SYNERGY SKY

Version: 5-3868 Last Updated: 03.08.2014

API Documentation

Provisioning Framework Management v1

Contents

1 How to read this document and Scope

This is the documentation for the official Provisioning Framework Management v1 API

Note
API is still under construction which means the API may be subject to change

This document displays how the different service path works, and how to interact with them. It also gives a brief introduction on how to integrate towards it, however it do require extensive knowledge from the integrator in building application towards web-services. There is no intention on delivering any best-practises in terms of integration, in this document. A Best practices is highly depending on scale of service and how extensive the Synergy Sky solution is being used in the complete service.

However, each service part are described with what webservice operation which are supported aswell as what parameters to send in and what to expect back. If a section is described with a GET service which only have response available as sub-chapter, it implies that the GET does not have any Request parameters.

Same goes with if there is no Response chapter, it implies that there are no response from the API, except the HTTP Response codes.

2 Caveats

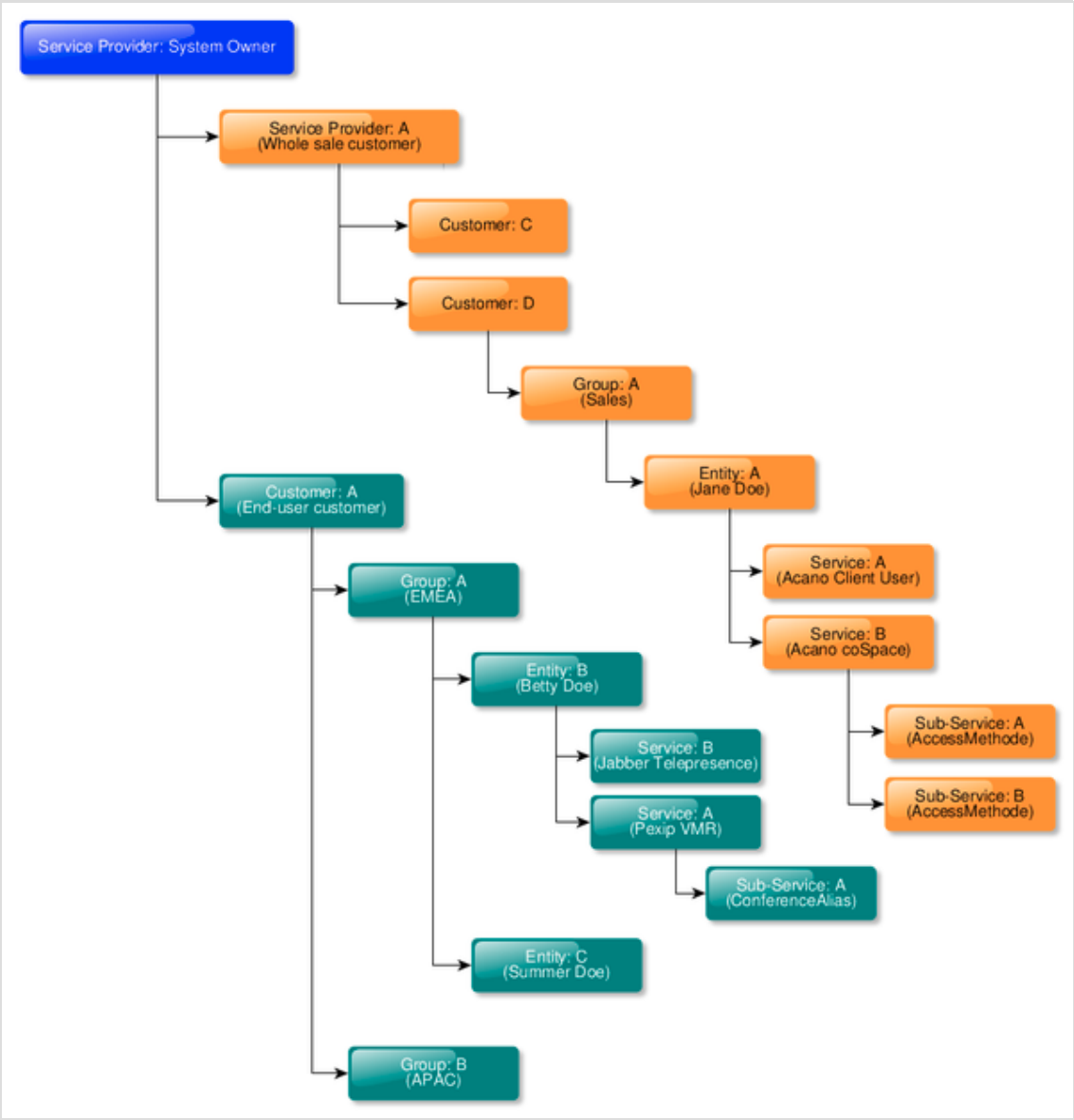
Case ID	Title	Impact	Importance
TBD	commit Does not replay any tracking ID	Does not allow integrator or UI to "know" when the committed job is done. In larger jobs (n*1000 objects, external provisioning can be taking massive amount of time, which means that it needs ability to be tracked.	High

3 API Concept

This API is loosely based around a REST-like[1] interfacing. This translate into a set of paths or service URLs, which are given available to the user of the API.

3.1 Service Providers, Customers and Groups

The root stricture of Synergy Sky is build around LDAP structure. Which consists of multiple levels. This allows the system owner to create a customized structure for the end-users of the system, which again will allow for granulated report and views. The different tree-structure object that exists is Service Providers, Customers and Groups. They can be arranged in certain way to achieve a structure of choice.



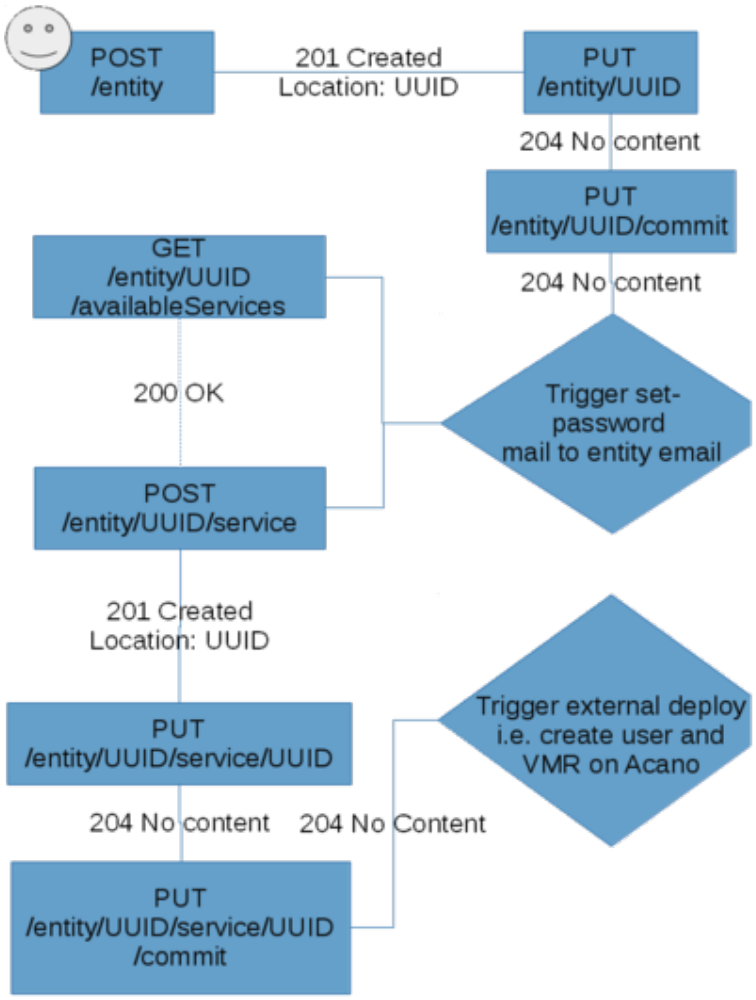
Example: A customer can have three departments; EMEA, APAC and Americas. The customer want each of the regions to be billed separate and directly to their business units. By separate the entries directly into their group, we can allow for this to happen. All data and "asset"-information are connected to their respective sub-group under a customer. And the customer is connected to a service provider (see fig 1).

This container structure is heavily used throuhgout the system, however it might be partly ignored, if there is no need for the granularity. In that case a simple customer -> Group -> Entity structure will be sufficient. This is entirly up to the system owner.

3.2 Entities and Services

When actually deploying the service - which as an example, can be an Acano VMR and/or Acano Client Login. you deploy that service under an entity/user. This is the personal owner of that specific service with a specific set of credentials. To be able to deploy a service you need an entity first. A step by step list of deploying a Acano client login user would be as follow:

TODO
Need to be updated to reflect new methodes



4 Standard Communication Types

The standard methods for interacting with the API are following the HTTP protocol where the method name are used against the object path for interaction.

Method	Action	Description
POST	Create	Create a new object in the system
PUT	Update	Modify existing target
GET	Request/Get	Get existing information
DELETE	Delete / Remove	Delete object from the system

Each API function will be documented according to Method type.

4.1 Authentication

Synergy Sky 3.0 implements only Basic Access Authentication [2] as form of authentcation. The authencation and session state is happening server side with a expiretime on 30 minutes, which allows the integrator to be reusing connection for infinite amount of time, as long as its keept refreshed.

To authenticate client from i.e. C# .Net, following class is called:

```
client.Authenticator = new HttpBasicAuthenticator(Username, Password);
```

Similar type of object can be created when using Python

```
requests.get('<api url>', auth=HTTPBasicAuth('user', 'pass'))
```

From wikipedia.org[3]:

Note

When the user agent wants to send the server authentication credentials it may use the Authorization header. The Authorization header is constructed as follows: Username and password are combined into a string **username:password**. The resulting string literal is then encoded using the RFC2045-MIME variant of Base64, except not limited to 76 char per line. The authorization method and a space i.e. **"Basic "** is then put before the encoded string. For example, if the user agent uses **"Aladdin"** as the username and **"open sesame"** as the password then the header is formed as follows: **Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==**

The authentication accounts is setup according to the installation guide

5 HTTP Response codes

HTTP Code	Name	Description
200	OK	Returned from a successful GET, DELETE and PUT
201	Created	Returned from a successful POST
204	No Content	Returned from a successful POST
400	Bad Request	Returned from a request had invalid or lacking of mandatory parameters
500	Internal Service Error	Returned from a request that failed, some invalid parameters do return 500 instead of 400 like requesting an UUID which do not exist in database

6 Versioning of API

The API is made available under the /api namespace of the server, this is followed by version requested API. Initially this is only /v1 However in the future breakable changes to the API - which will render current integration invalid will be published under an incremental API version.

Changes to the API that might not get a new version are

- Additional API service path that extends current functionality, but not alter existing.
- extended set of parameters that extends existing service paths, without breaking existing behavior

Changes to the API that do get a new version are

- Altered behavior of existing service path

7 Common usage

A short notice on external provisoning update. After a put, post or delete commando system is executed towards the API, the API notifies the provisioning sub-system which will start as soon as possible. There is no additional interaction needed to preforme external prosivioning.

7.1 POST

The POST method is used for creating/Adding a new resource to the system. It is applied on the base URI of the Method. When Posting or creating/adding a new resource the values are added into the body of the request in JSON format. Parameters are documented under each service path

When creating a new instance of any object you start with a POST - which then returns the UUID of the object in the **Location header.**

Most create function supports both loading create data, and attribute data in the initial post, which can in situation avoid and additional operation, however all used attributes needs to be known on the client side from before.

```
{
   "xxx":"yyy",
   "zzz":"aaa",Location header.
   "AttributeValues": [
       {“name”: “<keyname>”, “value”:”<data value>”},
       {“name”: “<keyname>”, “value”:”<data value>”}]
}
```

7.2 PUT

The PUT method is used for updating and setting parameters to a resource. To clarify the usage of the update routine of the API, it is working with changes in attributes only and not re-post of complete object. In different with many REST APIs you only post the attributes you want to change, and not a new object. The expected Content-type is **application/json** in a key/value format as following:

```
[
    {“name”: “<keyname>”, “value”:”<data value>”},
    {“name”: “<keyname>”, “value”:”<data value>”}
]
```

7.3 GET

The GET method is used for fetching data from the system, additional parameters are passed in as url parameters in addition to the URI path.

Response list is described under each respective API function, and consist of JSON formatted data.

7.4 DELETE

The DELETE method is used for removing the refereed object by the passed in UUID.

8 Accessing the API

8.1 Namespace and version handling

To access the REST-like API using HTTP an URL under the namespace /api/ on the API-Server is holding the API. Followed by version number.

/api/v1/

The version number is referring to which version number of the API the client are going to work against, when introducing changes that breaks with current API, a updated version number will be introduced with the new features, allowing running system to maintenance compatibility with the API.

8.1.1 Root

- /api/v1 Ok (Expected: 2.3)

8.1.2 Entity

- /api/v1/entity Ok (Expected: 2.3)
- /api/v1/entity/availableAttributes Ok (Expected: 3.0)
- /api/v1/entity/search Ok (Expected: 2.3)
- /api/v1/entity/<UUID> Ok (Expected: 2.3)
- /api/v1/entity/<UUID>/password Ok (Expected: 3.0)
- /api/v1/entity/<UUID>/availableServices Ok (Expected: 2.3)
- /api/v1/entity/<UUID>/service Ok (Expected: 2.3)
- /api/v1/entity/<UUID>/service/<UUID> Ok (Expected: 2.3)
- /api/v1/entity/<UUID>/service/<UUID>/availableSubServices Ok (Expected: 3.0)
- /api/v1/entity/<UUID>/service/<UUID>/subService Ok (Expected: 3.0)
- /api/v1/entity/<UUID>/service/<UUID>/subService/<UUID> Ok (Expected: 3.0)
- /api/v1/entity/<UUID>/resetPassword Ok (Expected: 3.0)

8.1.3 Service

- /api/v1/service/availableAttributes Ok (Expected: 3.0)
- /api/v1/service/serviceDetails Ok (Expected: 3.1)

8.1.4 Customer

- /api/v1/customer/search Ok (Expected: 2.3)
- /api/v1/customer/<UUID>/groups Ok (Expected: 2.3)
- /api/v1/customer/<UUID>/entities Ok (Expected: 2.3)

8.1.5 Structure

- /api/v1/structure Ok (Expected: 3.0)
- /api/v1/structure/availableAttributes Ok (Expected: 3.0)
- /api/v1/structure/<UUID> Ok (Expected: 3.0)
- /api/v1/structure/<UUID>/hide Ok (Expected: 3.3)
- /api/v1/structure/<UUID>/unhide Ok (Expected: 3.3)
- /api/v1/structure/treeRoot Ok (Expected: 3.0)
- /api/v1/structure/treeChildren Ok (Expected: 3.0)

8.1.6 Meeting IQ Parameters

- /api/v1/meetingiqparameter Ok (Expected: 3.0)
- /api/v1/meetingiqparameter/availableTypes Ok (Expected: 3.0)
- /api/v1/meetingiqparameter/availableAttributes Ok (Expected: 3.0)
- /api/v1/meetingiqparameter/<UUID> Ok (Expected: 3.0)

8.2 How to communicate with the API

We do expect the engineer that are going to integrate with the API, have some basic knowledge about talking to API on a general basic - to test out there are various good tools that can be giving result out of the box. PostMan[4] which is an extention to Google Chrome browser is a crossplatform solution that is free, which can be a handy starting point.

Below is a short Python example on how to query the API. For even simpler code examples the requets[5] library on python allows requests in few lines

```
import httplib2 as http
import json

try:
    from urlparse import urlparse
except ImportError:
    from urllib.parse import urlparse

headers = {
    'Accept': 'application/json',
    'Content-Type': 'application/json; charset=UTF-8'
}

uri = 'http://api.synergysky.com'
path = '/api/v1/entity/search?searchString=Doe&pageNumber=0&pageSize=20'

target = urlparse(uri+path)
method = 'GET'
body = ' '

h = http.Http()

# If you need authentication some example:
if auth:
    h.add_credentials(auth.user, auth.password)

response, content = h.request(
        target.geturl(),
        method,
        body,
        headers)

# assume that content is a json reply
# parse content with the json module
data = json.loads(content)
```

9 Enumerations

9.1 DataType

Integer	Value
1	String
2	Integer
3	Float
4	Boolean
5	DateTime
6	Password
7	URL
8	Dropdown
9	Time
10	E-Mail
12	SSHA1Password

9.2 Services and Sub-Services

Id	Service name	Type
1	SKY Admin Portal Login	Service
2	SKY Provisioning Portal Login	Service
4	Jabber Video	Service
5	Acano CoSpace	Service
501	Acano Access Method	Sub-Service
6	Acano User	Service
7	Pexip VMR	Service
701	Pexip Conference Alias	Sub-Service
8	Cisco Codian Conference	Service
9	Microsoft Lync	Service
10	Cisco VCS Registration	Service
1001	Cisco VCS Registration Alias	Sub-Service
11	Seevia	Service

9.3 Entity types

Id	Type name
1	Person
2	Meeting Room

9.4 Structure Types

Id	Type name
1	Service Provider
2	Customer
3	Group

9.5 Meeting IQ Parameter types

Id	Type name
1	Acano Call Leg Profile
2	Acano Call Branding Profile
3	Acano Client Call Branding Profile
4	Acano Ivr Branding Profile
5	Acano User Profile
6	Acano Call Profile

10 General Data models

10.1 AttributeValues Array

Most value parameters that are passed into a service or object comes in a form of a key/value list, which is using the AttributeValues object in an JSON array. In any documentation where its stated AttributeValues as type of data, it implies following data structure.

It occures both in response aswell as in requests

Attribute Name	Parameter type	Description
Name	string	The name of the attribute to be set or which is expressed
Value	string	The value of the attribute to be set or which is expressed

10.1.1 Example

```
{
        "XX": "YY",
        "ZZ": -1,
        "AttributeValues": [
                {"Name": "firstname","Value": "John"},
                 {"Name": "lastname","Value": "Smith"}]
}
```

10.2 Entity: Get all available attributes

Service path: **/api/v1/entity/availableAttributes**
This method is used for retrieving a list of available entity attributes.
HTTP Method accepted: [GET]

10.2.1 GET

10.2.1.1 Request

URL Parameter name	Parameter type	Description
entityTypeId	int	Entity type id, see enumeration table

10.3 Services: Get all available attributes

Service path: **/api/v1/service/availableAttributes**
This method is used for retrieving a list of available service attributes.
HTTP Method accepted: [GET]

10.3.1 GET

10.3.1.1 Request

URL Parameter name	Parameter type	Description
ServiceTypeId	int	Service type id, see enumeration table

10.4 Structure: Get all available attributes

Service path: **/api/v1/structure/availableAttributes**
This method is used for retrieving a list of available structure attributes.
HTTP Method accepted: [GET]

10.4.1 GET

10.4.1.1 Request

URL Parameter name	Parameter type	Description
structureTypeId	int	Structure type id, see enumeration table

10.5 Meeting IQ Parameter: Get all available attributes

Service path: **/api/v1/meetingiqparameter/availableAttributes**
This method is used for retrieving a list of available meeting iq parameter attributes.
HTTP Method accepted: [GET]

10.5.1 GET

10.5.1.1 Request

URL Parameter name	Parameter type	Description
meetingIqParameterTypeId	int	Meeting IQ Parameter type id, see enumeration table

10.6 General Response

Key Name	Parameter Type	Description	Example
Array			
Name	string	Attribute name	cn
DataType	integer	Attribute type ID see *DataType Enumeration*	1
IsRequired	boolean	If the attribute is required to be set	false
ValidationRegEx	string	Validation regex for the field, can be used by guest interface	[A-Za-z0-9._-]*
ValidationRegExErrorMessage	string	Validation Error message	{{{ 5 }}}
LabelText	string	Human readable lable for the attribute	Display Name
HelpText	string	TBD	null
Category	string	Attribute category group, all attributes within same category "belong" together	General
CategorySortOrder	integer	Gives the system suggested category sort order.	1
SortOrder	integer	The system suggestion on attribute sort order, within the category	1
IsReadOnly	boolean	If the attribute is a read only attribute	false
IsMultiValue	boolean	If the attribute can have multiple values. The values are sent as a comma-separated string.	false

10.7 Example


```
[{
    "Name": "FirstName",
    "DataType": 1,
    "IsRequired": true,
    "ValidationRegEx": null,
    "ValidationRegExErrorMessage": null,
    "LabelText": "First Name",
    "HelpText": null,
    "Category": "General",
    "CategorySortOrder": 1,
    "SortOrder": 1,
    "IsReadOnly": false
}, {
    "Name": "LastName",
    "DataType": 1,
    "IsRequired": true,
    "ValidationRegEx": null,
    "ValidationRegExErrorMessage": null,
    "LabelText": "Last Name",
    "HelpText": null,
    "Category": "General",
    "CategorySortOrder": 1,
    "SortOrder": 2,
    "IsReadOnly": false
}, {
    "Name": "mail",
    "DataType": 10,
    "IsRequired": true,
    "ValidationRegEx": null,
    "ValidationRegExErrorMessage": null,
    "LabelText": "E-mail",
    "HelpText": null,
    "Category": "General",
    "CategorySortOrder": 1,
    "SortOrder": 3,
    "IsReadOnly": false
}]
```

11 Get API Version

Service path: **/api/v1**
This is the root of the requested version, a GET on the directory will replay current running sub-version of the API.
HTTP Method accepted: [GET]

11.1 GET

11.1.1 Request

No request options available

11.1.2 Response

Plain text string, printing out current sub-version of the API

11.1.3 Example

Version output
"2.3 build 35"

12 Entity Interaction

12.1 Search for entity

Service path: **/api/v1/entity/search**
Searching the system for entities matching the parameters passed in
HTTP Method accepted: [GET]

12.1.1 GET

12.1.1.1 Request

URL Parameter name	Parameter type	Description
searchString	string	String containing the search from user.
pageNumber	integer	Paging, allowing for sequential get of larger data amount. Default: 0 (Optional)
pageSize	integer	Amount of max returning object in the search. Default: 20 (Optional)
includehiddennodes	Boolean	If return result should return hidden nodes as well, default action is false.

12.1.1.2 Response

Parameter name	Parameter type	Description
Array		
Id	string	The Entity ID in UUID form
Name	string	Entity full name
Email	string	Entity email address
GroupName	string	Written name of group the entity is child of
CustomerName	string	Customer name which the enity belongs to.

12.1.1.3 Example

Request
GET /api/v1/entity/search?searchString=aladdin&pageSize=10


```
[{
    "Id": "07bcaf0a-4f76-4c79-ae21-348a3a6e55d6",
    "EntityTypeId": 1,
    "Name": "Aladdin Bottlemaker",
    "Email": "aladdin.bottlemaker@email.com",
    "GroupName": "Sales",
    "CustomerName": "SynergyTestForce Inc."
}]
```

12.2 Add new entity

Service path: **/api/v1/entity**
This Method is used for adding new entity to the database.
HTTP Method accepted: [POST]

12.2.1 POST

12.2.1.1 Request

Body Parameter name	Parameter type	Description
ownerNodeId	string	The ID in UUID form of the Group node which the entity will be added under.
EntityTypeId	integer	Type of entity. See Enumeration table for Entity Types
password	string	The password for the entity. This is only available if "ExternalPasswordHandling" is enabled in the server config file.
AttributeValues	array	Array of name-value pairs of attribute values, see Paremeter Appendix (Entity: Person)

```
{
  "ownerNodeId": "cb8d7cf1-4240-40d8-8d78-05b038b6f27f",
  "entityTypeId": 1,
  "AttributeValues": [
    {"Name": "firstname","Value": "John"},
    {"Name": "lastname","Value": "Smith"},
    {"Name": "objectname","Value": "John Smith"},
    {"Name":"mail", "Value":"john.smith@mail.ru"}]
}
```

12.3 Set password

Service path: **/api/v1/entity/<UUID>/password**
This Method is used for setting the password for the entity. This is only available if "ExternalPasswordHandling" is enabled in the server config file.
HTTP Method accepted: [PUT]

12.3.1 PUT

12.3.1.1 Request

Body Parameter name	Parameter type	Description
password	string	The new password.

12.3.1.2 Example

```
{"password":"hunter2"}
```

12.4 Edit existing entity

Service path: **/api/v1/entity/<UUID>**
This service path is where you manipulate the selected entity on the system
HTTP Method accepted: [GET] [PUT] [DELETE]

Method	Description	Response
PUT	Updates current UUID with parameters submitted in body, the object does NOT requires a submission of the whole object, but accept updates of an array of parameters.	HTTP Return Codes
DELETE	Deletes the current entity and all its child services.	HTTP Return Codes
GET	Get the data structure of the current entity followed by child services.	See Request/Response below

12.4.1 PUT

12.4.1.1 Request

A Put is preformed on path with following format on the body content formatted in Content-Type **application/json**

Parameter	Type	Description
AttributeValues	array	Array of name-value pairs of attribute values

12.4.1.2 Example

```
{
        [{"name": "firstname","value": "John"},
        {"name": "lastname","value": "Smith"}]
}
```

12.4.2 GET

12.4.2.1 Response

Key Name	Parameter Type	Description	Example
Id	string	UUID of the current entity	b3279117-05fd-4745-b19e-ecf4c35c2325
EntityTypeId	int	the entity id number, see enumeration table	1
AttributeValues	Array	Array of the attribute values	-
Services	**Array**	Array of the services and values	-

Key Name	Parameter Type	Description	Example
Id	string	Service UUID	3271c54c-e400-4f99-b3df-fb954139d957
EntityId	string	Parrent Entity UUID	110a5ed2-5af2-42c1-ab1c-a1258dc64c9e
AttributeValues	**Array**	Array holding the attributes values for the service	-
ServiceTypeId	string	The service type Id, see enumeration table	7
ServiceTypeName	string	A constant string representation of the service type Id	Pexip VMR

CustomerName	string	Written name of the customer the entity belongs to	SynergyTestForce Inc.

12.4.2.2 Example


```
{
    "Id": "b3279117-05fd-4745-b19e-ecf4c35c2325",
    "EntityTypeId": 1,
    "AttributeValues": [{
        "Name": "firstname",
        "Value": "Egil"
    }, {
        "Name": "lastname",
        "Value": "Hasting"
    }, {
        "Name": "mail",
        "Value": "eh@synergysky.com"
    }],
    "Services": [{
        "Id": "3271c54c-e400-4f99-b3df-fb954139d957",
        "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
        "AttributeValues": [{
            "Name": "CarrierDomainUUID",
            "Value": "4d4a7884-0953-43d3-a3b8-0e6478c77856"
        }, {
            "Name": "CarrierImplementerUUID",
            "Value": "5721d09d-374a-4703-8aac-e78ae751dedd"
        },{/* snipped */} ],
        "ServiceTypeId": 7,
        "ServiceTypeName": "Pexip VMR"
    }, {
        "Id": "bac9c005-732d-44ff-9cd0-8fea0e199343",
        "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
        "AttributeValues": [{
            "Name": "FirstName",
            "Value": "Egil"
        }, {
            "Name": "LastName",
            "Value": "Hasting"
        }],
        "ServiceTypeId": 2,
        "ServiceTypeName": "Customer Service Login"
    }, {
        "Id": "69f2a9d5-324b-4e3c-a99a-23dacc6b4fc7",
        "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
        "AttributeValues": [{
            "Name": "CarrierDomainUUID",
            "Value": "a5bb71b4-2fb2-47e5-b2ac-4f88bf07555d"
        }, {
            "Name": "CarrierImplementerUUID",
            "Value": "b4095f8f-f317-44b2-bd98-3f6bda60b125"
        }, {
            "Name": "URI",
            "Value": "sip://EH@synergysky.com"
        }, {/* snipped */} ],
        "ServiceTypeId": 6,
        "ServiceTypeName": "Acano User"
    }],
    "CustomerName": "SynergyTestForce Inc."
}
```

12.5 Get available services to the Entity

Service path: **/api/v1/entity/<UUID>/availableServices**
This method will allow you to get all service available for creation given entity in UUID, the list is limited to assigned service billing template to the parent group of the entity. Which enforce limitation to end-user in what can be deployed.
HTTP Method accepted: [GET]

12.5.1 GET

12.5.1.1 Response

Parameter name	Parameter type	Description
Id	integer	Service Type ID, used to identify which service you want to add
Name	string	Service name, human readable

12.5.1.2 Example

```
[{
  "Id": 5,
  "Name": "Acano coSpace"
}]
```

12.6 Create new service/List services

Service path: **/api/v1/entity/<UUID>/service**
This Method is used for adding new service attached to the entity UUID into the database. Parameters are added into the body. The service UUID can be found found in the Location header of the response
HTTP Method accepted: [GET] [POST]

12.6.1 POST

12.6.1.1 Request

Parameter name	Parameter type	Description
serviceTypeId	Int	Type of service to be deployed see enumeration table
AttributeValues	Array	An array of key-value pairs

12.6.1.2 Example

```
{
        "serviceTypeId":5,
        "AttributeValues":
                [{"Name":"<name>","Value":"<value>"},
                {"Name":"<name>","Value":"<value>"}]
}
```

12.6.2 GET

12.6.2.1 Response

Key Name	Parameter Type	Description	Example
Services	**Array**	Array of the services and values	-

Key Name	Parameter Type	Description	Example
Id	string	Service UUID	5023c3ed-259a-4b9b-986b-201781229240
TemplateServiceId	string	▪ TemplateServiceId Not Ready (Expected: 3.0)	-
EntityId	string	Parrent Entity UUID	110a5ed2-5af2-42c1-ab1c-a1258dc64c9e
AttributeValues	**Array**	Array holding the attributes values for the service	-

Key Name	Parameter Type	Description	Example
Name	string	The "key" used to related the actual value connected to the Attribute type with the same name	cn
Value	string	The actual value of the attribute	null

ServiceTypeId	string	The service type Id	6
ServiceTypeName	string	A constant string representation of the service type Id	Acano User

12.6.2.2 Example


```
[{
        "Id": "4ebccce1-3381-4d6d-8d9d-5c2f65599627",
        "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
        "AttributeValues": [{
            "Id": "cce5dc21-c00e-4bd3-bad2-0e454bef70d3",
            "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
            "AttributeValues": [{
                "Name": "CarrierDomainUUID",
                "Value": "a5bb71b4-2fb2-47e5-b2ac-4f88bf07555d"
            }, {
                "Name": "CarrierImplementerUUID",
                "Value": "b4095f8f-f317-44b2-bd98-3f6bda60b125"
            }, {
                "Name": "URI",
                "Value": "sip://co.gh@synergysky.com"
            }, {
                "Name": "ObjectName",
                "Value": "CospaceTest"
            }, {
                "Name": "CarriedPeerURIUserpart",
                "Value": "co.eh"
            }, {
                "Name": "CarriedPeerMultiConcurrentCapacity",
                "Value": "10"
            }, {
                "Name": "MeetingIQParametersUUID",
                "Value": "db3c3880-b986-11e3-a5e2-0800200c9a66"
            }],
            "ServiceTypeId": 5,
            "ServiceTypeName": "Acano coSpace"
        }, {
            "Id": "69f2a9d5-324b-4e3c-a99a-23dacc6b4fc7",
            "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
            "AttributeValues": [{
                "Name": "CarrierDomainUUID",
                "Value": "a5bb71b4-2fb2-47e5-b2ac-4f88bf07555d"
            }, {
                "Name": "CarrierImplementerUUID",
                "Value": "b4095f8f-f317-44b2-bd98-3f6bda60b125"
            }, {
                "Name": "URI",
                "Value": "sip://EH@synergysky.com"
            }, {
                "Name": "ObjectName",
                "Value": "Egil Hasting"
            }, {
                "Name": "FirstName",
                "Value": "Egil"
            }, {
                "Name": "LastName",
                "Value": "Hasting"
            }, {
                "Name": "AcanoXmppUserName",
                "Value": "EH"
            }],
            "ServiceTypeId": 6,
            "ServiceTypeName": "Acano User"
        }]
```


12.7 Request information about a specific service

Service path: **/api/v1/service/serviceDetails**
This Method is used for requesting information about a specific service attached to the entity UUID into the database. Parameters are added into the body
HTTP Method accepted: [GET]

12.7.1 GET

Request URL: /api/v1/service/serviceDetails?entityServiceId=b79a4912-a755-43a3-b6bc-815ad94538d2

12.7.1.1 Response

Key Name	Parameter Type	Description	Example
Services	**Array**	Array of the services and values	-

Key Name	Parameter Type	Description	Example
Id	string	Service UUID	5023c3ed-259a-4b9b-986b-201781229240
EntityId	string	Parrent Entity UUID	110a5ed2-5af2-42c1-ab1c-a1258dc64c9e
AttributeValues	**Array**	Array holding the attributes values for the service	-

Key Name	Parameter Type	Description	Example
Name	string	The "key" used to related the actual value connected to the Attribute type with the same name	cn
Value	string	The actual value of the attribute	null

Key Name	Parameter Type	Description	Example
ServiceTypeId	string	The service type Id	6
ServiceTypeName	string	A constant string representation of the service type Id	Acano User
ProvisioningStatus	string	A constant string representation of the service type Id	Acano User

Key Name	Parameter Type	Description	Example
Status	string	The current provisioning status for this service	pending
StatusMessage	string	Any message related to the status	Provisioning failed due to...
LastUpdated	datetime	The timestamp when the provisioning status last was updated (UTC time zone)	2014-06-23T12:05:27.487

12.7.1.2 Example


```
[{
        "Id":"b79a4912-a755-43a3-b6bc-815ad94538d2",
        "EntityId":"0d1067bc-4c77-49ec-98b4-77eec9317c4b",
        "AttributeValues":
          [
             {"Name":"FirstName","Value":"Eivind"},
             {"Name":"LastName","Value":"Larsen"},
             {"Name":"HasAdvancedModeAccess","Value":"FALSE"}
          ],
        "ServiceTypeId":2,
        "ServiceTypeName":"SKY Provisioning Portal Login",
        "ProvisioningStatus":
          {
             "Status":"pending",
             "StatusMessage":"Pending",
             "LastUpdated":"2014-06-23T12:05:27.487"
          }
}]
```

13 Service Interaction

13.1 Edit existing service

Service path: **/api/v1/entity/<UUID>/service/<UUID>**
This Method allows for manipulation of the service specified by the UUID, under an entity spesified by the UUID.
HTTP Method accepted: [GET] [PUT] [DELETE]

13.1.1 PUT

13.1.1.1 Request

Parameter name	Parameter type	Description
AttributesValues	array	see enumeration table

13.1.1.1.1 Attribute Object

The attributes for the differnt services are injected into the api using POST or PUT with the data encapsulated into a AttributeValues array The attributes available for the different services are to be found in the Appendix.

13.1.1.2 Example

```
{
        [{"name": "firstname","value": "John"},
        {"name": "lastname","value": "Smith"}]
}
```

13.1.2 GET

13.1.2.1 Response

Key Name	Parameter Type	Description	Example
Id	string	Service UUID	5023c3ed-259a-4b9b-986b-201781229240
EntityId	string	Parrent Entity UUID	110a5ed2-5af2-42c1-ab1c-a1258dc64c9e
AttributeValues	**Array**	Array holding the attributes values for the service	-

13.1.2.2 Example


```
{
    "Id": "3271c54c-e400-4f99-b3df-fb954139d957",
    "EntityId": "110a5ed2-5af2-42c1-ab1c-a1258dc64c9e",
    "AttributeValues": [{
        "Name": "CarrierDomainUUID",
        "Value": "4d4a7884-0953-43d3-a3b8-0e6478c77856"
    }, {
        "Name": "CarrierImplementerUUID",
        "Value": "5721d09d-374a-4703-8aac-e78ae751dedd"
    }, {
        "Name": "URI",
        "Value": "miq://3271c54c-e400-4f99-b3df-fb954139d957"
    }, {
        "Name": "CarriedPeerMultiConcurrentCapacity",
        "Value": "10"
    }, {
        "Name": "PexipVMRDescription",
        "Value": "Egils VMR 1"
    }, {
        "Name": "PexipVMRAllowGuest",
        "Value": "FALSE"
    }],
    "ServiceTypeId": 7,
    "ServiceTypeName": "Pexip VMR"
}
```

13.2 Sub-Services: Get all available sub-service types

Service path: **/api/v1/entity/<UUID>/service/<UUID>/availableSubServices**
This method is used for retrieving a list of available sub-services service id.
HTTP Method accepted: [GET]

13.2.1 GET

13.2.1.1 Response

Parameter name	Parameter type	Description
Id	integer	Service Type ID, Type of service to be deployed, see paragraph 9.2
Name	string	Service name, human readable

13.2.1.2 Example


```
[{
  "Id": 501,
  "Name": "Acano accessMethode"
}]
```


13.3 Sub-Services: List and Create Sub-Services

Service path: **/api/v1/entity/<UUID>/service/<UUID>/subService**
This methode is used to create or list Sub-Services under the given service UUID
HTTP Method accepted: [GET] [POST]

13.3.1 POST

13.3.1.1 Request

Parameter name	Parameter type	Description
serviceTypeId	Int	Type of sub-service to be deployed, see paragraph 9.2
AttributesValues	array	see enumeration table

13.3.2 GET

13.3.2.1 Response

Parameter name	Parameter type	Description
Id	string	Sub-Service reference UUID
EntityId	string	Entity UUID this sub-service belongs to
ServiceTypeId	int	Type of sub-service to be deployed, see paragraph 9.2
ServiceTypeName	string	Human readable name for the service
AttributesValues	array	see enumeration table

13.3.2.2 Example


```
[{
    "Id": "a0f09c93-fda0-4345-9183-fa9245f22d4d",
    "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
    "AttributeValues": [{
        "Name": "CarrierDomainUUID",
        "Value": "4d4a7884-0953-43d3-a3b8-0e6478c77856"
    }, {
        "Name": "URI",
        "Value": "sip://meet.test@synergysky.com"
    }, {
        "Name": "CarriedPeerURIUserpart",
        "Value": "meet.test"
    }, {
        "Name": "PexipAliasDescription",
        "Value": "Test VMR"
    }],
    "ServiceTypeId": 701,
    "ServiceTypeName": "Pexip Conference Alias"
},
{ /* .... */},
{ /* .... */}]
```


13.4 Sub-Services: Interact with current subService

Service path: **/api/v1/entity/<UUID>/service/<UUID>/subService/<UUID>**
This method is used for interacting with the spesific sub-service.
HTTP Method accepted: [GET] [PUT] [DELETE]

13.4.1 POST

13.4.1.1 Request

Parameter name	Parameter type	Description
AttributesValues	array	see enumeration table

13.4.1.1.1 Attribute Object

The attributes for the differnt services are injected into the api using POST or PUT with the data encapsulated into a AttributeValues array The attributes available for the different services are to be found in the Appendix.

13.4.1.2 Example


```
{
        [{"name": "firstname","value": "John"},
        {"name": "lastname","value": "Smith"}]
}
```

13.4.2 GET

13.4.2.1 Response

Parameter name	Parameter type	Description
Id	string	Sub-Service reference UUID
EntityId	string	Entity UUID this sub-service belongs to
ServiceTypeId	int	Type of sub-service to be deployed, see paragraph 9.2
ServiceTypeName	string	Human readable name for the service
AttributesValues	array	see enumeration table

13.4.2.2 Example


```
[{
    "Id": "a0f09c93-fda0-4345-9183-fa9245f22d4d",
    "EntityId": "b3279117-05fd-4745-b19e-ecf4c35c2325",
    "AttributeValues": [{
        "Name": "CarrierDomainUUID",
        "Value": "4d4a7884-0953-43d3-a3b8-0e6478c77856"
    }, {
        "Name": "URI",
        "Value": "sip://meet.someone1@synergysky.com"
    }, {
        "Name": "CarriedPeerURIUserpart",
        "Value": "meet.someone1"
    }, {
        "Name": "PexipAliasDescription",
        "Value": "Someones VMR"
    }],
    "ServiceTypeId": 701,
    "ServiceTypeName": "Pexip Conference Alias"
}]
```

13.5 Search for customers in system

Service path: **/api/v1/customer/search**
This method search for the customers in the tree based upon the given string
HTTP Method accepted: [GET]

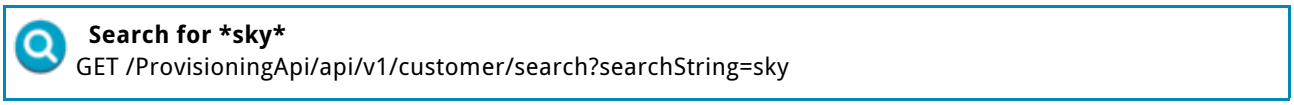
13.5.1 Request

URL Parameter name	Parameter type	Description
searchString	String	The ID of the Group node which the entity will be added under.
pageNumber	Int	Paging, allowing for sequential get of larger data amount.
pageSize	Int	Amount of max returning object in the search
includehiddennodes	Boolean	If return result should return hidden nodes as well, default action is false.

13.5.2 Response

Parameter name	Parameter type	Description
Array		
Id	string	UUID of the customer object
Name	string	Printable name of the customer found

13.5.3 Example




```
[
    {
        "Id": "acbff81e-8fe3-4eb8-a969-f55ba2eef69c",
        "Name": "Synergy Sky new"
    }
]
```

14 Structure Interaction

14.1 Customer: List all groups belonging to a Customer

Service path: **/api/v1/customer/<UUID>/groups**
This method lists all the groups under the customer, in the given UUID.
HTTP Method accepted: [GET]

14.1.1 GET

14.1.1.1 Response

Parameter name	Parameter type	Description
Array		
Id	string	UUID of the group object
Name	string	Printable name of the group
includehiddennodes	Boolean	If return result should return hidden nodes as well, default action is false.

14.1.1.2 Example

Search for *sky*
GET /ProvisioningApi/api/v1/customer/acbff81e-8fe3-4eb8-a969-f55ba2eef69c/groups


```
[
    {
        "Id": "a3abf635-6c44-49ab-9782-84fd62ce4b1f",
        "Name": "Employees"
    }
]
```


14.2 Customer: List all entities belonging to a customer

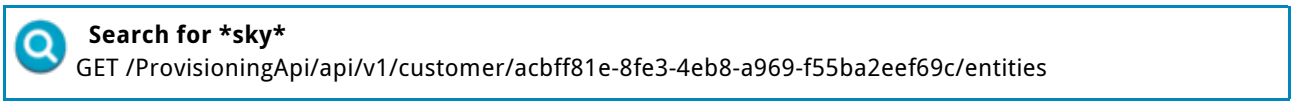
Service path: **/api/v1/customer/<UUID>/entities**
This method lists all the entities under the customer, in the given UUID.
HTTP Method accepted: [GET]

14.2.1 GET

14.2.1.1 Response

Parameter name	Parameter type	Description
Array		
Id	string	UUID of the entity object
EntityTypeId	int	The type id of the object, see chapter 9.3 for enumerations
Name	string	Entity full name
Email	string	Entity email
GroupName	string	Name of the group it belongs to
CustomerName	string	Name of the customer it belongs to

14.2.1.2 Example




```
[
    {
            "Id": "e4e6a5b1-93a7-4804-adac-88d56d3ed7ea",
            "EntityTypeId": 1,
            "Name": "Otto Obama",
            "Email": "otto.obama@ottosspik.com",
            "GroupName": "Otto's Spik",
            "CustomerName": "SynergyTestForce Inc."
    },
    {
            "Id": "4aefd408-3b99-4c09-a95b-2b85e40723cc",
            "EntityTypeId": 1,
            "Name": "Otto Firebird",
            "Email": "otto.firebird@ottosspik.com",
            "GroupName": "Otto's Spik",
            "CustomerName": "SynergyTestForce Inc."
    }
]
```

14.3 Structure: Add new structure object to tree

Service path: **/api/v1/structure**
This methode is used for creating structure objects (i.e. customers, service providers, groups)
HTTP Method accepted: [POST]

This method allows for manipulate the tree structure where each of the enity and then services are applied to, and in a larger deployment where there are separation between service providers.

The root of the tree in Synergy Sky 3.0 is an UUID fixed to **9000000d-c0de-ba5e-9090-909090909090** It is as a general rule not allowed to create a new root object, all objects need to be either related to the root as its parent or a sub level from the root.

14.3.1 POST

14.3.1.1 Request

Parameter name	Parameter type	Description
ParentNodeID	string	UUID of the parent object which is where new object should be placed under.
StructureTypeId	int	see enumeration table
AttributeValues	array	-

Note
Following limitations applies when adding nodes to parent nodes. 1) Service Provide node can only be placed under another Service Provider node. 2) Group node has to be placed under a Customer or Group node

14.3.1.2 Example

```
{
        "ParentNodeID": "9000000d-c0de-ba5e-9090-909090909090",
        "StructureTypeId":1,
        "AttributeValues":
                [{"Name":"objectname","Value":"Teleco Inc."}]
}
```

14.4 Structure: Modify a structure container

Service path: **/api/v1/structure/<UUID>**
This methode is used for modifying structure objects (i.e. customers, service providers, groups)
HTTP Method accepted: [GET] [PUT] [DELETE]

> **TODO**
> Remove the carrierdomain search, this is not longer valid. its now implied through the infrastructure component - also check the service/sub-service part of the api guide if anything needs to be modified there.

14.4.1 PUT

14.4.1.1 Request

Parameter name	Parameter type	Description
AttributeValues	array	-

14.4.1.2 Example

```
{
        [{"Name":"objectname","Value":"Teleco Inc."}]
}
```

14.5 Data source list: Get related data source list

This method allows the user to look up list-items based upon service type and attribute name. This is for example a list of profiles available for a spesific service/sub-service type. Or a list of implementors/Devices which the to-be provisoned object shall be placed on.

All objects that works by having internal references can be looked up using this methode. So if there is an intention to create a pulldown list over possible profile assigable to a coSpace or pexip vmr, this would be the method to use.

Service path: **/api/v1/service/dataSource**
This methode is used to create a new subService under the given service
HTTP Method accepted: [GET]

14.5.1 GET

14.5.2 Request

URL Parameter name	Parameter type	Description
serviceTypeId	int	Service Type ID, see enumeration
name	string	Attribute name to source looking up

Supported Attributes

Name	Description
carrierimplementeruuid	Returns back all available carriers (Devices) compatible with requested ServiceTypeId
carrierdomainuuid	Returns back all available domains (xmpp/sip/h323 suffix) compatible with requested ServiceTypeId
meetingiqparametersuuid	Returns back all available Parameter collections compatible with requested ServiceTypeId

14.5.2.1 Response

Key Name	Parameter Type	Description	Example
AttributeValues	**Array**	Array holding the attributes values for the service	-

14.5.2.2 Example

Example would be to list all "Acano profiles", which would be possible by using following parameters in the query.

- serviceTypeID = 5
- name = MeetingIQParametersUUID
- Constructed URL: /provisioningapi/api/v1/service/dataSource?serviceTypeId=5&name=MeetingIQParametersUUID


```
[{
    "Name": "Moderator",
    "Value": "db3c3880-b986-11e3-a5e2-0800200c9a66"
}, {
    "Name": "Guests",
    "Value": "afd93ca0-b987-11e3-a5e2-0800200c9a66"
}]
```


The values can then be used to set which profile the coSpace/AccessMethod should be member of.

```
- The value is then assigned back to the service objects attribute "MeetingIQParametersUUID"
```

- serviceTypeID = 5
- name = CarrierDomainUUID
- Constructed URL: /provisioningapi/api/v1/service/dataSource?serviceTypeId=5&name=CarrierDomainUUID


```
[{
    "Name": "Synergy Acano Domain",
    "Value": "a5bb71b4-2fb2-47e5-b2ac-4f88bf07555d"
}, {
    "Name": "Synergy Acano Domain test",
    "Value": "44d9e8de-f520-449b-915b-aa7203a68d3b"
}]
```


This lists out the carrier domain of a specific type which needs to be used when deploying a service (carried peer) on an entity.

15 Meeting IQ Parameter Interaction

15.1 Get available Meeting IQ Parameter types

Service path: **/api/v1/meetingiqparameter/availableTypes**
This method lists all meeting iq parameter types which can be used when creating a new meeting iq parameter.
HTTP Method accepted: [GET]

15.1.1 GET

15.1.1.1 Response

Parameter name	Parameter type	Description
Array		
Id	int	Type id of the meeting iq parameter type
Name	string	Name of the type

15.1.1.2 Example

[{

```
  "Id": 1,
  "Name": "Acano Call Leg Profile"
```

}]

15.2 Add new Meeting IQ Parameter

Service path: **/api/v1/meetingiqparameter**
This method is used for creating meeting iq parameters and getting all meeting iq parameters for a structure node
HTTP Method accepted: [GET] [POST]

15.2.1 POST

15.2.1.1 Request

Parameter name	Parameter type	Description
OwnerNodeID	string	UUID of the parent object which is where new object should be placed under.
MeetingIqParameterTypeId	int	see enumeration table
AttributeValues	array	-

15.2.2 GET

15.2.2.1 Request

Parameter name	Parameter type	Description
NodeId	string	UUID of the structure node.

15.3 Edit existing meeting iq parameter

Service path: **/api/v1/meetingiqparameter/<UUID>**
This service path is where you manipulate the selected meeting iq parameter
HTTP Method accepted: [GET] [PUT] [DELETE]

Method	Description	Response
PUT	Updates current UUID with parameters submitted in body, the object does NOT requires a submission of the whole object, but accept updates of an array of parameters.	HTTP Return Codes
DELETE	Deletes the current meeting iq parameter.	HTTP Return Codes
GET	Get the data structure of the current meeting iq parameter.	See Request/Response below

15.3.1 PUT

15.3.1.1 Request

A Put is preformed on path with following format on the body content formatted in Content-Type **application/json**

Parameter	Type	Description
AttributeValues	array	Array of name-value pairs of attribute values

15.3.1.2 Example

```
{
        [{"name": "acanocalllegprofiledefaultlayout","value": "dualStream"},
        {"name": "acanocalllegprofilesipmediaencryption","value": "required"}]
}
```

15.3.2 GET

15.3.2.1 Response

Key Name	Parameter Type	Description	Example
Id	string	UUID of the current entity	b3279117-05fd-4745-b19e-ecf4c35c2325
TypeId	int	the meeting iq parameter type id, see enumeration table	1
TypeDisplayName	string	The meeting iq parameter type display name, see enumeration table	Acano Call Leg Profile
AttributeValues	Array	Array of the attribute values	-

15.3.2.2 Example


```
{
        "Id": "2a77fbe4-1cab-4fbb-9fea-8f7f76c0486a",
        "TypeId": 1,
        "TypeDisplayName": "Acano Call Leg Profile",
        "AttributeValues": [{
                "Name": "ObjectName",
                "Value": "test3"
        },
        {
                "Name": "AcanoCallLegProfilePresentationDisplayMode",
                "Value": "singleStream"
        },
        {
                "Name": "AcanoCallLegProfileRxAudioMute",
                "Value": "TRUE"
        },
        {
                "Name": "AcanoCallLegProfileTxAudioMute",
                "Value": "FALSE"
        }]
}
```

16 TODO!

The api for showing a tree structure, used by UI elements in our provisioning API, is not yet documented - but will be at a later stage, however they do need some hardening before being available for public.

- /api/v1/structure/treeRoot Ok (Expected: 3.x)

-	-	-
includehiddennodes	Boolean	If return result should return hidden nodes as well, default action is false.

- /api/v1/structure/treeChildren Ok (Expected: 3.x)

-	-	-
includehiddennodes	Boolean	If return result should return hidden nodes as well, default action is false.

17 Remarks

1. ↑ http://en.wikipedia.org/wiki/REST
2. ↑ http://en.wikipedia.org/wiki/Basic_access_authentication]
3. ↑ http://en.wikipedia.org/wiki/Basic_access_authentication
4. ↑ https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm
5. ↑ http://docs.python-requests.org/en/latest/

Categories: Internal | API