchy of search results. Specifying a hierarchy to which the user has no access will return a Permission Denied error.

Note:

- If for both `hierarchy=sys.hcs` and `filter_hierarchy=sys.hcs.CSP` for example are used, then `filter_hierarchy=sys.hcs.CSP` takes precedence.

- If for neither `hierarchy=sys.hcs` nor `filter_hierarchy=sys.hcs.CSP` for example are used, then the user's hierarchy is applied.

The Request payload can also be a GET parameter, for example:

Task	Call	URL	Parameters	Response
Search	GET	/api/tool/Search/	format=json hierarchy=[hierarchy] filter_hierarchy=[hierarchy] query=[url_query]	JSON format of the search result.

The value of `[url_query]` is URL encoded string, for example:

```
data/Countries%20with%20country_name%20contains%20King
```

Furthermore, the `meta` property of the schema in the response to `/api/tool/Search/` contains action details for the export of search results. This includes the URL for the data export POST request:

```
/api/export/export_data/?url=/api/tool/Search/
```

as well as the URL:

```
/api/view/ExportData/add
```

which has a schema that lists the data export data type choices that will be used as a parameter to the POST call.

4.3. Bulk Load API

Two API calls are required.

Task	Call	URL	Parameters	Response

Submit file	POST	`/api/`
		`    uploadfiles/`
		This URL will be moved to tool/UploadFile in future.
			`hierarchy=[hierarchy]`
			`Content-Type:`
			`    multipart/form-data`
			
			`name='uploadedfile'`
			`filename=<filename>`
			`the file to upload`
				`{"uploadedfiles":`
				`[{"id": "<file_id>",`
				`"name": "<filename>"}]}`

The response is HTTP 202

Task	Call	URL	Parameters	Payload

Bulk Load	POST	`/api/tool/`
		`    BulkLoad/`
			`method=`
			`    bulkload_spreadsheet`
			`hierarchy=[hierarchy]`
				Examples:
				`{'bulkload_file':`
				`    '<filename>',`
				`'execute_immediately':`
				`    true}`
				or:
				`{'bulkload_file':`
				`    '<filename>',`
				`'execute_immediately':`
				`    false`
				`'execute_date':`
				`    '2013-06-20',`
				`'execute_time':`
				`    '12:00:00',`
				`'execute_timezone':`
				`    '0'}`

The following `curl` commands illustrate the two steps:

Step 1

```
curl -H 'Authorization: Basic <auth_key>'
    -F uploadedfile="@<file>.xlsx"
    'http://<hostname>/api/uploadfiles/'
```

Step 2

```
curl -H 'Authorization: Basic <auth_key>'
    -H 'Content-Type: application/json'
    -H 'accept: application/json'
    --data-binary '{"bulkload_file":"DEMO.xlsx","execute_immediately":true}'
    'http://<host>/api/tool/BulkLoad/?hierarchy=[hierarchy]&
      method=bulkload_spreadsheet&
      nowait=true&
      format=json'
```

The response to this call is for example as in the following table.

Response

```
{"href": "/api/tool/Transaction/0b340a6f-b658-48bb-ac8c-7562adc5572d",
 "success": true,
 "transaction_id": "0b340a6f-b658-48bb-ac8c-7562adc5572d"}
```

- If the Bulk Load is to be scheduled, the payload of the second task includes schedule details:

  – *execute_immediately* is set to false

  – *execute_date* is added in the format YYYY-MM-DD

  – *execute_time* is added in the format HH:MM:SS

  – *execute_timezone* is added in the format of a numeric value in minutes relative to UTC. For example, UTC is 0, UTC+2:00 is 120, UTC-1:00 is -60, and so on.

- An entry is also generated in the schedule; that is, an instance is added to the data/Schedule module.

- If the second task payload has *'execute_immediately':true*, a POST is generated to /api/data/Bulkload/. The payload includes the uploaded filename and a generated name and time stamp as well as a description, for example:

```
{'filename': '<file>.xlsx', 'description': 'Generated by Bulk Loader
Administration Tools', 'name': 'AnyUser.xlsx -- 2013-05-21
16:47:11.801664 (UTC)'}
```

To inspect the detailed progress and status of the transaction, use the API call from the response above:

```
GET /api/tool/Transaction/[pkid]
```

with parameters:

- `hierarchy=[hierarchy]`

- `format=json`

The response to this GET call is a JSON object that provides details of the transaction, as for example in the truncated snippet:

```
...
   "href": "/api/tool/Transaction/[pkid]
    "log_id": "53a8053ea616540708141f44",
    "message": "data_Countries_bulkloadsheet.xlsx is a valid
    "severity": "info",
    "time": "2014-06-23T10:45:18.029000",
    "transaction_id": "[pkid]"
  }
],
"pkid": "[pkid]",
"resource": {},
"rolled_back": "No",
"started_time": "2014-06-23T10:45:17.813000",
"status": "Success",
"sub_transactions": [
  {
    "action": "Execute Resource",
    "detail": "Execute : data_Countries_bulkloadsheet.xlsx -- ...
    "status": "Success",
```

(continues on next page)

```
      "submitted_time": "2014-06-23T10:45:19.567000",
      "transaction": "/api/tool/Transaction/[pkid1]     ...
    },
    {
      "action": "Create Schedule",
      "detail": "Name:data_Countries_bulkloadsheet.xlsx -- 2014- ...
      "status": "Success",
      "submitted_time": "2014-06-23T10:45:18.912000",
      "transaction": "/api/tool/Transaction/[pkid2]     ...
    },
    {
      "action": "Create Bulk Load",
      "detail": "Name:data_Countries_bulkloadsheet.xlsx -- 2014-06 ...
      "status": "Success",
      "submitted_time": "2014-06-23T10:45:18.419000",
      "transaction": "/api/tool/Transaction/[pkid3]     ...
    }
  ],
  "submitted_time": "2014-06-23T10:45:17.794000",
```

On the GUI, the same transaction displays as in the Transaction log image.

For long transactions, to retrieve a summary of the status of the transaction, the transaction can be polled, using `poll` in the URL, using the same parameters:

```
GET /api/tool/Transaction/poll/?transactions=[pkid]
```

In this case, there is a shortened response, for example:

```
{"[pkid]":
 {"status": "Processing",
  "href": "/api/tool/Transaction/0b340a6f-b658-48bb-ac8c-7562adc5572d",
  "description": null}
}
```

4.4. Move and Bulk Move

The following model types can be enabled for this operation:

- Data models
- Device models
- Relations

For an API call that carries out a move on a `<model_type>`, a POST payload in JSON format is added.

A move can only take place from a source hierarchy equal to or lower than `[target_hierarchy]`.

- Task

  Move the instance with [pkid] to [target_hierarchy]

Call	URL	Parameters	Payload
POST	/api/tool/DataMove/ ?model_type= <model_type>	format=json hierarchy=[hierarchy] context_hierarchy= [target_hierarchy] (nowait=true)	{"hrefs": ["/api/<model_type>/[pkid]"]}

- Task

  Move one or more model instances ([pkid1], [pkid2],...) from source hierarchy (pkid or dot notation) to target_hierarchy (pkid or dot notation).

Call	URL	Parameters	Payload
POST	/api/tool/DataMove/ ?model_type= <model_type>	format=json hierarchy=[hierarchy] context_hierarchy= [target_hierarchy] (nowait=true)	{"hrefs": ["/api/relation/Subscriber/[pkid1]", "/api/relation/Subscriber/[pkid2]", ...]}

For a list of hierarchy pkids and their dot notation available from [hierarchy], use the GET call:

```
GET api/relation/<model_type>/
 ?hierarchy=[hierarchy]
 &format=json
 &schema_rules=true
```

4.5. Custom Workflows

Custom Workflows can be added to:

- Domain Models

- Relations

A Custom Workflow can be called from a model *instance*.

The usage in the URLs and parameters below are:

[model]

- Domain Model [domain/DomainModelName]

- Relation [relation/RelationName]

[pkid]

Model instance pkid

[CustomWF]

Custom Workflow name. The name is of the format *add-*, *del-* to indicate the operation type.

- For Domain Models, the Custom Workflow name suffix corresponds with a Group name of Domain Model attributes.

- For Relations, the Custom Workflow name suffix corresponds with the alias of the joined model type.

To get the payload schema for a Custom Workflow, carry out a *list* API call for the instance, with parameters:

Task	Call	URL	Parameters
Get payload schema	GET	/api/[model]/[pkid]/	hierarchy=[hierarchy] format=json schema=true schema_rules=true

The response contains:

- The **action** to carry out the Custom Workflow. For example, for an **add** action on a domain model DOMAIN100 instance with a group name or alias called ADDRESS:

```
AddADDRESS: {
href: "/api/domain/DOMAIN100/523c2213a61654174273ab07/+AddADDRESS/"
title: "Addaddress"
schema: "ADDRESS"
method: "POST"
submit: "payload"
class: "add"
}
```

- The **schema** of the model in the response contains the specification of the **submit** payload for the Custom Workflow.

Task	Call	URL	Parameters	Payload
Call Custom add workflow	POST	/api/domain/[model]/[pkid]/+[CustomWF]/	hierarchy=[hierarchy]	See below

Payload for grouped attributes is defined in the schema that is returned from the GET call above. For PUT methods the resource data is replaced with the data specified in the request. All fields of the resource is replaced with the fields in the request.

This means that:

- fields not present in the request that are present in the resource will be dropped from the resource

- fields present in the request that are not present in the resource will be appended to the resource

- the data of fields present in the request is used to update fields that already exist in the resource

PATCH methods operate in two modes depending on the content type:

- `Content type: application/json`

- The values of data fields present in the request is used to update the corresponding resource fields. This means that:

  - Fields present in the request but not in the resource is appended to the resource. The value of each field that is already present in the resource is updated from the request data.

Note: Field values that are set to null in the request is dropped from the resource. Fields that are present in the resource but not in the request are left untouched.

`Content type: application/json-patch+json`

Existing resource data is patched according to RFC6902.

Modifying data fields:

- To drop the field from a data model, specify null as the parameter value (i.e. `{"field": null}`)

- To blank out a string value set the parameter value to an empty string (i.e. `{"field":""}`)

- When the key (field name) appears in the field for a parameter, then the field is updated with the supplied value.

- Any field that is not specified in the request will be left untouched

- When a key (field name) is specified but no value is supplied, or an empty string is supplied, the value is blanked out or set to NULL

5 Transactions

5.1. List Transactions

To list transactions on the system use the following operation

```
GET https://<server_address>/api/tool/Transaction/
  ?hierarchy=[hierarchy]
  &format=json
  &summary=true
```

The following query parameter illustrates how a second page of 50 transactions in the transaction log is requested.

```
skip=50
&limit=50
&hierarchy=[hierarchy]
&format=json
&summary=true
&direction=desc
&order_by=submitted_time
```

For further information on the query parameters refer to API Parameters above.

The synchronous response contains:

- pagination information
- meta data specifying the summary attributes of the transaction log view
- resources containing a list of the transactions in the transaction log

5.2. Get Instance Transactions

The status of a specific transaction can be retrieved by using a GET call to `/tool/Transaction` for a specific transaction pkid (also referred to as transaction ID or transaction_id). The transaction_id is available in for example the synchronous response to an asynchronous mutator transaction.

For example, if the transaction_id in the response is `[pkid]`, then the transaction can be polled with:

```
GET https://<server_address>/api/tool/Transaction/[pkid]
```

The GET response `data` section of the JSON content for a transaction also shows:

- `submitter_host_name`: the host name of the application node that scheduled the transaction.

- `processor_host_name`: the host name of the application node that processed the transaction (this value will only be set once the transaction is processed).

On a clustered system, these attributes make it possible to distinguish between the application nodes on which the transaction was respectively scheduled and processed.

Refer to the examples in the API Response topics, in particular, the topics POST/PUT/DELETE/PATCH Response and Asynchronous Mutator Transaction Status Callback.

5.3. Poll Transactions

It is recommended to use asynchronous transaction call back mechanism described in "Asynchronous Mutator Transaction Status Callback". If this can however not be used a consumer of the VOSS-4-UC API can also use this polling mechanism to poll the status of individual transactions using the poll action of the transaction tool. A user interface that allows a user to monitor the progress of a given transaction can use the following method to retrieve the status of a given transaction:

```
GET /api/tool/Transaction/[pkid]/poll/?format=json
```

The response contains essential status of the transaction, for example:

```
{
    [pkid]: {
        status: "Success",
        href: "/api/tool/Transaction/[pkid]",
        description: "Name:RDP-auser1857 Description:RD for auser1857"
    }
}
```

5.4. Replay Transactions

Transactions that have failed can, under certain circumstances, be replayed. This means that the transaction is re-submitted with the original request parameters. This is done by specifying the pkid of the transaction

```
GET https://<server_address>/api/tool/Transaction/[pkid]/replay/
```

The transaction current operation replays the transaction and the result returns the list view of the transaction log.

5.5. Edit and Replay Transactions

Transactions can, under certain circumstances, be edited and then replayed. This means that the transaction is re-submitted with the updated request parameters. This is done by specifying the pkid of the transaction:

```
GET https://<server_address>/api/tool/Transaction/[pkid]/edit-replay/
```

The transaction current operation edit and then replays the transaction and the result returns the list view of the transaction log.

5.6. Sub Transactions

The sub-transactions of a transaction with pkid can be retrieved by submitting the following URI

```
GET https://<server_address>/api/tool/Transaction/[pkid]/sub_transaction/
```

5.7. Log Transactions

The log messages of a transaction with pkid can be retrieved by submitting the following URI

```
GET https://<server_address>/api/tool/Transaction/[pkid]/log/
```

5.8. Transaction Choices

A URL endpoint and parameter is available to list the transaction actions as they may be shown in the transaction log.

- The API call to get the list of transaction actions uses the parameter and value: `field=action`, for example:

```
GET api/tool/Transaction/choices/?
    field=action&
    hierarchy=[hierarchy]&
    format=json
```

The output shows the list of transaction actions:

```
[
  {
    "value": "Auto Migrate Base Customer Dat",
    "title": "Auto Migrate Base Customer Dat"
  },
  {
    "value": "Auto Migrate Base Provider",
    "title": "Auto Migrate Base Provider"
  },
  {
    "value": "Auto Migrate Base Site Dat",
    "title": "Auto Migrate Base Site Dat"
  },
  {
    "value": "Auto Migrate Dial Plan",
    "title": "Auto Migrate Dial Plan"
  },
  {
    "value": "Auto Migrate Feature Subscriber Phone Cft",
    "title": "Auto Migrate Feature Subscriber Phone Cft"
  },
  {
    "value": "Auto Migrate Hotdial Data",
```

```
    "title": "Auto Migrate Hotdial Data"
  },
  {
    "value": "Auto Migrate Init Ippbx",
    "title": "Auto Migrate Init Ippbx"
  },
  {
    "value": "Auto Migrate Internal Number Inventory",
    "title": "Auto Migrate Internal Number Inventory"
},

...
```

5.9. Transaction Filters

In addition to the filter parameters that can be applied to transactions as indicated in the topic on API Parameters, transactions in particular can be filtered:

- by the following values for the URL parameter `filter_field`:

    - **Transaction ID:** `id`

    - **Start or end submitted time:** `submitted_time`

    - **The transaction message:** `message`

- by also listing sub transactions using the URL parameter and value `subtransactions=true`. By default, sub transactions are not listed, in other words, the value is `false`.

- To carry out a filter on sub-transactions of a parent transaction, the `/sub-transactions/` endpoint is added to the GET request:

```
/api/tool/Transaction/[parent-pkid]/sub-transactions/
```

- To carry out a filter on transaction logs of a parent transaction, the `/logs/` endpoint is added to the GET request:

```
/api/tool/Transaction/[parent-pkid]/log/
```

    The transaction filters do not apply to logs.

The parameters can have the `filter_condition` values:

- `eq` (equals)

- `ne` (not equals)

- `gt` (greater than)

- `gte` (greater than or equals)

- `lt` (less than)

- `lte` (less than or equals)

The date-time is a `filter_text` value for `filter_field=submitted_time`.

The format follows RFC3339: [https://tools.ietf.org/html/rfc3339] and is *YYYY-MM-DDTHH:MM:SS.fZ*, where:

- "T" is the time separator and the character should be added.

- "Z" indicates UTC time and the character should be added.

- "f" represents the decimal fraction of a second and the character should not be added. The specification of the decimal fraction is optional.

For example:

June 29 2016 14 hours 41 minutes 0.01 seconds UTC, is:

```
2016-06-29T14:41:00.01Z
```

To filter for transactions after this date-time, the API call is:

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=submitted_time
    &filter_text=2016-06-29T14:41:00.01Z
    &filter_condition=gt
```

To filter between transaction IDs or times, two parameter sets are needed.

For example:

- To filter transaction IDs between 12000 and 13000:

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=id
    &filter_text=12000
    &filter_condition=gt
    &filter_field=id
    &filter_text=13000
    &filter_condition=lt
```

- To filter transactions between June 29 2016 14 hours 41 minutes UTC and June 29 2016 15 hours 41 minutes UTC (no fraction of a second in the example):

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=submitted_time
    &filter_text=2016-06-29T14:41:00Z
    &filter_condition=gt
    &filter_field=submitted_time
    &filter_text=2016-06-29T15:41:00Z
    &filter_condition=lt
```

If the upper or lower bound in the filter are not available, the transactions with values between the filter values and the bound are returned.

When the URL parameter `subtransactions=true` is used, the `data` object in the JSON API response shows:

- a parent transaction has: `parent:   null`

- a sub transaction has: `parent:   <pkid>`, where `<pkid>` is the value of the parent attribute `pkid`.

The example snippets below show the values of `parent`:

```
data: {
  username: "system",
  status: "Success",
  description: "",
  parent: null,
  pkid: "01a559c5-e77f-40e7-8403-683d7204d1e1",
  friendly_status: "Success",
  detail: "HcsLdapSyncSchedule--1",
  action: "Execute Schedule",
  href: "/api/tool/Transaction/01a559c5-e77f-40e7-8403-683d7204d1e1/",
  txn_seq_id: "17693",
  data_type_: "tool/Transaction",
  message: "",
  submitted_time: "2016-07-14T12:13:41.758000Z"
}


data: {
  username: "system",
  status: "Success",
  description: "",
  parent: "f4daa234-590d-4002-a3b0-8c329c583d1d",
  pkid: "019f44a3-df6e-4e4f-86f3-a09a6b91e482",
  friendly_status: "Success",
  detail: "10.120.2.221",
  action: "Import Ldap",
  href: "/api/tool/Transaction/019f44a3-df6e-4e4f-86f3-a09a6b91e482/",
  txn_seq_id: "17695",
  data_type_: "tool/Transaction",
  message: "models completed.",
  submitted_time: "2016-07-14T12:13:43.075000Z"
}
```

- In the case of a transaction error, the `message` attribute value will show the corresponding error message. If a custom message was defined in a provisioning workflow, and the response is the result of the workflow, the value will be the custom message.

  To filter transaction messages, the parameter `filter_field=message` is used, with at least one of the following additional filter criteria:

  - a date range of maximum 7 days using `submitted_time`

  - an additional `filter_field`, with one of the conditions:

    * `filter_condition=contains`

    * `filter_condition=startswith`

    Also required is the case sensitive parameter:

    * `ignore_case`

    Note that a filter is by default case insensitive. If the case is explicitly set, then it should be added *to each filter parameter group* in order to ensure proper parameter grouping.

  The additional criteria do not apply to sub-transaction message filters, because the `[parent-pkid]` in the URL serves as an additional filter.

  The example below is a message filter that contains "Invalid business key", by date range:

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=submitted_time
    &filter_text=2016-06-25T14:41:00Z
    &filter_condition=gt
    &ignore_case=false
    &filter_field=submitted_time
    &filter_text=2016-06-29T15:41:00Z
    &filter_condition=lt
    &ignore_case=false
    &filter_field=message
    &filter_text=Invalid%20business%20key
    &filter_condition=contains
    &ignore_case=false
```

6   API Examples

6.1.   API Examples Overview and Conventions

The example sections illustrate the use of the API for a number of calls, using the `curl` command line tool. Each example shows the command and the console output.

- API calls are illustrated:
    - From server `http://localhost`
    - Referencing a relation: `relation/LineRelation`.
- User authorization is for two administrator users:
    - one user has additional permissions to import and bulk load.
    - The Authorization header and hierarchy parameter in the URL identify the two users.
    - Field Display Policies and Configuration Templates in URL parameters can differ according to the MenuLayout associated with the user role, for example, `&policy_name=LineMenuFDPSite` is a Field Display Policy applied to `relation/LineRelation` from a Site administrator user menu.
- Where a response to an API call shows an instance of `/api/tool/Transaction/`, the transaction instance details can be inspected with a GET call to this instance. An example is shown in this section.
- Some payload files and console output is truncated (indicated with ellipses or text: "snippets").
- Line breaks have been added to console output in the examples for better formatting.
- The a selection of the MS Excel bulk load sheet `LineRelation.xlsx` columns are shown in a table.

6.2.   POST

- task: POST and instance of `relation/LineRelation`
- user: site administrator
- hierarchy: `55b9dc81a6165413b9d16ab6`
- Field Display Policy: `LineMenuFDPSite`
- Configuration Template: `line-cft`

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA=='
      -H 'Content-Type:application/json'
      --data-binary @post-payload.json
      -X POST 'http://localhost/api/relation/LineRelation/
              ?hierarchy=55b9dc81a6165413b9d16ab6
              &policy_name=LineMenuFDPSite
              &template_name=line-cft
              &nowait=true
              &format=json'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1... connected
> POST /api/relation/LineRelation/
        ?hierarchy=55b9dc81a6165413b9d16ab6
        &policy_name=LineMenuFDPSite
        &template_name=line-cft
        &nowait=true
        &format=json HTTP/1.1
> Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA==
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
              libcurl/7.22.0
              OpenSSL/1.0.1
              zlib/1.2.3.4
              libidn/1.23
              librtmp/2.3
> Host: localhost
> Accept: */*
> Content-Type:application/json
> Content-Length: 1941
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 13:10:46 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, DELETE, HEAD, OPTIONS
< X-CSRFToken: d2q7nV4aWDWFpuazsnRvJVMcj9qX5Ksg
< Set-Cookie: csrftoken=d2q7nV4aWDWFpuazsnRvJVMcj9qX5Ksg;
              httponly; Path=/
< Set-Cookie: sessionid=hahbo0wy7sa8u8rfaiz2tcqxvkvwshp8;
              httponly;
              Path=/
<
+ Connection #0 to host localhost left intact
+ Closing connection #0
{"href": "/api/tool/Transaction/aff36c0b-ff6a-431b-be58-d2f636edb7cd/",
 "success": true,
 "transaction_id": "aff36c0b-ff6a-431b-be58-d2f636edb7cd"}
```

Snippet of the file: `post-payload.json`:

```
{
  "data": {
    "partyEntranceTone": "Default",
    "cfaCssPolicy": "Use System Default",
    "autoAnswer": "Auto Answer Off",
    "callForwardNotRegisteredInt": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "routePartitionName": "Site-locus1",
    "callForwardOnFailure": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "shareLineAppearanceCssName": "Intl24HrsEnh-locus1",
    "callForwardBusy": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "pattern": "90217",
    "patternPrecedence": "Default",
    "callForwardNoAnswer": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "callForwardNoCoverage": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "callForwardNotRegistered": {
      "callingSearchSpaceName": "Intl24HrsEnh-locus1"
    },
    "usage": "Device",
    "alertingName": "techsupport",
    "enterpriseAltNum": {
      "isUrgent": false,
      "addLocalRoutePartition": false,
      "advertiseGloballyIls": true
```

6.3. GET

- task: GET all instances of `relation/LineRelation`

- user: site administrator

- hierarchy: `55b9dc81a6165413b9d16ab6`

- Field Display Policy: `LineMenuFDPSite`

- Configuration Template: `line-cft`

```
curl -v
    -H 'Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA=='
    'http://localhost/api/relation/LineRelation/
    ?hierarchy=55b9dc81a6165413b9d16ab6
    &policy_name=LineMenuFDPSite
    &template_name=line-cft
    &nowait=true
    &format=json'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1... connected
```

```
> GET /api/relation/LineRelation/
      ?hierarchy=55b9dc81a6165413b9d16ab6
      &policy_name=LineMenuFDPSite
      &template_name=line-cft
      &nowait=true
      &format=json HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
              libcurl/7.22.0
              OpenSSL/1.0.1
              zlib/1.2.3.4
              libidn/1.23
              librtmp/2.3
> Host: localhost
> Accept: */*
> Authorization: Basic YWRtaW5AbG9jdXMxMxLmNvbTpwYXNzd29yZA==
>
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Fri, 31 Jul 2015 08:51:11 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, DELETE, HEAD, OPTIONS
< X-CSRFToken: GdcfnSz25RsL16Qe9lN6ESHBAw8ElDvM
< Set-Cookie: csrftoken=GdcfnSz25RsL16Qe9lN6ESHBAw8ElDvM; httponly; Path=/
< Set-Cookie: sessionid=7kj9nhbh2ra5r1awn40md0059ebm956k; httponly; Path=/
<
```

Snippet of one of the returned instances:

```
...


    },
        "pattern": "90124",
        "patternPrecedence": "Default",
        "callForwardNoAnswer": {
          "destination": null,
          "forwardToVoiceMail": false,
          "callingSearchSpaceName": null
        },
        "hrInterval": null,
        "callForwardNoCoverage": {
          "destination": null,
          "forwardToVoiceMail": false,
          "callingSearchSpaceName": null
        },
        "callForwardNotRegistered": {
          "destination": null,
          "forwardToVoiceMail": false,
          "callingSearchSpaceName": null
        },
        "usage": "Device",
        "summary_device": "10.120.2.216, 8443, prov1.cust1",
```

```
        "hrDuration": null,
        "parkMonForwardNoRetrieveVmEnabled": false,
        "alertingName": "Helpdesk",
        "description": "DN created without device from QAS.",
        "directoryURIs": null,
        "aarVoiceMailEnabled": false,
        "hierarchy_path": "sys.prov1.cust1.locus1",
        "parkMonForwardNoRetrieveIntCssName": null,
        "parkMonForwardNoRetrieveDn": null,
        "allowCtiControlFlag": true,
        "defaultActivatedDeviceName": null,
        "parkMonReversionTimer": null,
        "releaseClause": "No Error",
        "e164AltNum": {
          "numMask": null,
          "addLocalRoutePartition": false,
          "advertiseGloballyIls": false,
          "routePartition": null,
          "isUrgent": false
        },
        "callForwardAll": {

...
```

6.4. PUT

- task: Update instance `relation/LineRelation/55b9fe59a6165413b9d17628`

- user: site administrator

- hierarchy: `55b9dc81a6165413b9d16ab6`

- Field Display Policy: `LineMenuFDPSite`

- Configuration Template: `line-cft`

- Payload file: `put-payload.json`

Snippet of `put-payload.json`, showing the updated `alertingName` value to "Helpdesk":

```
...

{
  "data": {
    "partyEntranceTone": "Default",
    "cfaCssPolicy": "Use System Default",
    "autoAnswer": "Auto Answer Off",
    "callForwardNotRegisteredInt": {
      "forwardToVoiceMail": false
    },
    "routePartitionName": "Site-locus1",
    "callForwardOnFailure": {
      "forwardToVoiceMail": false
    },
    "rejectAnonymousCall": false,
    "aarKeepCallHistory": true,
```

```
    "callForwardBusy": {
      "forwardToVoiceMail": false
    },
    "pattern": "90124",
    "patternPrecedence": "Default",
    "presenceGroupName": "Standard Presence group",
    "callForwardNoAnswer": {
      "forwardToVoiceMail": false
    },
    "callForwardNoCoverage": {
      "forwardToVoiceMail": false
    },
    "callForwardNotRegistered": {
      "forwardToVoiceMail": false
    },
    "usage": "Device",
    "alertingName": "Helpdesk",

...
```

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA=='
      -H 'Content-Type:application/json'
      --data-binary @put-payload.json
      -X PUT 'http://localhost/api/relation/LineRelation/55b9fe59a6165413b9d17628/
              ?hierarchy=55b9dc81a6165413b9d16ab6
              &policy_name=LineMenuFDPSite
              &template_name=line-cft
              &nowait=true
              &format=json'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1... connected
> PUT /api/relation/LineRelation/55b9fe59a6165413b9d17628/
    ?hierarchy=55b9dc81a6165413b9d16ab6
    &policy_name=LineMenuFDPSite
    &template_name=line-cft
    &nowait=true
    &format=json HTTP/1.1
> Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA==
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
              libcurl/7.22.0
              OpenSSL/1.0.1
              zlib/1.2.3.4
              libidn/1.23
              librtmp/2.3
> Host: localhost
> Accept: */*
> Content-Type:application/json
> Content-Length: 1926
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 13:00:33 GMT
< Content-Type: application/json
```

```
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
< X-CSRFToken: GgxBBhTjkB2IUib2lHgIVzeohhmK2arc
< Set-Cookie: csrftoken=GgxBBhTjkB2IUib2lHgIVzeohhmK2arc;
             httponly;
             Path=/
< Set-Cookie: sessionid=8skxwiqojuyz5xl37cdcflbr5ct5ncrk;
             httponly;
             Path=/
<
+ Connection #0 to host localhost left intact
+ Closing connection #0
{"href": "/api/tool/Transaction/0bebcaa2-df37-420f-bd15-3a00ea056092/",
 "success": true,
 "transaction_id": "0bebcaa2-df37-420f-bd15-3a00ea056092"}
```

6.5. DELETE

- **task:** Delete instance `relation/LineRelation/55ba2482a6165413b9d19fb8`

- **user:** site administrator

- **hierarchy:** `55b9dc81a6165413b9d16ab6`

- **Field Display Policy:** `LineMenuFDPSite`

- **Configuration Template:** `line-cft`

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA=='
      -X DELETE 'http://localhost/api/relation/LineRelation/55ba2482a6165413b9d19fb8/
                ?hierarchy=55b9dc81a6165413b9d16ab6
                &policy_name=LineMenuFDPSite
                &template_name=line-cft
                &nowait=true
                &format=json'* About to connect() to localhost port 80 (#0)
+   Trying 127.0.0.1... connected
> DELETE /api/relation/LineRelation/55ba2482a6165413b9d19fb8/
        ?hierarchy=55b9dc81a6165413b9d16ab6
        &policy_name=LineMenuFDPSite
        &template_name=line-cft
        &nowait=true
        &format=json HTTP/1.1
> Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA==
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
             libcurl/7.22.0
             OpenSSL/1.0.1
             zlib/1.2.3.4
             libidn/1.23
             librtmp/2.3
> Host: localhost
> Accept: */*
```

```
>
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 13:21:00 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
< X-CSRFToken: a6fFDYZyk9ET8K8xTq9HITFrRi8TROrV
< Set-Cookie: csrftoken=a6fFDYZyk9ET8K8xTq9HITFrRi8TROrV;
            httponly;
            Path=/
< Set-Cookie: sessionid=9i0w39d1d32mdx6fs2skl564y8pmhmu9;
            httponly;
            Path=/
<
+ Connection #0 to host localhost left intact
+ Closing connection #0
{"href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/",
 "success": true,
 "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"}
```

6.6. Bulk Load Example

- task: Bulk load instances of `relation/LineRelation/`

- user: provider administrator

- hierarchy: `55b9daeca6165413b9d166de`

- Bulk load file: `LineRelation.xlsx`

Snippet of file to Bulk load: `LineRelation.xlsx`

# Hierarchy Node	# Device	# CFT Template	# Directory Number	# Alerting Name
sys.prov1.cust1. locus1	10.120.2.216, 8443, prov1.cust1	line-cft	90218	techsupport
sys.prov1.cust1. locus1	10.120.2.216, 8443, prov1.cust1	line-cft	90219	techsupport
sys.prov1.cust1. locus1	10.120.2.216, 8443, prov1.cust1	line-cft	90220	techsupport
sys.prov1.cust1. locus1	10.120.2.216, 8443, prov1.cust1	line-cft	90221	techsupport
sys.prov1.cust1. locus1	10.120.2.216, 8443, prov1.cust1	line-cft	90222	techsupport

Upload the file:

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk'
      -F uploadedfile='@LineRelation.xlsx'
      'http://localhost/api/uploadfiles/
       ?hierarchy=55b9daeca6165413b9d166de'* About to connect() to localhost port 80␣
→(#0)
+   Trying 127.0.0.1... connected
> POST /api/uploadfiles/?hierarchy=55b9daeca6165413b9d166de HTTP/1.1
> Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
              libcurl/7.22.0
              OpenSSL/1.0.1
              zlib/1.2.3.4
              libidn/1.23
              librtmp/2.3
> Host: localhost
> Accept: */*
> Content-Length: 10455
> Expect: 100-continue
> Content-Type: multipart/form-data;
                boundary=---------------------------5a0f36378f19
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 15:09:25 GMT
< Content-Type: text/html; charset=utf-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept-Encoding
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: POST, OPTIONS
< X-CSRFToken: C4ceiFEWSbjif1O4Jzhr1gZV9ytd9f2F
< Set-Cookie: csrftoken=C4ceiFEWSbjif1O4Jzhr1gZV9ytd9f2F;
              httponly;
              Path=/
< Set-Cookie: sessionid=07z03pbatb1qelahcc01ygufgzsr6i35;
              httponly;
              Path=/
<
+ Connection #0 to host localhost left intact
+ Closing connection #0
{"uploadedfiles": [
  {"name": "LineRelation.xlsx",
   "id": "55ba3e25a616541bb906b209"}
 ]}
```

Bulk load the file:

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk'
      -H 'Content-Type: application/json'
      -H 'accept: application/json'
      --data-binary '{"bulkload_file":"LineRelation.xlsx",
                      "execute_immediately":true}'
      -X POST 'http://localhost/api/tool/BulkLoad/?
```

(continues on next page)

```
                    hierarchy=55b9daeca6165413b9d166de
                    &method=bulkload_spreadsheet
                    &nowait=true
                    &format=json'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1... connected
> POST /api/tool/BulkLoad/
        ?hierarchy=55b9daeca6165413b9d166de
        &method=bulkload_spreadsheet
        &nowait=true
        &format=json HTTP/1.1
> Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
                libcurl/7.22.0
                OpenSSL/1.0.1
                zlib/1.2.3.4
                libidn/1.23
                librtmp/2.3
> Host: localhost
> Content-Type: application/json
> accept: application/json
> Content-Length: 64
>
+ upload completely sent off: 64out of 64 bytes
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 14:51:22 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, HEAD, OPTIONS
< X-CSRFToken: iFh5q8FUBxoXyyiLcELHoO8W5IDFbAiP
< Set-Cookie: csrftoken=iFh5q8FUBxoXyyiLcELHoO8W5IDFbAiP;
                httponly;
                Path=/
< Set-Cookie: sessionid=3ayny2y73i43u6sj9bdyoawhhtr8wbm8;
                httponly;
                Path=/
<
+ Connection #0 to host localhost left intact
+ Closing connection #0
{"href": "/api/tool/Transaction/16e1e599-494a-4898-944a-0528915d2f42/",
 "success": true,
 "transaction_id": "16e1e599-494a-4898-944a-0528915d2f42"}
```

6.7. Export Example

- task: Export an instance `relation/LineRelation/55ba3e55a6165413b9d1a18d` as a formatted `.xlsx` spreadsheet file called: `55ba3e55a6165413b9d1a18d.xlsx`

- user: provider administrator

- hierarchy: `55b9daeca6165413b9d166de`

```
$ curl -v
      -H 'Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk'
      -o 55ba3e55a6165413b9d1a18d.xlsx
      'http://localhost/api/relation/LineRelation/55ba3e55a6165413b9d1a18d/export/
      ?hierarchy=55b9daeca6165413b9d166de
      &export_format=xlsx
      &template_name=line-cft
      &policy_name=LineMenuFDPProv
      &schema=true
      &schema_rules=true'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1...  % Total    % Received % Xferd  Average Speed   Time    Time ⌞
↪  Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
0     0    0    0    0     0      0      0 --:--:-- --:--:-- --:--:--     0connected
> GET /api/relation/LineRelation/55ba3e55a6165413b9d1a18d/export/
      ?hierarchy=55b9daeca6165413b9d166de
      &export_format=xlsx
      &template_name=line-cft
      &policy_name=LineMenuFDPProv
      &schema=true
      &schema_rules=true HTTP/1.1
> Authorization: Basic YWRtaW5AcHJvdjEuY29tOnBhc3N3b3Jk
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
             libcurl/7.22.0
             OpenSSL/1.0.1
             zlib/1.2.3.4
             libidn/1.23
             librtmp/2.3
> Host: localhost
> Accept: */*
>
  0     0    0    0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
< HTTP/1.1 200 OK
< Server: nginx/1.1.19
< Date: Thu, 30 Jul 2015 15:45:05 GMT
< Content-Type:
   application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-CSRFToken: tey9Z6fdlDtwEMYczJ2UmSleIolfG4ys
< Content-Disposition: attachment;
    filename=relation_LineRelation_exportedsheet_formatted_2015-07-30_17-45-03.xlsx
< Content-Language: en-us
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Allow: GET, HEAD, OPTIONS
< Set-Cookie: fileDownloadToken=downloaded; Path=/
< Set-Cookie: csrftoken=tey9Z6fdlDtwEMYczJ2UmSleIolfG4ys;
             httponly;
             Path=/
< Set-Cookie: sessionid=aioz1ykt36ht47fzthjpljektbg1z1yr;
             httponly;
             Path=/
<
{ [data not shown]
100  9906    0  9906    0     0   3744      0 --:--:--  0:00:02 --:--:--  3745
+ Connection #0 to host localhost left intact
+ Closing connection #0
```

6.8.   Example Transaction

- task: GET transaction instance `tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/`

- user: site administrator

- hierarchy: `55b9dc81a6165413b9d16ab6`

The transaction shows the workflow steps to delete the instance of the `relation/LineRelation`.

The example is from the JSON format of the transaction with detail: "Delete Line Relation".

```
curl -v
    -H 'Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA=='
    'http://localhost/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/
    ?hierarchy=55b9dc81a6165413b9d16ab6
    &nowait=true
    &format=json'
+ About to connect() to localhost port 80 (#0)
+ Trying 127.0.0.1... connected
> GET /api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/
    ?hierarchy=55b9dc81a6165413b9d16ab6
    &nowait=true
    &format=json HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
            libcurl/7.22.0
            OpenSSL/1.0.1
            zlib/1.2.3.4
            libidn/1.23
            librtmp/2.3
> Host: localhost
> Accept: */*
> Authorization: Basic YWRtaW5AbG9jdXMxLmNvbTpwYXNzd29yZA==
>
< HTTP/1.1 202 ACCEPTED
< Server: nginx/1.1.19
< Date: Fri, 31 Jul 2015 11:44:27 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept, Cookie, Accept-Language, X-CSRFToken
< Content-Language: en-us
< Allow: GET, POST, HEAD, OPTIONS
< X-CSRFToken: pcWhI6fzSbevYskrNVcP34JDZOWH6Nti
< Set-Cookie: csrftoken=pcWhI6fzSbevYskrNVcP34JDZOWH6Nti;
            httponly;
            Path=/
< Set-Cookie: sessionid=nyoefznzm1qy9t51qq6v2x0vgkmbvbij;
            httponly;
            Path=/
<
```

Response JSON `data` attribute snippet showing some workflow steps:

```
{

```

<div align="right">(continues on next page)</div>

```
...
  "data": {
    "username": "admin",
    "status": "Success",
    "rolled_back": "No",
    "resource": {
      "hierarchy": "sys.prov1.cust1.locus1",
      "model_type": "relation/LineRelation",
      "current_state": "/api/relation/LineRelation/55ba2482a6165413b9d19fb8/ Entity",
      "pkid": "55ba2482a6165413b9d19fb8"
    },
    "log": [
      {
        "severity": "info",
        "format": "text",
        "log_id": "55ba24bea6165413b9d19fcd",
        "href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/log/
                ?log_id=55ba24bea6165413b9d19fcd",
        "time": "2015-07-30T13:21:02.637000",
        "message": "Step 2 - End",
        "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"
      },
      {
        "severity": "info",
        "format": "text",
        "log_id": "55ba24bea6165413b9d19fcc",
        "href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/log/
                ?log_id=55ba24bea6165413b9d19fcc",
        "time": "2015-07-30T13:21:02.637000",
        "message": "Step 2 - Condition unmet, skipping step. \n[\n
                    ..(SNIPPED)
        "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"
      },
      {
        "severity": "info",
        "format": "text",
        "log_id": "55ba24bea6165413b9d19fcb",
        "href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/log/
                ?log_id=55ba24bea6165413b9d19fcb",
        "time": "2015-07-30T13:21:02.609000",
        "message": "Step 2 - Start update data/InternalNumberInventory\nat hierarchy␣
→level
                    ..(SNIPPED)
        "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"
      },
      {
        "severity": "info",
        "format": "text",
        "log_id": "55ba24bea6165413b9d19fca",
        "href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/log/
                ?log_id=55ba24bea6165413b9d19fca",
        "time": "2015-07-30T13:21:02.605000",
        "message": "Step 1 - End",
        "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"
      },
```

```
      {
        "severity": "info",
        "format": "text",
        "log_id": "55ba24bda6165413b9d19fc5",
        "href": "/api/tool/Transaction/01de8720-d627-4e53-8e1b-e1ad66edb7bd/log/
                ?log_id=55ba24bda6165413b9d19fc5",
        "time": "2015-07-30T13:21:01.280000",
        "message": "Step 1 - Start remove device/cucm/Line\n
         at hierarchy level sys.prov1.cust1.locus1",
        "transaction_id": "01de8720-d627-4e53-8e1b-e1ad66edb7bd"
      },


...
```

7   Backward Compatibility

7.1.   API Backwards Compatibility

7.1.1.   Backwards Compatibility Overview

The VOSS-4-UC API is versioned and while the latest API and models are in use, the system is backwards compatible with earlier API versions for several models and operations on these. This means that API requests follow the schema and data as specified by the API version.

Note: Please note that from release 19.1.1 onwards, if the VOSS-4-UC installation is a *new* installation, in other words *not upgraded*, then the system will not automatically have backwards compatibility support for older API versions.

For such systems, if support for older API versions is required, then the systems will require the separate subsequent installation of version specific backward compatibility files (VSF bundles). Contact your VOSS representative for information on the required VSF bundles.

From 10.6.2 onward, the default web server security protocol has been set to TLSv1.2, and the TSLv1 and SSLv3 protocols are no longer supported.

It is therefore easier to use a network device or custom code written for an earlier VOSS-4-UC version. An HTTP Request parameter or Header change in developer code may be required for API calls.

Third-party clients that are written to use a particular API version continues to work against newer servers as long as the VOSS-4-UC server continues to support the API version used by the client. VOSS-4-UC supports APIs from the current release and two previous releases (N-2).

For device models, an internal version mapping table for API versions and device versions is also maintained. This table creates a fixed mapping of device versions to API version. The mapping is created on a principle of the latest supported device version at the time of the API version release. For example, the version 10.0 UC Application schema is mapped to API version 10.1.2.

No transforms are carried out on Relation models, because transforms are carried out on their component Data- and Device models.

Tools such as Bulk Load and Search, are backwards compatible from an operations and URL structure point of view. However, the data processed or generated by the tools do not support Backwards Compatibility. Data must be adapted to conform to the latest schema when presenting it to the API. Data obtained from the Tool interfaces are interpreted according to the latest schema.

Note: For Update operations: if a model schema was changed so that a new field is added to a schema within a list of objects and data exists in the field for a current instance on the system, then the data in

---

the new field cannot be maintained when updating the instance through a backwards compatible request. Request data replaces all the data within a list of objects. There is currently no workaround for this issue.

7.1.2. API Version

The API Version is represented in major.minor.revision format, for example: 10.1.2, 10.6.1, 10.6.2, or 10.6.3.

The API Version can be seen in the meta section of the resource as follows:

```
"meta": {
 "tags": [],
 "pkid": "",
 "schema_version": "0.1",
 "hierarchy": "sys",
 "version_tag": "0.2",
 "api_version": "10.6.1",
 "model_type": "data/DataModel" },
```

7.1.3. Supported Models and Methods

Supported models are:

- Data Models
- Device Models
- Relations
- Views

The supported HTTP methods on models from the API are:

- GET
- POST
- PUT

7.1.4. Versions Supported

Support for backwards compatibility was introduced in 10.6(1) release.

- The 10.6(2) release is backwards compatible with the 10.6(1) release.
- The 10.6(1) release is backwards compatible with the 10.1(2) release.
- The 10.6(1) release is not compatible with the 10.1(1) release.
- The 10.1(2) release is not compatible with the 10.1(1) release.

7.1.5. Specifying the API Version

Third-party API clients must specify the `api_version` when integrating with VOSS-4-UC 10.6(x). This is required if the third-party client wants to use backwards compatibility, since it ensures that the client continues to receive consistent schemas. Specifying the API version can be done in one of two ways.

In the Query Parameter:

```
GET http://localhost/api/data/Countries/?hierarchy=[hierarchy]
 &schema=true
 &format=json
 &api_version=10.6.1
```

The Query Parameter approach is the recommended method for a client to specify the API Version.

In the Request Header:

```
GET http://localhost/api/data/Countries/?hierarchy=[hierarchy]
&schema=true
&format=json

Request headers
X-Version: 10.6.1
```

7.1.6. Omitting the API Version

If the API Version is omitted, then the following behavior is expected:

- If the URL contains `/v0/`, the 10.1(2) API schemas are used.

- If the URL does not contain `/v0/`, the most recent version API schemas are used.

7.2. Backwards Compatibility Exceptions

10.6(3) Exceptions

Prior to 10.6(3), the SyncTo hierarchy of the user in the Provisioning Status was set to the hierarchy where the User is created. Whenever the user is moved up or down the SyncTo was updated to the hierarchy where it is moved. To move subscribers between Sites, the SyncTo hierarchy behavior has been changed in a backwards incompatible way. The SyncTo Hierarchy focuses on updating the syncTo in ProvisionalStatusDAT when the user changes from a Manual user to a Subscriber user pushed to a Cisco Unified Communications Manager. In 10.6(3) the strategy is changed as follows:

- Manually created Users (User Management not Subscriber Management) have the SyncTo hierarchy set to the hierarchy the user is created.

- Manually created Users (User Management) that are pushed to Cisco Unified Communications Manager (becoming a Subscriber) from Customer or Site has their SyncTo hierarchy updated to that of the Cisco Unified Communications Manager. Note:

  If the user SyncTo hierarchy is above the Cisco Unified Communications Manager hierarchy when it was created, then the SyncTo does not change and remains at the upper hierarchy.

- Manually created Subscribers (Subscriber Management) have there SyncTo hierarchies set to to that of the Cisco Unified Communications Manager.

- LDAP users are not affected (do not have their SyncTo hierarchy changed).

- If a user is created on an intermediate node or site directly (either manually or through QuickAddSubscriber) and then pushed to Cisco Unified Communications Manager, the user is scoped only to the hierarchy where he or she was created.

10.6(1) Exceptions

The following changes in Cisco Unified Communications Domain Manager may impact API backwards compatibility with the previous version:

- Customer Management - Customer name change is now allowed.

- Unicode - Customer names may not have spaces.

- Localization - Language codes are now four characters.

10.1(2) Exceptions

The 10.1(2) release is not backwards compatible with the 10.1(1) release.

11.5(2) Exceptions

From 11.5.2 onwards for `/api/relation/Subscriber`:

- The API URL `cached` parameter on the Subscriber *list* (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will always display cached information and will not refresh the information from the device during a list query with `cached=false`.

- The API URL `summary` parameter on the Subscriber *list* (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will always display summarized information and will not display full CUCM User data with `summary=false`.

- The API parameter `traversal=up` on the Subscriber *list* (`/api/relation/Subscriber/`) will not be honoured. Data presented to the API will default to display resources down the hierarchy tree with `traversal=up`.

The following attribute name changes have been made to the Subscriber (`/api/relation/Subscriber/`) *list* view:

Release 11.5.1 ES1	Release 11.5.2
firstName	firstname
lastName	lastname
phoneProfiles.profileName	extension_mobility
associatedDevices.device	phone1, phone2, phoneN…
summary_device	device
(not present in 11.5.1)	user_type
(not present in 11.5.1)	entitlement_profile
(not present in 11.5.1)	hierarchy_friendly_name
(not present in 11.5.1)	single_number_reach
(not present in 11.5.1)	voicemail
(not present in 11.5.1)	webex

11.5(2) and 11.5(3) Exceptions

Releases 11.5(2) and 11.5(3) are *not* backward compatible for delete operations on `/view/HcsDeleteSiteVIEW/`

For example, given:

- a POST action

- with Request Headers containing *'X-Version': u'11.5.3''*

- to endpoint `/api/view/HcsDeleteSiteVIEW/`

- and Payload:

```
{"site_hn": "<site hierarchy name>",
 "removeMisc2": true,
 "removeMisc1": true,
 "removeDp": true,
 "setCucmUserLocal": true,
 "removeRole": true,
 "removeUsers": true,
 "removeSubMgmt": true,
 "purgeDevices": true,
 "deleteSite": true,
 "purgeUsers": true}
```

the following required fields are not backward compatible:

- `confirm`

- `target_hierarchy`

7.3. API Version Differences

There are some differences in customer facing models in this release with respect to the previous releases. These differences are captured in the CUCDM API Schema Diff document available at cisco.com.

7.4. Backwards Compatibility Table

Important: From release 19.1.1 onwards, API backwards compatibility as shown in the tables below require the installation of a Version Specific File (VSF) bundle that is available from VOSS upon request.

The table below shows CUCDM and VOSS-4-UC Home schema version support according to API request version.

If you are using the release version as in the **Release** column and wish to be compatible with a previous version, then your `&api_version` API request parameter should be referenced from the corresponding **API v.** column.

For example, if your release is CUCDM 11.5.3 and you wish to use the API for CUCDM 10.6.3, then your API request parameter will be `&api_version=10.6.3`.

VOSS-4-UC

Release	API v.not specified	API v.next	API v.19.2.1	API v.19.1.2	API v.19.1.1	API v.11.5.3	API v.11.5.2	API v.11.5.1	API v.10.6.3	API v.10.6.2	API v.10.6.1	API v.10.1.2
Next	Latest	Next schema version	19.2.1	19.1.2	19.1.1	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	
19.2.1	Latest		19.2.1	19.1.2	19.1.1	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	
19.1.2	Latest			19.1.2	19.1.1	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	
19.1.1	Latest				19.1.1	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	
18.1	Latest					11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
17.6	Latest						11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2

CUCDM

Release	API v.not specified	API v.next	API v.11.5.4	API v.11.5.3	API v.11.5.2	API v.11.5.1	API v.10.6.3	API v.10.6.2	API v.10.6.1	API v.10.1.2
Next	Latest	Next schema version	11.5.4	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
11.5.4	Latest		11.5.4	11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
11.5.3	Latest			11.5.3	11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
11.5.2	Latest				11.5.2	11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
11.5.1	Latest					11.5.1	10.6.3	10.6.2	10.6.1	10.1.2
10.6.3	Latest						10.6.3	10.6.2	10.6.1	10.1.2
10.6.2	Latest							10.6.2	10.6.1	10.1.2
10.6.1	Latest								10.6.1	10.1.2
10.1.2	Latest								10.1.2	

Note:

- The next release is where either major or minor digit increments (format is *major.minor.revision*).

- If the API version is not specified, the *latest* schema versions apply to all model instances, except device specific models, where the schema of the device version that is associated to the hierarchy or instance, will be used. Container type models, like Relations, will use the latest definitions to determine the super schema, but the device model primitives inside the Relation will be determined by the device version associated with the hierarchy.

Examples of backward compatibility support:

- 11.5.x will still support two Backwards Compatible releases: 10.6.x and 10.1.x

- The next major release (increment in major or minor number) will see 11.5.x and 10.6.x still supported, but 10.1.x will be removed from support.

7.5. API Backward Compatibility and Import

The VOSS-4-UC system maintains a Data Model version data store containing all versions that have been imported onto the system.

While there is always a current version of a Data Model in use on the system, a check is carried out during the import of data:

- If the current version is newer than the definition of the imported data, then the imported definition data is flagged internally as automigration: false to prevent resources from auto-migrating from a newer version to an older version.

- Importing an older version will not replace the latest definition as the default schema. The older version will only be added to the version store.

The snippet example below shows the automigration attribute:

```
{ "meta": {},
 "resources": [
   { "data": {
     "name": "test_mig_dm"
     ...
     },
     "meta": {
       "hierarchy": "sys",
       "model_type": "data/DataModel",
       "schema_version": "0.1",
       "version_tag": "0.3",
       "automigration": false
     }
   }
 ]
}
```

This model definition version store makes it possible for version definition imports to be sequence independent, allowing a freshly installed system to construct the version history for backwards compatibility.

7.6. HIL API Backward Compatibility

HIL currently supports API versions for v10_1_2 and v10_6_1. You can use the following commands to get the current API version of the target system and to set the API version of the target system:

- show hcs hil target apiversion - To get the current API version of target system

- set hcs hil target apiversion - To set the API version of target system

Note:

If there are any active HIL sessions, CLI will prompt the number of active sessions and will check for confirmation before changing the API version

8   General API Reference

8.1.   Using the API Reference

For each resource, a study of the general Reference for Actions in conjunction with the lists of actions for a resource provides the reference for the resource.

The Field Reference for a resource provides payload details. The list below explains the Field Reference for a resource:

- The field Title is indicated in bold. An asterisk indicates the field is mandatory.
- If the field Type is an array, its the Field Name has a [n] suffix.
- Object and array names are listed to provide the context of fields.
- If a field belongs to an object or an array, the full name is in dot separated notation.
- Where cardinality is shown, the range is [MinItems..MaxItems].
- If a field has a Default value, the value is shown.
- If a field has a Pattern, the regular expression pattern is shown.

In addition, a number of conventions are followed some general guidelines should be noted.

- The full URL includes the host name: `https://[hostname]`; for example, `https://172.29.232.62`
- Variables are enclosed in square brackets.
- [hierarchy] is the hierarchy which can be specified as:
  - UUID (Universally Unique Identifier); for example, `1c012432c0deab00da595101` or
  - In dot notation; for example, `ProviderName.CustomerName.LocationName`

  For a list of available hierarchy UUIDs and their dot notations, refer to the data in the response of the call:

```
GET /api/data/HierarchyNode/?format=json
```

- [pkid] is the ID of the resource instance. Refer to the List action reference for the resource.
- [filename] refers to a file.
- where a custom action (with "+" in the URL) is available, the POST method is used to execute the Provisioning workflow with the name following the "+". For more information, consult the custom workflow section of the API Guide.

Relations, Domain models, and Views may have parameters where the choices are constructed from unexposed models (and that may not be available in the API Reference). You can obtain these choices by using the URL specified in the `target` attribute of the schema of parameter.

To illustrate, below is an extract of the schema for a model called relation/SystemUser that contains a parameter SSOUser, which links a user in the system to a user in an SSO identity provider server. This is done by mapping the SSO user name (sso_username) of the user in the SSO server (sso_idp) to a user in VOSS-4-UC (data/User).

The schema of relation/SystemUser shows that the choices that are available from SSO Identity provider servers are stored in the model data/SsoIdentityProvider. The list of SSO identity providers could be obtained by using the URL in the `target` attribute of the schema.

```
GET /api/relation/SystemUser/?hierarchy=[hierarchy]
  &format=json
  &schema=1
```

The following is an extract of the schema of relation/SystemUser:

```
...

"SSOUser": {
        "items": {
              "type": "object",
              "properties": {
                    "sso_username": {
                          "required": true,
                          "type": "string",
                          "description": "The name identifier that is used for
                            an SSO authenticated user.",
                          "title": "SSO Username"
                    },
                    "sso_idp": {
                          "target": "/api/data/SsoIdentityProvider/choices/
                            ?hierarchy=[hierarchy]
                            &field=entity_id
                            &format=json
                            &auth_token=[auth_token]",
                          "format": "uri",
                          "required": true,
                          "choices": [],
                          "target_attr": "entity_id",
                          "target_model_type": "data/SsoIdentityProvider",
                          "title": "SSO Identity Provider",
                          "type": "string",
                          "description": "The entity id of the SSO Identity
                            Provider."
                    },

 ...
```

8.2. API Schema

The schema for a resource is obtained in the request parameter: ?format=json&schema=1. This way of requesting the schema is only available when requesting an Add form or when viewing a resource.

A specific url is also available for obtaining the schema of a resource:

GET /api/{model_type}/{model_name}/schema/?format=json&hierarchy=pkid

All the schemas are in JSON format.

For more information on the specific API schemas, refer to the Cisco Unified Communications Domain Manager, API Schema Reference Guide

http://www.cisco.com/c/dam/en/us/td/docs/voice_ip_comm/hcs/10_6_3/CUCDM_10_6_3/API_Reference_Docs/Schema1063-Reference-HCM_Standard.docx

8.3. Notifications

Cisco Unified Communications Domain Manager APIs support sending out notifications when instances of certain models are added, deleted or changed. For more details on which models are supported for notifications, and how to configure these notifications please refer to the Cisco Unified Communications Domain Manager, Release 10.6(2) Maintain and Operate Guide

8.4. Meta data

8.4.1. Meta data

The meta data of a resource provides :

- **tags**: List of instance tag names - for tag management, see *API Parameters*.
- **version_tag**: List of version tags - for version tag management, see *API Parameters*.
- **model_type**: The complete model type with name.
- **references**: Information of how this resource relates to other resources
- **summary_attrs**: Summary attributes used for list views.
- **actions**: The actions that can be performed in this resource
- **path**: The hierarchy (business node) path to the existing resource
- **singleton**: If set, instances of the resource can be restricted to one per system or hierarchy.

These are discussed in more detail below.

8.4.2. References

References in the system represent the reference of an entity in the system as Hypermedia as the Engine of Application State (HATEOS). Each reference position is represented by an object pair pkid and href.

- **device**: A list of one device that relates the resource to a device resource in the system.
- **owner**: The owner reference would exist if the current resource was created by a Domain Model (feature model)
- **self**: A list of containing a reference to the current resource.

- **parent** (reserved for hierarchy): An entry containing the parent in the hierarchy: zero for root node, one for other resources.

- **children** (reserved for hierarchy): A list of zero or more children in the hierarchy tree below the resource.

For example:

```
"references": {
    "device": [{
        "pkid": "",
        "href": ""
    }],
    "owner": [{
        "pkid": "",
        "href": ""
    }],
    "self": [{
        "pkid": "5135fc0f31790a3000a83b2b",
        "href": "/api/data/CallManager/5135fc0f31790a3000a83b2b"
    }],
    "children": [],
    "parent": [{
        "pkid": "5135fb8331790a2ffee7d7ab",
        "href": "/api/data/HierarchyNode/5135fb8331790a2ffee7d7ab"
    }]
}
```

8.4.3. Summary Attributes

For resources that will be displayed on the GUI as a summarized list, data fields can be selected for this list. Members of the **summary_attr** list identify:

- **title** as the list column header on the GUI display

Note: If the default Field Display Policy for the resource contains a value, this will be displayed in the column header.

- **name** as the resource field to show in the column of the list

For example:

```
summary_attrs: [
  {
   title: "Name"
   name: "name"
  }
  {
   title: "Description"
   name: "description"
  }
]
```

8.4.4. Path

The **path** object in the meta of a specific resource contains the list of parent pkid values in the hierarchy sequence to which the resource belongs. This list represents the navigation path from the root node of the hierarchy to the specific resource.

Example

```
path: [
 "50c1e21fa61654441dd8edc4",
 "50c1e2a2a61654441eaebcf8",
 "50c1e2a4a61654441eaebcfe",
 "50c1e2a6a61654441eaebd01"
]
```

In the example above, the **path** to the current resource node is:

```
"50c1e21fa61654441dd8edc4" (ancestor)

--> "50c1e2a2a61654441eaebcf8" (ancestor)

    --> "50c1e2a4a61654441eaebcfe" (parent)

        --> "50c1e2a6a61654441eaebd01" (current node)
```

8.4.5. Model Type

Model type is referenced from **model_type** in the schema. The reference is to the type of model and the model name - see: model_type .

Example

`"data/CallManager"` in

```
{"meta":
      {"model_type": "data/CallManager",
      ...
      },
}
```

8.4.6. Actions

The actions in the metadata of a resource schema provide HTTP method calls to a resource for a number of purposes. The available actions in the schema depend on whether the call request is:

- to a *specific* resource, i.e. pkid is specified in the call, or

- to the resource *in general*, i.e. no pkid in the call

- the actions contain a schema property to indicate which requests will support asynchronous transaction handling. This behavior is controlled by the `nowait` parameter in the URL.

8.4.7. Singleton

The Data model type resource called `data/Datamodel` has an attribute `singleton` that can take any of 3 values:

- None (default): no singleton constraint

- `system`: a system singleton that only allows one instance throughout the system

- `hierarchy`: a hierarchy singleton that only allows one instance every hierarchy

For example, the snippet below shows a simple `data/Datamodel` instance called `LoginBanner` that can itself only have one instance per hierarchy:

```
"data": {
    "doc": "doc",
    "Meta": {
        "operations": [
            "add",
            "clone",
            "export",
            "export_bulkload_template",
            "get",
            "help",
            "move",
            "list",
            "migration",
            "transform",
            "remove",
            "tag",
            "tag_version",
            "update",
            "field_display_policy"
        ],
        "summary_attrs": [
            "login_banner"
        ],
        "singleton": "hierarchy",
        "attr_props": [
            {
                "title": "Login Banner",
                "required": true,
                "type": "string",
                "displayable": true,
                "name": "login_banner"
            }
        ]
    },
    "name": "LoginBanner"
}
```

8.5. Generic Actions

8.5.1. Choices

Format:

```
GET http://<server_address>/api/<resource_type>/<resource_name>/choices/
 ?hierarchy=[hierarchy]
 &format=json
```

Action	choices
Description	Get a list of resource instances at a hierarchy as value-title pairs. Requires a business key in the resource model definition.
Method	GET
URL	/api/<resource_type>/<resource_name>/choices/ or without resource specification: /api/choices/
Parameters	hierarchy=[hierarchy], format=json, pagination parameters (see API Parameters), filter parameters (see Filter Parameters for Choices).
Response	A JSON payload with: • pagination details • meta information: query, list of instance references with `pkid` and `href` of data • choices data: list of value-title pairs of the business keys. On the GUI choices list, the response title displays, while the value is returned. Filter parameters can modify this standard behavior - see Filter Parameters for Choices.
support_async	false

Example:

• Request

```
GET http://<server_address>/api/data/Countries/choices/
 ?hierarchy=[hierarchy]
 &format=json
```

• Response

```
HTTP 200 OK
Vary: Accept
X-Request-ID: 9bcd77b4cd27dccd0f18a1d8d22e7ddab85aa848
Content-Type: text/html; charset=utf-8
Allow: GET, HEAD, OPTIONS
Response-Content:
{

    pagination : {
        direction :  asc,
        maximum_limit :  2000,
        skip :  0,
        limit :  0,
        total_limit : null,
        total :  37
```

(continues on next page)

```
    },
    meta : {
        query :  /api/data/Countries/choices/,
        references : [
            {
                pkid :  5a16c3c68963f91b84baf357,
                href :  /api/data/Countries/5a16c3c68963f91b84baf357/
            },
            ...
        ]
    },
    choices : [
        {
            value :  ["Australia", "AUS", "hcs"],
            title :  ["Australia", "AUS", "hcs"]
        },
        ...
```

8.5.2. Add

Action	add
title	Get the GUI Add form.
method	GET
URL	/api/<resource_type>/<resource_name>/add/
Parameters	hierarchy=[hierarchy], format=json
Response	The schema of <resource_type>/<resource_name> as JSON
support_async	false
class	add

When adding the `&schema=1` parameter, the response contains the schema of the payload for the Create action.

The schema required to add the resource may be different from the schema that is used to obtain the details of the resource. Refer to the schema of the GUI Add form.

The actions in the response shows the URL for the POST API call to create an instance (see Create action).

For example, the request below shows the required details. (Using variables [hierarchy])

Request:

```
GET /api/data/AccessProfile/add/
  ?hierarchy=[hierarchy]
  &format=json
  &schema=1
```

Response snippet - POST call:

```
"create": {
    "class": "add",
    "href": "/api/data/HierarchyNode/?hierarchy=[hierarchy]",
```

```
    "method": "POST",
    "support_async": true,
    "title": "Create"
}
```

Response snippet - schema:

```
"schema": {
    "$schema": "http://json-schema.org/draft-03/schema",
    "properties": {
        "description": {
            "description": "A general description for the hierarchy node.",
            "title": "Description",
            "type": "string"
        },
        "name": {
            "description": "The name by which this hierarchy node will
              be known.",
            "pattern": "^[A-Za-z0-9_\\- ]+$",
            "required": true,
            "title": "Name",
            "type": "string"
        },
        "node_type": {
            "choices": [],
            "description": "A type label for this node which refers
              to a Hierarchy Node Type.",
            "format": "uri",
            "is_password": false,
            "items": {
                "is_password": false
            },
            "readonly": false,
            "required": false,
            "target": "/api/data/HierarchyNodeType/choices/?hierarchy=[hierarchy]",
            "target_attr": "name",
            "target_model_type": "data/HierarchyNodeType",
            "title": "Hierarchy node type",
            "type": "string"
        }
    },
    "schema_version": "0.1",
    "type": "object"
},
```

8.5.3. Bulk Update

Action	bulk_update (1)
title	First task: Obtain the URL and schema needed to construct the payload to modify the resource.
method	GET
URL	/api/<resource_type>/<resource_name>/bulk_update/
Parameters	hierarchy=[hierarchy], format=json
Response	Returns: POST call that is used to update the instances of resource schema that is used to create POST payload.
support_async	false
class	update

Action	bulk_update (2)
title	Second task: Perform the bulk modify on the required list of instances each with [pkid] with the payload constructed from the schema in the first task.
method	POST
URL	/api/<resource_type>/<resource_name>/bulk_update/
Parameters	hierarchy=[hierarchy], format=json
Payload	Contains update data and instance pkids to update.

Use the GET request to obtain a list of instance pkids to select for the bulk update:

```
GET /api/<resource_type>/<resource_name>/?hierarchy=[hierarchy]&format=json
```

As an example, the request below shows the required details for a particular model. (Using variables [hierarchy], [pkid1], [pkid2])

Request for POST call and schema:

```
GET /api/data/AccessProfile/bulk_update/
  ?hierarchy=[hierarchy]
  &schema_rules=true
  &format=json
  &schema=1
```

Response snippet - POST call:

```
"bulk_update": {
    "class": "update",
    "href": "/api/data/AccessProfile/bulk_update/?hierarchy=[hierarchy]",
    "method": "POST",
    "support_async": true,
    "title": "Bulk Modify"
}
```

Response snippet - schema:

```
"schema": {
    "$schema": "http://json-schema.org/draft-03/schema",
    "properties": {
        "description": {
            "description": "A description for the Access Profile.",
            "required": false,
            "title": "Description",
            "type": "string"
        },
        "full_access": {
            "description": "Enabling this flag, grants the user full
                system access.",
            "required": false,
            "title": "Full Access",
            "type": "boolean"
        },
        "miscellaneous_permissions": {
            "description": "The list of miscellaneous operations permitted by
                this Access Profile.",
            "items": {
                "choices": [
                    {
                        "title": "",
                        "value": ""
                    }
                ],
                "type": "string"
            },
            "required": false,
            "title": "Miscellaneous Permissions",
            "type": "array"
        },
        "name": {
            "description": "The name that is given to the Access Profile.",
            "required": false,
            "title": "Name \*",
            "type": "string"
        },
        "type_specific_permissions": {
            "description": "The list of types that are permitted by
                this Access Profile.",
            "items": {
                "properties": {
                    "operations": {
                        "description": "The operations that are permitted by this Access
                            Profile for the given type.",
                        "items": {
                            "choices": [
                                {
                                    "title": "",
                                    "value": ""
                                }
                            ],
                            "type": "string"
                        },
                        "required": false,
                        "title": "Permitted Operations \*",
```

```
          "type": "array"
        },
        "type": {
          "choices": [],
          "description": "The type that is permitted by this Access Profile.
            This field supports the use of the * wildcard.",
          "format": "uri",
          "required": false,
          "target": "/api/choices/?hierarchy=[hierarchy]&format=json",
          "target_model_type": "",
          "title": "Permitted Type \*",
          "type": "string"
        }
      },
        "type": "object"
    },
    "required": false,
    "title": "Type Specific Permissions",
    "type": "array"
  }
},
"schema_version": "0.1.8",
"type": "object"
},
```

Example POST request to carry out the update:

```
POST /api/data/AccessProfile/bulk_update/
  ?hierarchy=[hierarchy]
  &nowait=true
  &format=json
```

Payload example - bulk updating the description of instances having pkid1, pkid2 with the string "profile":

```
{"data":{"description":"profile"},
 "meta":{
   "references":{
     "form_href":"/api/data/AccessProfile/bulk_update/
       ?hierarchy=[hierarchy]
       &schema=
       &schema_rules=true"
   }
 },
 "request_meta":{
   "hrefs":[
     "/api/data/AccessProfile/[pkid1]",
     "/api/data/AccessProfile/[pkid2]"
   ]
 }
}
```

8.5.4. Clone

Action	clone
Title	Clone instance with [pkid]. The schema rules are applied.
Method	GET
URL	/api/<resource_type>/<resource_name>/[pkid]/clone/
Parameters	hierarchy=[hierarchy], schema=true, schema_rules=true
Response	A JSON payload with: A POST action URL. The unchanged model [pkid] payload to be modified to create a new instance.
support_async	false
Class	clone

• For the instance pkids that can cloned, refer to the List GET call for the resource:

```
GET http://<server_address>/api/<resource_type>/<resource_name>/
 ?hierarchy=[hierarchy]
 &format=json
```

• Use the POST action in the response and a modification of the response as the payload to create the clone of the the instance with pkid.

8.5.5. Configuration Template

Action	configuration_template
title	Obtain the schema and URL needed to create a Configuration Template instance for the resource.
method	GET
URL	/api/<resource_type>/<resource_name>/configuration_template/
Parameters	hierarchy=[hierarchy]
Response	POST call to create the schema for the configuration template of specified resource.
support_async	false
class	config

• Use the returned properties to create the POST payload data for the specified resource.

• For a details on what to add to the POST payload, see the schema in the response.

• The POST call is of the format:

```
POST http://<server_address>/api/data/ConfigurationTemplate/
  ?hierarchy=sys
```

For example, the request:

```
GET /api/data/AccessProfile/configuration_template/
  ?hierarchy=[hierarchy]
```

```
&format=json
&schema=true
&schema_rules=true
```

The response includes the required POST call:

```
"create": {
  "class": "add",
  "href": "/api/data/ConfigurationTemplate/?hierarchy=[hierarchy]",
  "method": "POST",
  "support_async": true,
  "title": "Create"
}
```

The response includes the Configuration Template schema for the relevant model. The `template` property of the schema applies to the relevant model. This schema is used to create a payload for the POST.

A simple example of a payload containing a Configuration Template for a model `data/AccessProfile` with name "CFT1" that adds a value to the Access Profile description "Access Profile for:":

```
{"data":
 {"name":"CFT1",
  "target_model_type":"data/AccessProfile",
  "merge_strategy":"additive",
  "template":{
    "description":"Access Profile for:"}
 },
 "request_meta":{},
 "meta":{
    "references":{
      "form_href":"/api/data/AccessProfile/configuration_template/
        ?hierarchy=[hierarchy]"
    }
  }
}
```

8.5.6. Create

Action	create
title	Create an instance of a resource.
method	POST
URL	/api/<resource_type>/<resource_name>/
Parameters	hierarchy=[hierarchy]
Payload	See add schema of the object for the payload specification
support_async	true
class	add

Response is a pkid of the created instance.

To obtain the schema of the resource, use the GET request:

```
GET /api/<resource_type>/<resource_name>/
?hierarchy=[hierarchy]
&format=json
```

To apply a configuration template when creating the resource, use the parameter `&configuration_template` with its value the name of an existing Configuration Template. For details on the parameter, refer to the topic on API parameters.

8.5.7. Delete

Action	Delete
title	Delete instance with [pkid]
method	DELETE
URL	/api/<resource_type>/<resource_name>/[pkid]/
Parameters	hierarchy=[hierarchy]
Payload	N/A
support_async	true
Class	delete

Action	Bulk Delete
Title	Bulk delete [pkid1],[pkid2]...
Method	DELETE
URL	/api/<resource_type>/<resource_name>/
Parameters	hierarchy=[hierarchy]
Payload	{"hrefs":["/api/<resource_type>/<resource_name>/[pkid1]", "/api/<resource_type>/<resource_name>/[pkid2]", ...]}
support_async	true
Class	delete

For the instance pkids [pkid1],[pkid2], ... that can be added to the DELETE call payload (the instance pkids to delete), use to the GET call for the resource.

```
GET /api/<resource_type>/<resource_name>/
  ?hierarchy=sys
  &format=json
```

8.5.8. Execute

Action	Execute (instance)
Title	Execute
Method	POST
URL	/api/<resource_type>/<resource_name>/execute/
Parameters	hierarchy=[hierarchy]
Support Async	true
Class	execute

For example, execute a data synchronization action for a device. In this case, the call would be:

```
POST /api/data/DataSync/[pkid]/execute/
  ?hierarchy=[hierarchy]
  &nowait=true
  &format=json
```

And for a device `data/CallManager`, the payload would be similar to:

```
{
    "data": {
        "asynchronous": false,
        "device_type": "data/CallManager",
        "model_type_list": "minimum CUCM models",
        "name": "minimum CUCM models",
        "refresh_existing_data": true,
        "sync_type": "pull"
    },
    "meta": {
        "references": {
            "form_href": "/api/data/DataSync/[pkid]/
             ?hierarchy=[hierarchy]"
        }
    },
    "request_meta": {}
}
```

8.5.9. Export

Action	export (instance)
Task	Get a selected [export_format] of the schema and a single instance with [pkid] of <resource_type>/<resource_name>; optionally with tag_version at [version] and Configuration Template as [configtemplate].
Call	GET
URL	/api/<resource_type>/<resource_name>/export/[pkid]/
Parameters	hierarchy=[hierarchy], version=[version], export_format=[raw_xlsx|xlsx|json], schema=, schema_rules=, template_name=[configtemplate]
Response	The response is an attachment: a compressed zip of the JSON file
support_async	false
Class	export

For `export_format=raw_xlsx`, the response is a "raw" MS Excel spreadsheet with columns corresponding to the JSON format export and response format:

```
Content-Disposition: attachment;
  filename=<resource_type>_<resource_name>_exportedsheet_CCYY-MM-DD_HH-MM-SS.xlsx
Content-Language:en
Content-Type:
  application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

For `export_format=xlsx`, the response is a MS Excel spreadsheet, formatted to show all columns and response format:

```
Content-Disposition: attachment;
  filename=<resource_type>_<resource_name>_exportedsheet_formatted_CCYY-MM-DD_HH-MM-SS.
→xlsx
Content-Language:en
Content-Type:
  application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

For `export_format=json`, the response is a time stamped zip file of data in JSON and a response format:

```
Content-Disposition: attachment;
  filename=export_CCYY-MM-DD_HH:MM:SS.MS.json.zip
Content-Language:en
Content-Type:application/x-zip
```

The XLSX format can be used to bulk load instances of the resource and the JSON format can be used to import instances of the resource.

Action	Bulk Export
Title	Get a selected [export_format] the schema and instances [pkid1], [pkid2],... of the resource; optionally with tag_version at [version] and Configuration Template as [configtemplate].
Method	POST
URL	/api/<resource_type>/<resource_name>/export/
Parameters	hierarchy=[hierarchy], version=[version], export_format=[raw_xlsx|xlsx|json], schema=, schema_rules=, template_name=[configtemplate]
Payload	{"hrefs": ["/api/<resource_type>/<resource_name>/[pkid1]", "/api/<resource_type>/<resource_name>/[pkid2]", ...]}
support_async	true
Class	export

For `export_format=raw_xlsx`, the response is a MS Excel spreadsheet and response format:

```
Content-Disposition: attachment;
  filename=<resource_type>_<resource_name>_exportedsheet_CCYY-MM-DD_HH-MM-SS.xlsx
Content-Language:en
Content-Type:
  application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

For `export_format=xlsx`, the response is a MS Excel spreadsheet and response format:

```
Content-Disposition: attachment;
  filename=<resource_type>_<resource_name>_exportedsheet_formatted_CCYY-MM-DD_HH-MM-SS.
→xlsx
Content-Language:en
Content-Type:
  application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

For `export_format=json`, the response is a time stamped export zip file and a response format:

```
Content-Disposition: attachment;
  filename=export_CCYY-MM-DD_HH:MM:SS.MS.json.zip
Content-Language:en
Content-Type:application/x-zip
```

The XLSX format can be used to bulk load instances of the resource and the JSON format can be used to import instances of the resource.

To get the list of all instance pkids [pkid1],[pkid2], ..., use the List action of the resource:

```
GET http://<server_address>/api/<resource_type>/<resource_name>/
  ?hierarchy=sys
```

8.5.10. Export BulkLoad Template

Action	export_bulkload_template
Title	Get a compressed file of the Bulk Load spread sheet template for the resource, optionally with a Field Display Policy as [policy] or Configuration Template as [configtemplate].
Method	POST
URL	/api/<resource_type>/<resource_name>/export_bulkload_template/
Parameters	hierarchy=[hierarchy], policy_name=[field_display_policy], template_name=[configtemplate], schema=, schema_rules=
Response	The response is an attachment of the format: <resource_type>_<resource_name>_bulkloadsheet.xlsx
support_async	true
Class	export

Example request:

```
POST /api/data/DATA1/export_bulkload_template/
 ?hierarchy=[hierarchy]
 &template_name=[configtemplate]
 &policy_name=[field_display_policy]
 &schema=
 &schema_rules=
 &format=json
```

Example response:

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Mon, 09 Mar 2015 15:13:06 GMT
Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept, Cookie, Accept-Language
Content-Language: en-us
Allow: POST, OPTIONS
Content-Disposition: attachment; filename=data_DATA1_bulkloadsheet.xlsx
```

The returned spreadsheet will reflect the applied Configuration Template and Field Display Policy as indicated in the POST parameters.

8.5.11. Field Display Policy

Action	field_display_policy
title	Obtain the schema and URL needed to create a Field Display Policy instance for the resource.
method	GET
URL	/api/<resource_type>/<resource_name>/field_display_policy/
Parameters	hierarchy=[hierarchy]
Response	Field Display Policy schema and rules that include the POST and a reference to the target model used to create the field display policy for the resource.
support_async	false
class	display_policy

- Use the action in the response to create the POST payload for the specified resource.

The response snippet below shows the POST method to create the Field Display Policy:

```
"meta": {
  "actions": [
    {
      "create": {
        "class": "add",
        "href": "/api/data/FieldDisplayPolicy/
         ?hierarchy=[hierarchy]
         &policy_name=[field_display_policy]",
        "method": "POST",
        "support_async": true,
        "title": "Create"
         }
 ...
```

- The Field Display Policy schema in the response shows properties to add to the POST payload.

Example POST payload for target model `data/DATA1` (from an Admin Portal form with `[form_FDP]` and `[form_CFG]` applicable):

```
{
  "data": {
    "field_overrides": [
      {
       "field": "name",
       "help_text": "Help Name"
      }
    ],
    "groups": [
      {
       "fields": [
         "name",
         "surname"
        ],
       "title": "G1"
      }
```

(continues on next page)

```
    ],
    "name": "FDP2",
    "target_model_type": "data/DATA1"
  },
  "meta": {
    "references": {
      "form_href": "/api/data/DATA1/field_display_policy/
      ?hierarchy=[hierarchy]
      &policy_name=[form_FDP]
      &template_name=[form_CFG]"
    }
  },
  "request_meta": {}
}
```

8.5.12. Help

Action	help
Title	Get the on-line Help for the resource.
Method	GET
URL	/api/<resource_type>/<resource_name>/help
Parameters	hierarchy=[hierarchy]
Response	On-line help of the resource as HTML
support_async	false
class	help

8.5.13. List

Action	list
title	List the resources of a given type in the system.
method	GET
URL	/api/<resource_type>/<resource_name>/
Parameters	hierarchy=[hierarchy], format=json. The schema is returned irrespective of whether the parameter `&schema=true` is sent.
Response method	The <resource_type>/<resource_name> schema and all instances in JSON format.
support_async	false
class	list

8.5.14. Update

Action	update (instance) (same for modify)
title	Modify an instance of a resource
method	PUT
URL	/api/<resource_type>/<resource_name>/[pkid]
Parameters	hierarchy=[hierarchy]
Payload	See the schema of the resource for the payload specification
support_async	true
class	update

The update action replaces current resource values with the payload values. The payload should contain the all the attributes in schema.

The response is a pkid of the updated instance.

To apply a configuration template when creating the resource, use the parameter `&configuration_template` (for further information on the API parameter, see the API Guide).

8.6. Custom Device Connection Actions

8.6.1. Import

Action	import (instance)
Title	Execute
Method	POST
URL	/api/<resource_type>/<resource_name>/import/
Parameters	hierarchy=[hierarchy]
Support	Async
Class	import

For example, a full synchronization of the VOSS-4-UC cache with a device.

8.6.2. Test Connect

Action	test_connect (instance)
Title	Test Connection
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/test_connect/
Parameters	hierarchy=[hierarchy], format=json
Support Async	true
Class	test_connection

8.7. Custom Device Actions

8.7.1. Apply

Action	apply
Title	Apply
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+apply/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/[pkid]/+apply/schema/
Support Async	true
Class	custom

8.7.2. Assign

Action	assign
Title	Assign
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+assign/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+assign/schema/
Support Async	true
Class	custom

For example, device/cucm/PresenceUser

8.7.3. Do

Action	do
Title	.
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+do/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+do/schema/
Support Async	true
Class	custom

Example resources:

- cucm/AuthenticateUser

- cucm/DeviceLogin

- cucm/DeviceLogout

- etc...

8.7.4. Lock

Action	lock
Title	Lock
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+lock/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+lock/schema/
Support Async	true
Class	custom

For example, a cucm phone.

8.7.5. Promote

Action	promote
Title	Promote
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+promote/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+promote/schema/
Support Async	true
Class	custom

8.7.6. Reset

Action	reset
Title	Reset
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+reset/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+reset/schema/
Support Async	true
Class	custom

8.7.7. Vendor Config

Action	vendor_config
Title	Vendor Config
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+vendor_config/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+vendor_config/schema/
Support Async	true
Class	vendor_config

8.7.8. Wipe

Action	wipe
Title	Wipe
Method	POST
URL	/api/<resource_type>/<resource_name>/[pkid]/+wipe/
Parameters	hierarchy=[hierarchy], format=json
View	/api/<resource_type>/<resource_name>/+wipe/schema/
Support Async	true
Class	custom

8.7.9. Update LDAP Authentication

Action	update_ldap_auth
Title	Set up device/cucm/LdapAuthentication before import. Also use this call for update.
Method	POST
URL	/api/device/cucm/LdapAuthentication/+update_ldap_auth/
Parameters	hierarchy=[hierarchy], format=json
Support Async	true
Class	custom

For payload, see device/cucm/LdapAuthentication schema.

8.7.10. Update LDAP System

Action	update_ldap_system
Title	Set up device/cucm/LdapSystem before import. Also use this call for update.
Method	POST
URL	/api/device/cucm/LdapSystem/+update_ldap_system/
Parameters	hierarchy=[hierarchy], format=json
Support Async	true
Class	custom

For payload, see device/cucm/LdapSystem schema.

8.8. Other elements

8.8.1. Data

The **data** of a resource is an object containing all the required and set fields of a model.

The data instance shows names as they are defined in the schema of the resource while the values of the names are contained in the instance.

Example of a single data instance of a resource of model type `data` and model name `CallManager`.

```
data: {
    iso_country_code: "AUS"
    pkid: "51ef319c746fae3622c710e4"
    pstn_access_prefix: "9"
    service_access_prefix: "13"
    default_user_locale: "English United States"
    network_locale: "United States"
    premium_access_prefix: "8"
    international_access_prefix: "011"
    country_name: "Australia"
    international_dial_code: "61"
    emergency_access_prefix: "000"
    national_trunk_prefix: "0"
    hierarchy_path: "sys"
}
```

8.8.2. Resources

The **resources** object in a resource list is represented as a list of objects containing meta and data of resources of the requested model type and model name.

Below and example of a **resources** object outline.

```
"resources": [{
    "meta": { ...  },
    "data": { ...  }
},
{
    "meta": { ...  },
    "data": { ...  }
},
...
]
```

8.8.3. Schema

You can obtain the schema for a resource in the request parameter: `?format=json&schema=true`. This way of requesting the schema is only available when requesting an Add form or when viewing a resource.

A specific URI is also available for obtaining the schema of a resource:

GET /api/(str:model_type)/(str:model_name)/schema/?format=json&hierarchy=pkid

The JSON schema uses the IETF draft-zyp-json-schema-03 (http://tools.ietf.org/html/draft-zyp-json-schema-03 and https://github.com/json-schema/json-schema)

The schema provides:

- the properties of a field for each object in the schema describing the data of a resource:

  - **$schema**: The schema URI, currently http://json-schema.org/draft-03/schema.

  - **title**: This is the default field name.

  - **required**: The property and value **true** value is a property if the field is mandatory

  - **type**: The data type of the field. See the definitions and conventions in use. If the data type is:

    * object, then the object itself has a **schema**

    * array, then it has the property **items**

  - **format**: if the **type** is string, a further format of the string can be selected.

  - **choices**: In the case that the data element takes a value from a specified list of values.

  - **target**: where a resource is linked to another resource, this resource is indicated as the **target**.

  - **target_attr**: the specific attribute of the **target**.

  - **attr_props**: an object containing the list of properties of each attribute.

  - **minItems**: minimum number if the data type is an array.

  - **maxItems**: maximum number if the data type is an array.

  - **items**: the specified items if the data type is an array.

  - **documentation** and **description**: text content to document and describe the object.

  - **name**: name of the resource. [a-zA-Z_] characters are allowed

  - **default**: default value, if specified.

  - **valid_re**: the regular expression that a **string** data type should adhere to.

Example

Refer to the example data as in Data. The schema for the example data provides properties of each element:

```
schema: {
 $schema: "http://json-schema.org/draft-03/schema"
 type: "object"
 properties: {
 host: {
        required: true
        type: "string"
        title: "Host Name"
 }
 username: {
        required: true
        type: "string"
        title: "Admin Username"
 }
 password: {
        required: true
        type: "string"
        title: "Admin Password"
 }
 version: {
        target: "/api/data/CallManagerVersion/choices/
          ?hierarchy={hierarchy}&field=version&format=json"
        format: "uri"
```

(continues on next page)

```
        title: "Version"
        choices: [ ]
        target_attr: "version"
        type: "string"
}
port: {
        type: "string"
        title: "Port"
}
import: {
        type: "boolean"
        title: "Immediate Import"
}
data_sync: {
        target: "/api/data/DataSync/choices/
          ?hierarchy={hierarchy}&field=name&format=json"
        title: "Data Sync"
        format: "uri"
        choices: [ ]
        target_attr: "name"
        type: "string"
}
}
}
```

8.8.4. Pagination

This object contains the **pagination** information of a resource list. Three fields always exist in this object:

- **skip**: The offset index into the total resource list to be skipped

- **limit**: The number of resource to be returned

- **total**: The total number of resources that exist on the system/hierarchy. If a GET call was made with the parameter `count=false`, then this value will be 0.

Example of a pagination object within a list response.

```
"pagination": {
    "skip": 0,
    "total": 25,
    "limit": 250
}
```

Index

V

voss
    voss set_debug, 37