

Application Information anSwitch V7

REST API BEST PRACTICES & EXAMPLES

Classification: Public
Status: Released
Version: E3.0
Author: D. Bochsler

INTRODUCTION & MOTIVATION

- ▶ This training covers the topics:
 - ▶ Description of the anSwitch V7 REST API.
 - ▶ The available REST methods and data formats.
 - ▶ The configurable anSwitch V7 database objects.
 - ▶ Best practices for using the anSwitch V7 REST API.
 - ▶ Examples



*IT'S NOT
MAGIC
IT'S "KNOW
HOW"*

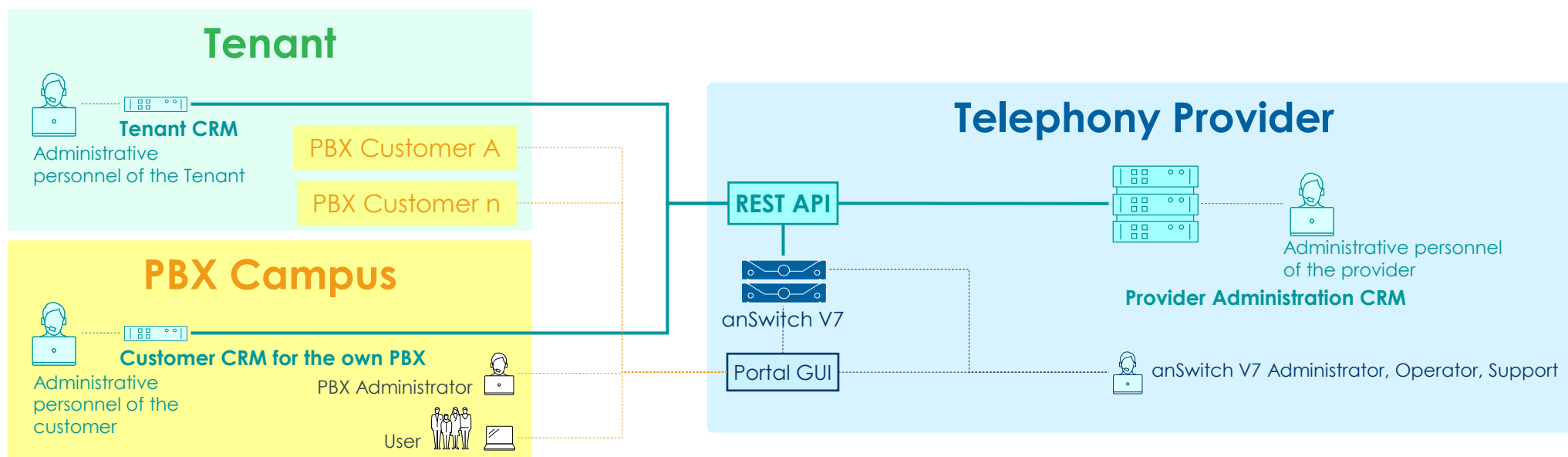
TABLE OF CONTENTS

1	OVERVIEW REST API	9	EXAMPLE: CREATE PBX & PBX EXTENSION
2	SUPPORTED REST METHODS & DATA FORMATS	10	EXAMPLE: ASSIGN & PROVISION PHONES
3	FILTER & SORT & LIMIT OF GET REQUEST RESULTS		▶ Terminal type "anDesktop"
4	MANAGEABLE DATABASE OBJECTS		▶ Terminal type "an IP-Phone"
			▶ Terminal type "anConnect"
5	ERROR HANDLING & TROUBLE SHOOTING		▶ Terminal type "3 rd Party SIP-Phone, Auto-Provisioning"
6	SETTING UP THE REST API INTERACTION		▶ Terminal type "3 rd Party SIP-Phone, URL-Provisioning"
7	BEST PRACTICES	11	EXAMPLE: DELETE A PBX OR PBX EXTENSION
8	"INSOMNIA" A REST CLIENT APPLICATION	12	EXAMPLE: USEFUL PBX & TERMINAL COMMANDS
			▶ Example Code: Force a Phone to re-Download its Config
			▶ Example Code: List all Terminals of a PBX Extension
			▶ Example Code: Set the Time Zone of a Phone
			▶ Example Code: Set the Ringing Tones of a Phone
			▶ Example Code: Get the Voice Mail Messages
			▶ Example Code: Get a Call Recording Ordered via CSTA
			▶ Example Code: Upload Music on Hold for a PBX
			▶ Example Code: Manage Call Forward CF
		13	EXAMPLE: MANAGE CONTACTS
		14	EXAMPLE: LIST CDRS OF A PBX EXTENSION

1 OVERVIEW REST API

OVERVIEW ANSWITCH V7 REST API

- ▶ The REST API allows to configure the anSwitch V7 directly from a customer relationship management CRM.
 - ▶ **REST** stands for: **RE**presentational **S**tate **T**ransfer (For further introductory information, see this [Wikipedia article](#))
- ▶ In a multi-tenant setup of an anSwitch V7 every tenant can have its own CRM application that manages just the own resources.
- ▶ A PBX customer may have its own CRM application with just access to the resources of the own PBX.



OVERVIEW ANSWITCH V7 REST API FEATURES

- ▶ The anSwitch V7 configuration REST API cover these general features:
 - ▶ HTTP/HTTPS as transport protocol.
 - ▶ Security by:
 - ▶ Individual login and authentication for each CRM application.
 - ▶ Access only to the assigned resources.
 - ▶ Secure data transfer by HTTPS.
 - ▶ Various supported data formats are available for the transfer: JSON, XML and CSV.

VIA REST API MANAGEABLE DATABASE OBJECTS

▶ OrgUnit

- ▶ Tenant
- ▶ PBX
- ▶ PBX Extension

▶ User

- ▶ User account
- ▶ User role

▶ Address (phone number)

- ▶ Public numbers
- ▶ Private PBX numbers

▶ Terminal (phones)

- ▶ Phone type:
 - ▶ anDesktop, an IP-Phone, anConnect
 - ▶ 3rd party SIP phone/terminal
- ▶ Location of registration
- ▶ Presence subscriptions

▶ Call Distribution

- ▶ Forward
- ▶ Advanced Call Distribution ACD
- ▶ Interactive Voice Response IVR
- ▶ Timetable
- ▶ Holiday
- ▶ Fax server
- ▶ Pager / Intercom
- ▶ Audio File

▶ Pricelist & Rating & TopStop

- ▶ Pricelist
- ▶ Billing & Billing Limit

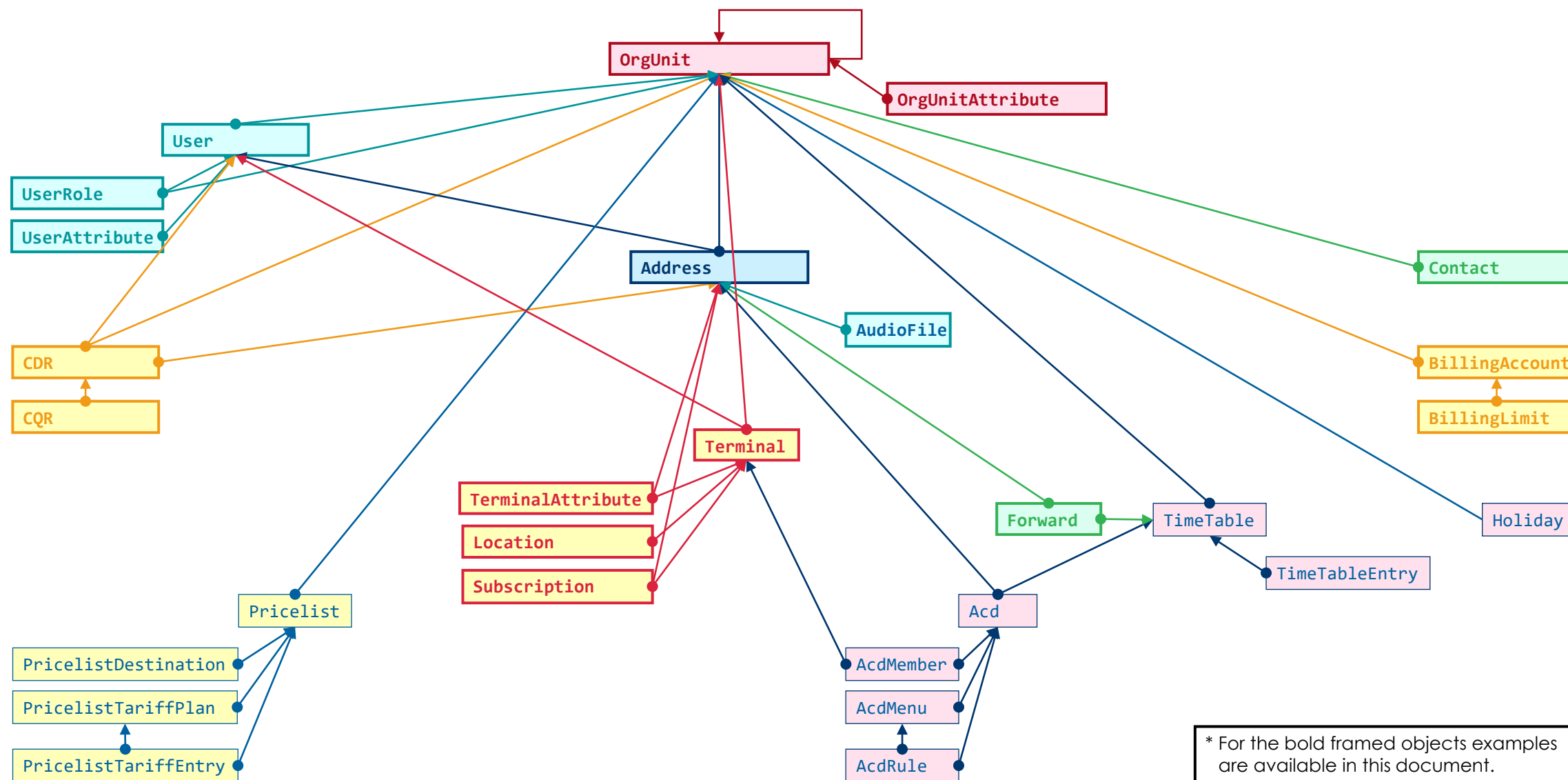
▶ Call Detail Record CDR

▶ Call Quality Record CQR

▶ Contact

- ▶ PBX contacts
- ▶ PBX extension user contacts
- ▶ Short numbers

VIA REST API MANAGEABLE DATABASE OBJECTS



WARNING → DIRECT ACCESS TO THE ANSWITCH V7 DATABASE!

Warning

- An external CRM application that manages its assigned resources via the REST API has full control over its accessible instances in the database of the anSwitch V7!
- Careless manipulations via the REST API can destroy beyond repair!
(Only a DB restore could help eventually with all its other negative side effects.)
 - ▶ Careless manipulations via REST API can destroy the operational functionality of an instance, e.g. PBX
 - ▶ There is no check of the configured values of a property of an instance. There is no built-in "undo".
- Code development for an external application and its testing should therefore definitely take place either on a test anSwitch V7 or before the anSwitch V7 is put into operation.

GREAT POWER → GREAT RESPONSIBILITY!

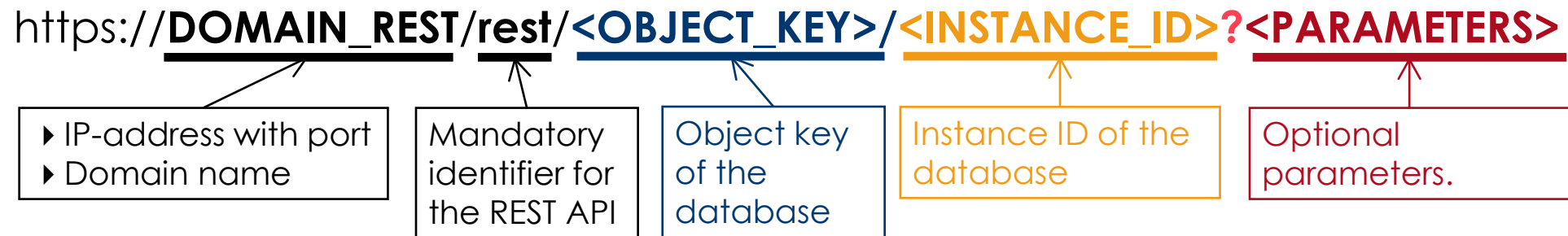
2 SUPPORTED REST METHODS & DATA FORMATS

SUPPORTED REST METHODS

- ▶ The following REST methods are supported:
 - ▶ **POST** → Create a new instance.
 - ▶ **PUT** → Modify an existing instance .
 - ▶ **GET** → Retrieve data of an instance or a list of instances.
 - ▶ **DELETE** → Delete an instance.

URL REQUEST DEFINITION

► URL request definition:



Examples

POST `https://DOMAIN_REST/rest/orgUnits`

PUT `https://DOMAIN_REST/rest/orgUnits/519`

GET `https://DOMAIN_REST/rest/orgUnits/519`

GET `https://DOMAIN_REST/rest/orgUnits?where=name.like('Provider A')&properties=id`

GET `https://DOMAIN_REST/rest/orgUnits/519?properties=name`

SUPPORTED DATA FORMATS

- ▶ The applicable data formats are:

Json → is the **default**

XML

CSV is possible but not recommended

Examples:

```
{
  "name": "Mobile – an IP-Phone",
  "type": "anIpPhone",
  "orgUnitId": 523,
  "userId": 113,
  "password": "fsdfgsdfgf98ug00hzHJB",
  "username": "fvg56AS056"
}
```

```
<terminal>
  <name>Mobile – an IP-Phone</name>
  <type>anIpPhone</type>
  <orgUnitId>523</orgUnitId>
  <userId>113</userId>
  <password>fsdfgsdfgf98ug00hzHJB</password>
  <username>fvg56AS056</username>
</terminal>
```

```
"Mobile – an IP-Phone",
"anIpPhone",523,113,fsdfgsdfgf98ug00hzHJB
S,fvg56AS056
```

JSON formatting rules apply:

- ▶ White spaces and new lines are allowed.
- ▶ String values (type varchar) must be double quoted "...".
- ▶ Properties and their values must be separated by a comma ','.

XML formatting rules apply.

- ▶ The order of the properties must be known/evaluated.
- ▶ String values with white spaces must be double quoted "...".
- ▶ The values must be separated by a comma ','.

- ▶ Example of the same GET request for receiving data in the different formats:

Json: **PUT** https://DOMAIN_REST/rest/terminals/308

XML: **PUT** https://DOMAIN_REST/rest/terminals/308.xml

CSV: **PUT** https://DOMAIN_REST/rest/terminals/308.csv

DEFINITION OF A POST & PUT REQUEST

- ▶ A **POST** request **creates** a new instance of an object.

- ▶ Example of a POST request, JSON formatted

POST https://DOMAIN_REST/rest/orgUnits

```
{
  "name": "PBX REST Test",
  "type": "pbx",
  "description": "PBX of company REST",
  "parentId": 25
}
```

- ▶ Response

200 OK

```
{
  "name": "PBX REST Test",
  "id": 519,
  "type": "pbx",
  "parentId": 25,
  "description": "PBX of company REST"
}
```

A successful POST is confirmed with a **200 OK**.

◀ The ID of the instance is needed for later referencing.

- ▶ A **PUT** request **changes** the value of a property of an instance.

- ▶ Example of a PUT request, JSON formatted

PUT https://DOMAIN_REST/rest/orgUnits/519

```
{
  "name": "PBX 1234-abc-0001",
  "description": "PBX of company A"
}
```

Several the values of several properties of an instance can be changed.

- ▶ Response

200 OK

```
{
  "name": "PBX 1234-abc-0001",
  "id": 519,
  "type": "pbx",
  "parentId": 25,
  "description": "PBX of company A"
}
```

A successful PUT is confirmed with a **200 OK**.

DEFINITION OF A GET REQUEST

▶ A **GET** Request **retrieves** data of an instance or a list of instances

- ▶ Example of a GET that retrieves all instances.

GET https://DOMAIN_REST/rest/orgUnits	
	No body

- ▶ Response

200 OK	
<pre>{ "orgUnits": [{ "name": "PBX REST Test", ... }, { "name": "PBX Company A", ... }] }</pre>	The whole list of all instances is returned. ← Object name

- ▶ Example of a GET that retrieves of a defined instance.

GET https://DOMAIN_REST/rest/orgUnits/ 519	
	No body

- ▶ Response

200 OK	
<pre>{ "orgUnits": [{ "name": "PBX REST Test", "id": 519, "type": "pbx", "parentId": 25, "description": "PBX of company REST" }] }</pre>	All properties of an instances is returned. ← Object name

DEFINITION OF A DELETE REQUEST

- ▶ A **DELETE** request **deletes** the instance.

- ▶ Example of a DELETE that retrieves an instances.

DELETE https://DOMAIN_REST/rest/orgUnits/519
No body

- ▶ Response

200 OK
No body

Warning

There is no undo!

Warning

Prevent dead data in the database!

Make sure to identify other instances that reference to this to delete instance!

- ➔ Consider what to do with any of this eventually orphaned instances:
 - ▶ Delete it too?
 - ▶ Re-reference it to another instance?

3 FILTER & SORT & LIMIT OF GET REQUEST RESULTS

OVERVIEW FILTER & SORT & LIMIT & SPECIFIC PROPERTIES

- ▶ Various parameters and parameter combinations can be used in a GET request to obtain the desired data according to the requirements..
 - ▶ **Filter**
Return only those instances whose property values match the given conditions.
 - ▶ **Sort**
Sort the found instances according the value of the defined instance property ascending or descending.
 - ▶ **Limit**
Return only a defined range of instances.
 - ▶ **Specific Properties**
Return just the defined property and their values of an instance.

OVERVIEW FILTER & SORT & LIMIT & SPECIFIC PROPERTIES

► Build-up of GET request parameters:

GET https://DOMAIN_REST/rest/<OBJECT_KEY>?<FILTER>&<SORT>&<LIMIT>&<PROPERTIES>



► The order of the GET parameters is free

GET https://DOMAIN_REST/rest/<OBJECT_KEY>?<FILTER>&<SORT>&<LIMIT>

GET https://DOMAIN_REST/rest/<OBJECT_KEY>?<LIMIT>&<PROPERTIES>

GET https://DOMAIN_REST/rest/<OBJECT_KEY>?<SORT>&<LIMIT>&<FILTER>&<PROPERTIES>

Examples

GET https://DOMAIN_REST/rest/orgUnits?where=name.like('Provider A')&properties=id

GET https://DOMAIN_REST/rest/orgUnits?where=name.like('Front Desk').and(parentId.eq(26))

GET https://DOMAIN_REST/rest/cdrs?where=accOrgUnitId.eq(28)&limit=5&ascending=id&properties=id,
destNumber,timeConnect,timeEnd,chargePublic

FILTER INSTANCES

▶ Filter the instances according properties.

- ▶ The filter parameter:

`where=<FILTER_CONDITIONS>`

- ▶ The filter condition can be a single property or a combination of properties with conditions.

▶ Examples:

- ▶ Search the tenant with property "name", the string must match exactly "Provider A":

`GET .../orgUnits?where=name.like('Provider A')`

- ▶ Search the instances where:

- * the property "name" begins with "Bo" and has any number following characters.
- * and the property "telNumber" starts with "079" and has any number following digits.

`GET .../addresses?where=name.like('Bo%').and(telNumber.like('079%'))`

- ▶ List all CDR of a PBX between a start and an end date/time:

`GET .../cdrs?accOrgUnitId.eq(28).and(timeStart.ge(1696111200000)).and(timeEnd.le(1698793200000))`

▶ SQL like operators are supported.

▶ Boolean operators

▶ eq()	=	equal
▶ ne()	!=	not equal
▶ and()	&&	logical AND
▶ or()		logical OR

▶ Number operators

▶ eq()	=	equal
▶ ne()	!=	not equal
▶ lt()	<	less than
▶ gt()	>	greater
▶ le()	≤	less or equal
▶ ge()	≥	greater or equal
▶ like()		LIKE
▶ notLike()		NOT LIKE
▶ in(<list>)		IN
▶ notIn(<list>)		NOT IN

▶ String operators

▶ eq()	=	equal
▶ ne()	!=	not equal
▶ like()		LIKE
▶ notLike()		NOT LIKE
▶ in(<list>)		IN
▶ notIn(<list>)		NOT IN

▶ SQL like Regex are supported!

SORT INSTANCES

- ▶ Sort the instances according a property value.

- ▶ The sort parameter:

- ascending=<PROPERTY_NAME>

- descending=<PROPERTY_NAME>

- ▶ Examples:

- ▶ Search the contacts of a PBX Extension where:

- * the property "name" begins with "B".

- * and sort the list ascending according the property "name".

- ```
GET .../ contacts?where=orgUnitId.eq(26).and(name.like('B%'))&ascending=name
```

- ▶ List the CDRs of a PBX where:

- \* The CDRs are between a start and an end date/time.

- \* and sort the list descending according the property "timeStart ".

- ```
GET .../cdrs? where=accOrgUnitId.eq(28).and(timeStart.ge(1696111200000)).and(timeEnd.le(1698793200000))&descending= timeStart
```

LIMIT INSTANCES

- ▶ Limit the listed instances according a range of instances.

- ▶ The limit parameter:

- limit=<RANGE>

- limit=<OFFSET>,<RANGE>

- ▶ Examples:

- ▶ Search the contacts of a PBX Extension where:

- * the property "name" begins with "B".

- * and sort the list ascending according the property "name".

- * and limit the list to the first 5 occurrences.

- GET .../ contacts?where=orgUnitId.eq(26).and(name.like('B%'))&ascending=name&limit=5

- ▶ List the CDRs of a PBX where:

- * the CDRs are older than defined start date/time.

- * and limit the list to 200 CDRs with an offset of 100 (CDR instances between 100 and 300 will be returned).

- GET .../cdrs?where=accOrgUnitId.eq(28).and(timeStart.ge(1696111200000))&limit=100,200

- ▶ Get the first 100 CDRs:

- GET .../cdrs?where=accOrgUnitId.eq(28).and(timeStart.ge(1696111200000))&limit=100

SPECIFIC PROPERTIES

- ▶ Limit the returned properties of an instance according a list of property names.

- ▶ The specific properties parameter:

`properties=<PROPERTY_NAME>`

- ▶ Separate a list of properties with a comma.

- ▶ Examples:

- ▶ List the names and IDs of all PBX's of a tenant:

- * the property "name" begins with "B".

- * and sort the list ascending according the property "name".

- * and limit the list to the first 5 occurrences.

`GET .../orgUnits?where=parentId.eq(4).and(type.like('pbx'))&properties=name,id`

- ▶ List the contacts of a PBX with their name, dialable destination number and short number:

`GET .../contacts?where=orgUnitId.eq(26)&properties=name,telNumberNormalized,shortNumber`

- ▶ List the name and description of a PBX:

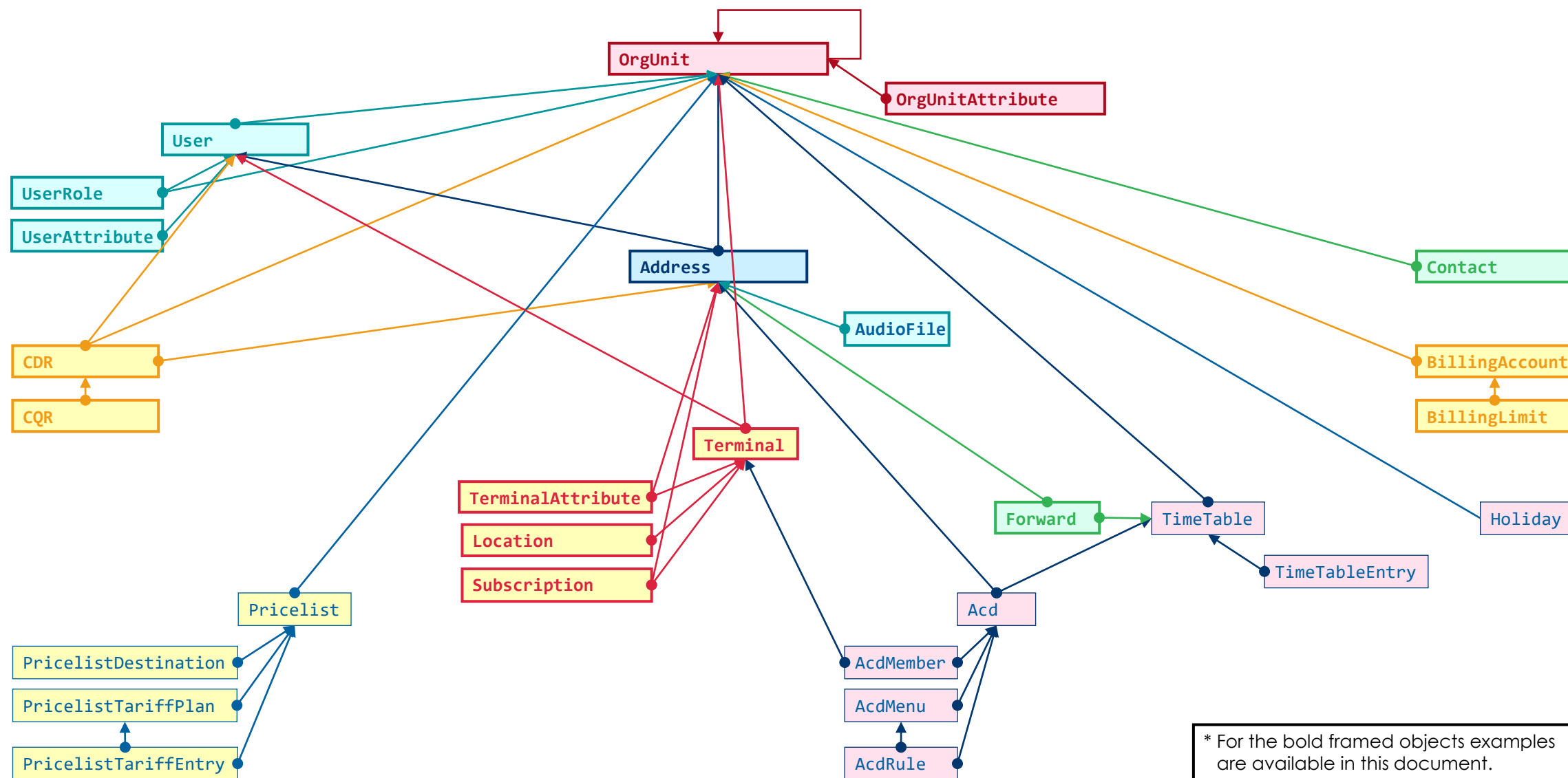
`GET .../orgUnits/519?properties=name,description`

200 OK

```
{
  "orgUnits": [
    {
      "name": "PBX REST Test",
      "description": "PBX of company REST"
    }
  ]
}
```

4 MANAGEABLE DATABASE OBJECTS

VIA REST API MANAGEABLE DATABASE OBJECTS



OVERVIEW INSTANCE PARAMETERS

► Explanation of the upcoming parameter lists

Property name of the database

Database data type:

Primary key : Unique id of the object

INT : Number, 0 - 4'294'967'295

BIGINT : Number, 0 - 184'467'440'737'709'551'615

VARCHAR : String [max. length]

Allowed value:

[1 - n] : Number out of a range until
max. of n (max. of the type)

String : Free string e.g., a name

[a | b | c] : List of allowed strings

Request URL with
<OBJECT_KEY>.

Portal UI menu
path for
configuring an
instance.

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/orgUnits			
Portal UI > Menu: PBX > Sub-menu: Organization Units			
id	Primary key	[1 - n]	Unique OrgUnit-Id, assigned ...
name	VARCHAR [45]	String	
type	VARCHAR [45]	[system tenant pbx extension department]	

HOW TO GET INFORMATION ABOUT MISSING DB OBJECTS AND UNKNOWN PROPERTY VALUES?

- ▶ The documentation about database objects and their valid property values may lag when a new anSwitch V7 version is available.
- ▶ Contact the Aarenet support and place a helpdesk ticket about the missing information.
- ▶ Hint for how to get unknown valid property values!
 - ▶ For getting valid values of e.g., new or extended properties proceed as following:
 1. Via the Portal UI configure the requested parameter in an instance.
 2. Read out the instance via the REST API.
 3. Check in the returned data the value of the corresponding property.

OBJECT-KEY: ADDRESSES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/addresses			
Portal UI > Menu: PBX > Sub-menu: Public Numbers Portal UI > Menu: PBX > Sub-menu: PBX Settings > Tile: Public Numbers Portal UI > Menu: PBX > Sub-menu: Extensions > Extension Setup > Tile: Number			
id	Primary key	[1 - n]	Unique Address-Id, assigned automatically by the DB at creation time
name	VARCHAR [45]	String	Displayed name at the called side
number	VARCHAR [45]	String	Public PSTN or private phone number
orgUnitId	INT	[1 - n]	OrgUnit-Id of the PBX Extension this address belongs to
locationAddrId	INT	[1 - n]	
publicAddrId	INT	[1 - n]	The public PSTN phone number which is the DDI number of the private number
flags	INT	[0 - n]	
scopeOuId	INT	[1 - n]	OrgUnit-Id of the PBX this address belongs
userId	INT	[1 - n]	User-Id of the user account this address is associated with
validFrom	BIGINT	0 - n	Unix time stamp this address is valid from Must be exact to the milli seconds, e.g. 1634737080000
validUntil	BIGINT	0 - n	Unix time stamp this address is valid until Must be exact to the milli seconds, e.g. 1634737080000

OBJECT-KEY: AUDIOFILES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/audioFiles			
Portal UI > Menu: PBX > Sub-menu: PBX Settings > Tile: On Hold Music			
Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Tile: VoiceMail Box			
Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Tile: Dial by Name			
Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Link: Advanced Call Distribution ACD > Tile: Audio Files			
Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Link: Interactive Voice Response IVR > Tile: Audio Files			
id	Primary key	[1 - n]	Unique AudioFile-Id, assigned automatically by the DB at creation time
name	VARCHAR [45]	String	Name of the audio-file
number	VARCHAR [45]	String	Private phone number a VoiceMail Box message belongs to
duration	BIGINT	[1 - n]	Duration of the audio file in milli seconds
type	VARCHAR [12]	[greeting message static prompt]	"greeting" : Personal VoiceMail Box welcome "message" : VoiceMail Box message "static" : Audio file for ACD/IVR, PBX music on hold "prompt" : Auto attendant – Dial by name
addressId	INT	[1 - n]	Address-Id this audio-file belongs to
time	BIGINT	[1 - n]	Unix timestamp of creation or upload of the audio-file
flags	INT	[0 – n]	

OBJECT-KEY: CDRS

cdrs	Var Type [max Length]	Value	Remark
https://DOMAIN_REST/rest/ cdrs			
Portal UI > no page			

Note

For details about the available CDR fields, see the training documentation:
"Rating & Call Detail Record CDR"
(doc id: training_as7_706_sys_rating_cdr)

OBJECT-KEY: CONTACTS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/contacts			
Portal UI > Menu: PBX > Sub-menu: Contacts Portal UI > Menu: PBX Member > Sub-menu: Contacts			
id	Primary key	[1 - n]	Unique Contact-Id, assigned automatically by the DB at creation time
type	VARCHAR [32]	[none block]	"block" : Incoming calls to this contact will be blocked
name	VARCHAR [32]	String	Name of the contact
telNumber	VARCHAR [32]	String	Phone number with local writing customs e.g.: "012 (345) 67 - 89 "
telNumberNormalized	VARCHAR [32]	String	Phone number with striped special and white characters e.g.: 0123456789 → This number will be dialed
shortNumber	VARCHAR [32]	String	Short phone number of this contact
orgUnitId	INT	[1 - n]	OrgUnit-Id this contact belongs to

OBJECT-KEY: FORWARDS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/forwards			
Portal UI > Menu: PBX > Sub-menu: Extensions > Extension Settings > Tile: Number Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Tile: Call Forwarding			
id	Primary key	[1 - n]	Unique Forward-Id, assigned automatically by the DB at creation time
addressId	INT	[1 - n]	Address-Id this forward belongs to
destination	VARCHAR [45]	Number	Phone number or *#-code, e.g. *86300
type	VARCHAR [8]	[cfu cfb cfnr cff]	"cfu" : call forward unconditional "cfb" : call forward if busy "cfnr" : call forward if not reachable "cff" : call forward fallback
flags	INT	[0 – n]	
delay	INT	[0 – n]	Delay in milli seconds, e.g. 1'500 (equals 1.5sec)
timePattern	VARCHAR [32]	String	

OBJECT-KEY: HOLIDAYS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/holidays			
Portal UI > Menu: PBX > Sub-menu: Holidays			
id	Primary key	[1 - n]	Unique Terminal-Attribute-Id, assigned automatically by the DB at creation time
orgUnitId	INT	[1 - n]	OrgUnit-Id of the PBX or department
name	VARCHAR [45]	String	Name of the holiday
day	INT	[1 - 31]	Number of day
month	INT	[1 - 12]	Number of month
year	INT	yyyy	Number of year, e.g. 2023

OBJECT-KEY: LOCATIONS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/locations			
Portal UI → none			
id	Primary key	[1 - n]	Unique Location-Id, assigned automatically by the DB at creation time
terminalId	INT	[1 - n]	Terminal-Id this location belongs to
addressId	INT	[1 - n]	Address-Id this location belongs to
ip	VARCHAR [45]	String	
port	INT	[1 - n]	
contact	VARCHAR [256]	String	
expires	BIGINT	[0 - n]	
endpointId	INT	[1 - n]	
userAgent	VARCHAR [256]	String	
route1	VARCHAR [256]	String	
route2	VARCHAR [256]	String	
flags	INT	[0 - n]	

OBJECT-KEY: ORGUNIT & ORGUNITATTRIBUTES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/orgUnits			
Portal UI > Menu: PBX > Sub-menu: Organization Units			
id	Primary key	[1 - n]	Unique OrgUnit-Id, assigned automatically by the DB at creation time
name	VARCHAR [45]	String	Unique name
description	VARCHAR [256]	String	Free description
type	VARCHAR [45]	[system tenant pbx extension department]	
parentId	INT	[1 - n]	Parent OrgUnit-Id this OrgUnit belongs to

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/orgUnitAttributes			
Portal UI > Menu: PBX > Sub-menu: Organization Units > Tile: All Attributes			
id	Primary key	[1 - n]	Unique OrgUnit-Attribute-Id, assigned automatically by the DB at creation time
name	VARCHAR [128]	String	Names & values , see separate lists: ► For PBX: see page "List of Important PBX 'OrgUnit Attributes' Names & Values"
value	VARCHAR [128]	String	
orgUnitId	INT	[1 - n]	OrgUnit-Id this attribute belongs to

LIST OF IMPORTANT PBX 'ORGUNITATTRIBUTES' NAMES & VALUES

Name of OrgUnit Attribute	Value	Remark
Portal UI > Menu: PBX > Sub-menu: PBX Settings > Tile: Properties		
maxExternalChannels	[0 – n]	Default: 0, no limitations
maxServiceExtensions	[0 – n]	Default: 0, no limitations
maxExtensions	[0 – n]	Default: 0, no limitations
timezone	"timezone[01-xx]"	Example: UTC+3 → "timezone03"
dateformat	dd.MM.yyyy	
timeformat	HH:mm	
ctiDomain	String	CTI-Domain of a PBX
Portal UI > Menu: Operations > Sub-menu: Organization Units		
cloudId	String	an IP-Phone Cloud ID
ctiDomain	String	CTI-Domain of a PBX
Portal UI > Menu: PBX > Sub-menu: PBX Settings > Tile: On Hold Music		
musicOnHoldId	Number	See examples "Upload an Audio-File for PBX Extension "Sales IVR"" how to get the ID of an audio-file.

Note

The "Valid from" and "Valid until" is defined per public number, see "Create the Public Phone Numbers of PBX "PA-PBX-0B""

OBJECT-KEY: TERMINALS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/terminals			
Portal UI > Menu: PBX > Sub-menu: Extension Related Features > Tile: Assigned Phones			
id	Primary key	[1 - n]	Unique Terminal-Id, assigned automatically by the DB at creation time
username	VARCHAR [45]	String	SIP username
password	VARCHAR [45]	String	SIP password → will be stored encrypted in the DB
orgUnitId	INT	[1 - n]	PBX Extension OrgUnit-Id this terminal belongs to
userId	INT	[1 - n]	User-Id this terminal belongs to
emergencyLocationId	INT	[1 - n]	
name	VARCHAR [45]	String	Name of this terminal → Choose a unique name. This helps for identification.
type	VARCHAR [45]	[terminal type/profile]	Terminal type of predefined phone types: "anConnect" "anIpPhone" "anIpDeskphone" "yealinkT21PE2" "snomD785" "grandstreamGRP2613" ... or terminal profile of a phone template: "profile1" ...

OBJECT-KEY: TERMINALATTRIBUTES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/ terminalAttributes			
Portal UI > Menu: PBX > Sub-menu: Phones > Tile: Phone Setup Portal UI > Menu: PBX > Sub-menu: Phones > Phone Related Features > Tile: Phone Notification Portal UI > Menu: PBX > Sub-menu: Phones > Phone Related Features > Tile: Phone Keys			
id	Primary key	[1 - n]	Unique Terminal-Attribute-Id, assigned automatically by the DB at creation time
name	VARCHAR [128]	String	Names & values , see separate lists on page "List of Important 'terminalAttributes' Names & Values"
value	VARCHAR [128]	String	
terminalId	INT	[1 - n]	Terminal-Id this attribute belongs to

LIST OF IMPORTANT 'TERMINALATTRIBUTES' NAMES & VALUES

Name of Terminal Attribute	Value	Remark
Portal UI > Menu: PBX > Sub-menu: Phones > Phone Setup		
provisionOpt	[autoprov manual sip clicktocall anDesktop anipphone dectphone]	"autoprov": Configuration via redirections server and terminal MAC address "manual": Configuration of the config-download URL in the terminal "sip": Configuration completely manual of the terminal "clicktocall": Configuration for CTI base WebPhone or click-to-call button on a Web page
mac	00:00:00:00:00:00	MAC address needed with provisioning option "autoprov"
oneTimeCode	String	Return value needed with "manual" e.g.: 469b40bc446bde4f https://DOMAIN/cfg/469b40bc446bde4f701c9413fe35bb11d5acc345803baf0e.xml
webAdminname	"admin"	
webAdminPassword	String	Define the password string → will be stored encrypted in the DB
webUsername	"user"	
webUserPassword	String	Define the password string → will be stored encrypted in the DB
timezone	"timezone[01-xx]"	Example: UTC+3 → "timezone03"
Portal UI > Menu: PBX > Sub-menu: Phones > Phone Related Features > Tile: Phone Notification		
alertExternal	[1 - n]	Ringtone number 1 – n, the max. number depends on the phone type
alertInternal	[1 - n]	Ringtone number 1 – n, the max. number depends on the phone type
Portal UI > Menu: PBX > Sub-menu: Phones > Phone Related Features > Tile: Phone Keys		
fkey[n]	[fkey1 – fkeyn]	fkey[n]: n = 1 – the max. key number depends on the phone type
fkey[n]	"speed/608/0987654321/Customer C"	Speed Dial: [speed/Address-ID of PBX extension/destination/label]
fkey[n]	"team/608/711/Front Desk"	Team key: [team/Address-ID of PBX extension/destination/label]
fkey[n]	"orbit/608/1/Park Call"	Park key: [orbit/Address-ID of PBX extension/1/label]
fkey[n]	"line/609//711 Front Desk"	Line key: [line/Address-ID of PBX extension the number belongs to/label]

OBJECT-KEY: USERS

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/OrgUnits/users			
Portal UI > Menu: PBX > Sub-menu: Users			
id	Primary key	[1 - n]	Unique User-Id, assigned automatically by the DB at creation time
name	VARCHAR [45]	String	
firstName	VARCHAR [45]	String	First name
lastName	VARCHAR [45]	String	Last name
password	VARCHAR [45]	String	Define the password string → will be stored encrypted in the DB
language	VARCHAR [45]	[en de fr it vi]	Used language in the Portal UI and default system notifications "en" : English "de" : German "fr" : French "it" : Italian "vi" : Vietnamese
email	VARCHAR [45]	string@string.string	Valid email address → Must be unique in the anSwitch V7 system!
pin	VARCHAR [45]		
pinFails	INT	[0 – n]	
pinBlocked	BIGINT	[0 – n]	
passwordFails	INT	[0 – n]	Counter of failed logins
passwordBlocked	BIGINT	[0 >0]	>0: the login is blocked
orgUnitId	INT	[1 - n]	OrgUnit-Id of the PBX Extension this user account belongs to
flags	INT	[0 - n]	

OBJECT-KEY: USERROLES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/OrgUnits/userRoles			
Portal UI > Menu: PBX > Sub-menu: Users > Tile: Roles			
id	Primary key	[1 - n]	Unique UserRole-Id, assigned automatically by the DB at creation time
userId	INT	[1 - n]	User Account Id this role belongs to
orgUnitId	INT	[1 - n]	OrgUnit-Id the user account with this role have access rights
role	VARCHAR [45]	["admin" "operator" "pbx" "department" "extension" "rest"]	"admin" : Administrator "operator" : Operator "pbx" : PBX Administrator "department" : Department Administrator "extension" : PBX Member "rest" : REST API
flags	INT	[0 - n]	

OBJECT-KEY: USERATTRIBUTES

DB Property Name	DB Data Type	Allowed Values	Remark
https://DOMAIN_REST/rest/ userAttributes			
Portal UI > Menu: Operator > Sub-menu: Users > Tile: Attributes			
id	Primary key	[1 - n]	Unique Terminal-Attribute-Id, assigned automatically by the DB at creation time
name	VARCHAR [45]	String	Names & values , see separate lists on page "List of Important 'userAttributes' Names & Values"
value	VARCHAR [256]	String	
userId	INT	[1 - n]	Terminal-Id this attribute belongs to

LIST OF IMPORTANT 'USERATTRIBUTES' NAMES & VALUES

Name of Terminal Attribute	Value	Remark
Portal UI > Menu: Operator > Sub-menu: Users > Tile: Attributes		
ctiDeviceId	sip:<INTERNAL_ NUMBER>@<PBX_DOMAIN_NAME>	CTI identification of an agent's PBX Extension which is monitored by a CTI application. Example: sip:21@pbx.customer.com

5 ERROR HANDLING & TROUBLE SHOOTING

INTRODUCTION OF ERROR HANDLING

- ▶ Host & Path Problems
 - ▶ Host, server or service not reachable or doesn't reply
- ▶ Authentication Problems
 - ▶ HTTP Status Code: 401 → Login is not possible
- ▶ OrgUnit & Instance Problems
 - ▶ HTTP Status Code: 403, 404 → Instance doesn't exist or not allowed to write to
- ▶ Parameter & Value Problems
 - ▶ HTTP Status Code: 404 → Parameters doesn't exist or not allowed to write to

Note

The anSwitch V7 REST server doesn't return an error code or message when:

- ▶ A parameter was configured with an invalid value.
- ▶ The parameter was not configured at all.

➔ To solve this problem, see section "Best Practices":

[How to Check a Newly Configured Parameter Value?](#)

HOST & PATH PROBLEMS

► Host & Path Problems:

Error Message/Code or Error Behavior	Meaning	Action	Remarks
No response from the host Possible error messages from the REST client application: ► connect ETIMEDOUT IP_ADDRESS_HOST:IP_PORT ► Connection timeout	The requested URL can't be reached. The service might be temporarily down, or it may have moved permanently to a new web address.	<ul style="list-style-type: none">► Check if the host IP address is reachable via the IP network► Check if the Portal component is running► Check if the URL, IP address, IP ports and path are correct	

AUTHENTICATION PROBLEMS

► Authentication Problems

Error Message/Code or Error Behavior	Meaning	Action	Remarks
Code: 401	Unauthorized The username and/or password are incorrect or doesn't exist.	<ul style="list-style-type: none">► Check if the user exists► Check if the user's role is "Rest API"► Check the username and password	

ORGUNIT & INSTANCE PROBLEMS

► OrgUnit & Instance Problems

Error Message/Code or Behavior	Meaning	Action	Remarks
Code: 403	The user is not allowed to write to this instance. The user is not allowed to create this instance.	► Check if the user has the right to access the OrgUnit	
Code: 404	Instance does not exist The OrgUnit or instance system-ID doesn't exist or was deleted.	► Check if the OrgUnit exists ► Check if the instance system-ID exists	

PARAMETER & VALUE PROBLEM

▶ Parameter & Value Problem

Error Message/Code or Behavior	Meaning	Action	Remarks
Code: 400	Bad request Unsupported format	▶ ?????	PUT with empty body: { }
Code: 403	Not allowed to write this instance The requested parameter (key) doesn't exist	▶ Check if you have a typo in the parameter name ▶ Check if the object type of the instance exists	

TROUBLE SHOOTING

▶ Check as Administrator the Portal Log.

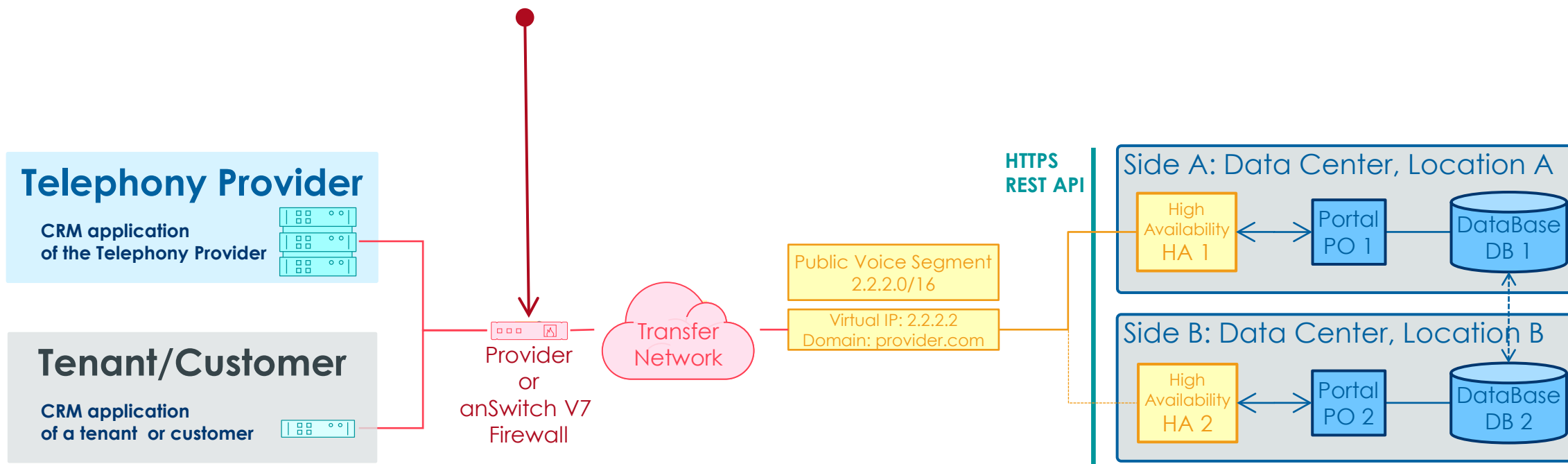
- > Menu: Operator
 - > Sub-Menu: Logs
 - > Select Log Type: Portal
 - > Search for Regex Pattern, e.g.: `RestServlet`
-

- ▶ For object, instance and property problems check for key words like:
 - ▶ `RestServlet`
 - ▶ `REST API`
- ▶ For CRM authentication problems check for key words like:
 - ▶ `Authenticator`
 - ▶ `PermissionConfig`

6 SETTING UP THE REST API INTERACTION

IP CONNECTIVITY TO THE REST API

1. Make sure that the IP routing between the provider's and or customer's CRM and the Public VoIP segment of the anSwitch V7 is end-to-end.
 - ▶ The REST API is accessible via the virtual IP address or a domain name, e.g.:
`https://provider.com/rest`
 - ▶ Configure appropriate firewall policies: anSwitch REST → **HTTPS, TCP port: 443**



CREATE A USER ACCOUNT FOR THE CRM APPLICATION

2. Via the Portal GUI, an administrator or operator configures a user account for the CRM application access with read/write permissions for the desired OrgUnit.

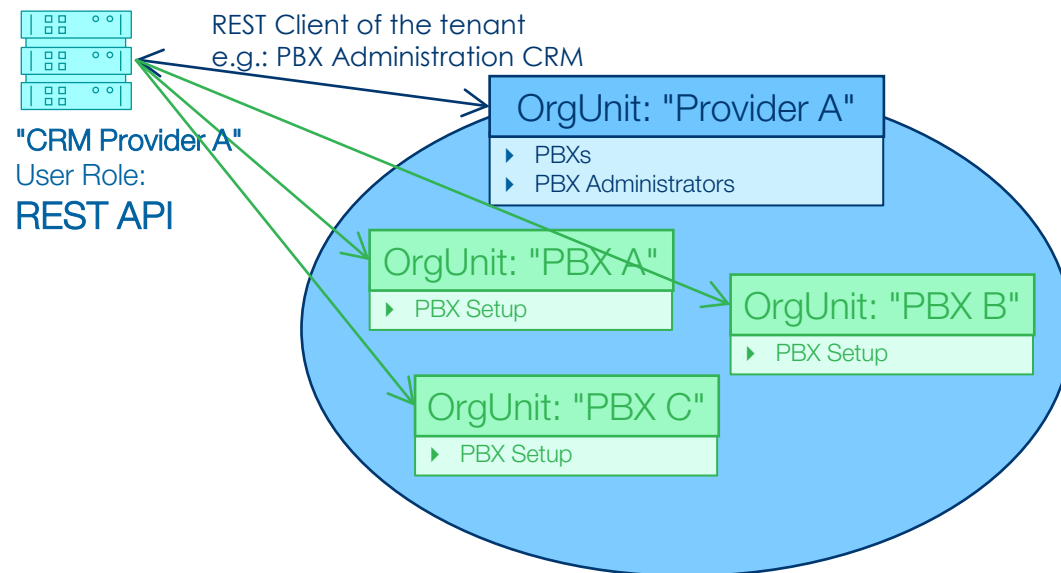
- ▶ For example:
 - ▶ A tenant must be enabled to manage its PBXs so the CRM needs access to OrgUnit "Provider A".

- ▶ Create the user for the CRM access.

> Menu: Operations
> Sub-Menu: Users

- ▶ The essential parameters needed for the configuration:

Name	Value	Remark / Example
E-mail	Username, e.g.: crm@provider-a.com
Password	
Role	Rest API	
Access to OrgUnit	



User

Users

Save

Base Settings

E-Mail
crm@provider-a.com

First Name
CRM

Last Name
Provider A

Language
English

Settings

Owner
Provider A

Password
.....

LDAP User Name

User blocked

Roles

+ New × Delete

Role	Access to OrgUnit	Parent OrgUnit
Rest API	Provider A	system

New role

Save

Select the Role
Rest API

OrgUnit
Provider A

PREPARE THE CUSTOMERS REST CLIENT

3. Set up the CRM application of the customer:

- ▶ Configuring the CRM application for
 - ▶ IP connectivity
 - ▶ REST authentication with the assigned user account credentials.
- ▶ Write code for managing the anSwitch V7 database via the REST API
 - ▶ Writing code to handle the REST API is not rocket science, but it must be done with the care of an accountant.
→ See the Warning!
- ▶ It is **strongly** recommended to write and test the code on an anSwitch V7 test system.
 - ▶ Use a free REST application for experimenting with REST requests, see [section "'Insomnia" a REST Client Application"](#).
 - ▶ Consult the examples, see section ["Example: Create PBX & PBX Extension"](#) and following.

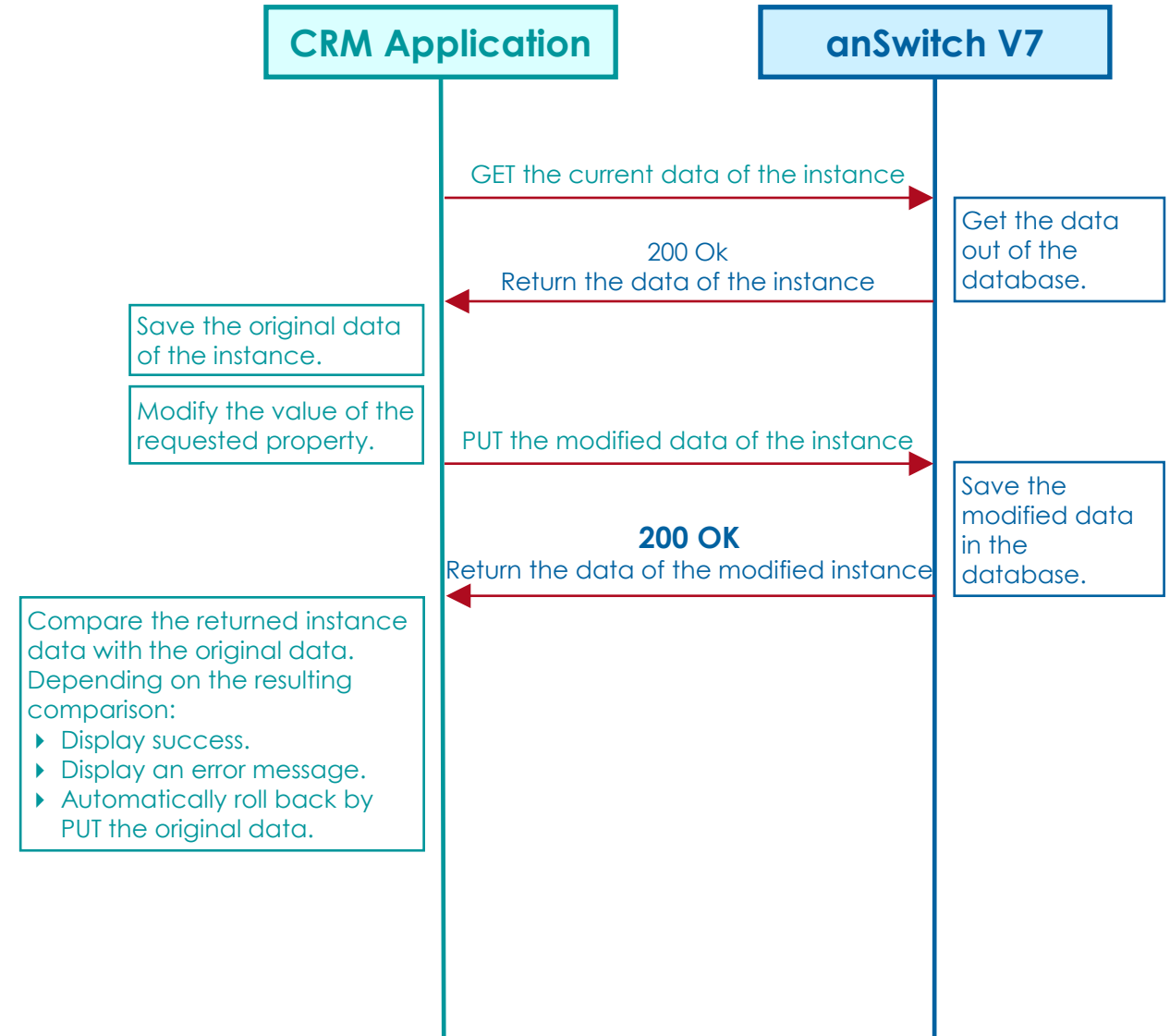
7 BEST PRACTICES

HOW TO TEST NEW POST, PUT, GET AND DELETE REQUESTS?

- ▶ How to test new POST, PUT, GET and DELETE requests?
 - ▶ For exploring and testing new requests for the configuration of parameters it is recommended to use in the first place a 3rd party tool before writing code for the own CRM system.
 - ▶ For MS Windows the application "Insomnia" by Kong in its free version is fully sufficient for testing purposes, see section ["Insomnia" a REST Client Application](#)

HOW TO CHECK A NEWLY CONFIGURED PROPERTY VALUE?

- ▶ How to check a newly configured property value?
 - ▶ In principle, it can be assumed that if a POST or PUT was confirmed with **200 OK**, the data was saved correctly in the anSwitch V7 database.
 - ▶ Error messages are described in section "[Error Handling & Trouble Shooting](#)"
 - ▶ If a CRM application wants to make sure the process on the right side is recommended.



HOW TO GET PROPERTY VALUES?

- ▶ The data base objects, its properties and allowed values are described in section "Manageable DataBase Objects".

8 "INSOMNIA" A REST CLIENT APPLICATION

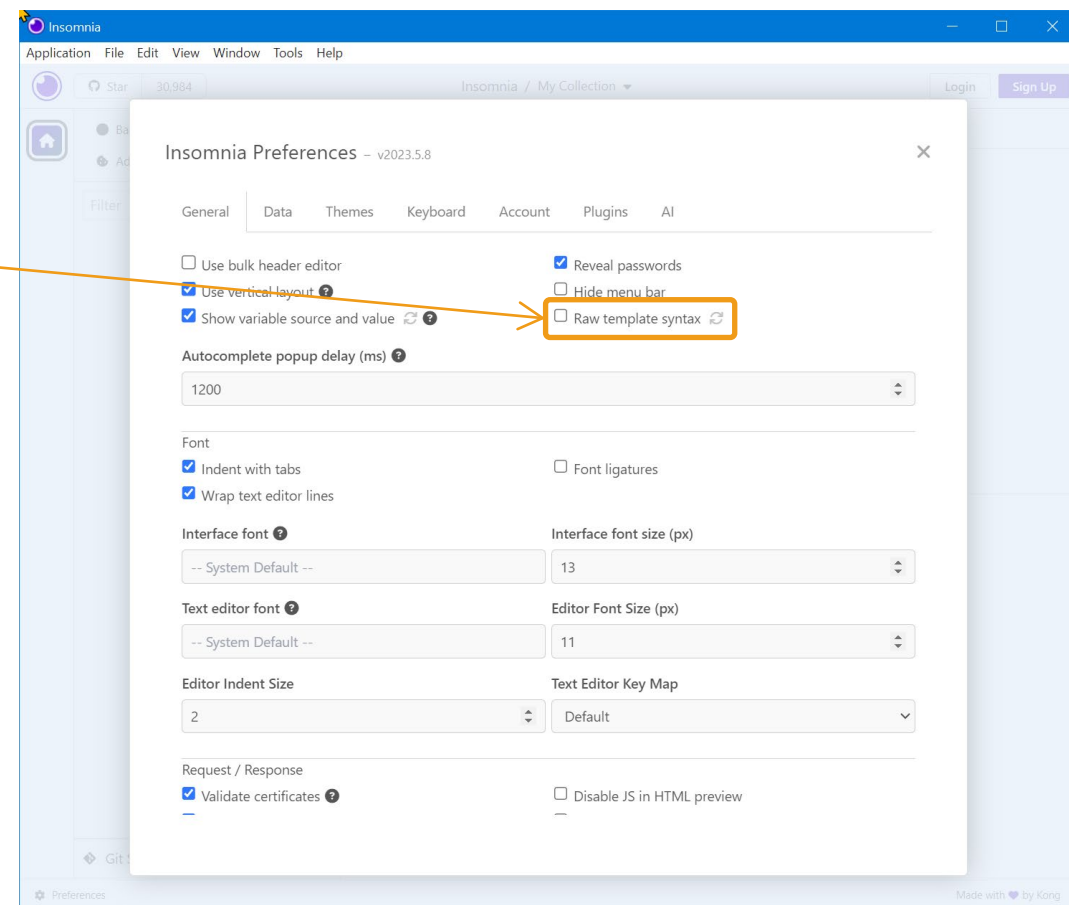
OVERVIEW "INSOMNIA" A REST CLIENT APPLICATION

- ▶ For testing GET, POST, PUT, DELETE commands with the anSwitch V7 REST API it is a good idea to use a 3rd party product.
- ▶ For MS Windows the application "Insomnia" by Kong in its free version is fully sufficient for testing purposes
<https://insomnia.rest>
- ▶ It is easy to use!
- ▶ Download Insomnia installer
<https://insomnia.rest/download>

BASIC CONFIGURATIONS

► Set preferences:

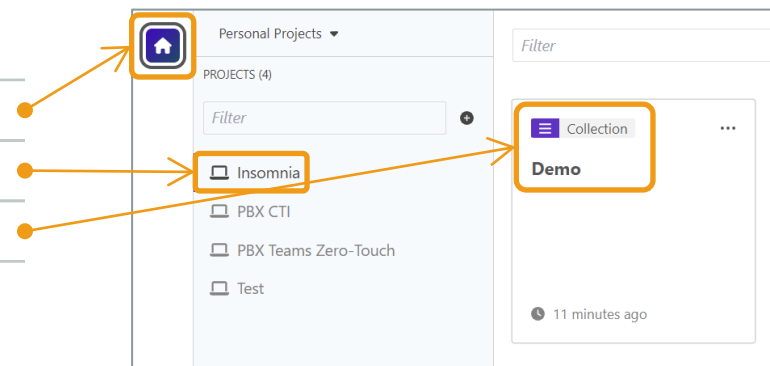
- > Menu: Application
 - > Sub-Menu: Preferences
 - > Select tab: General
 - > Configure parameter:
 - Raw template syntax: Disable it



CONFIGURE ENVIRONMENT VARIABLES FOR THE CRM ACCESS

► Navigate to the collection:

- > Select the "Personal Projects"
- > Select the desired "Project", e.g. Insomnia
- > Open the desired "Collection", e.g. Demo



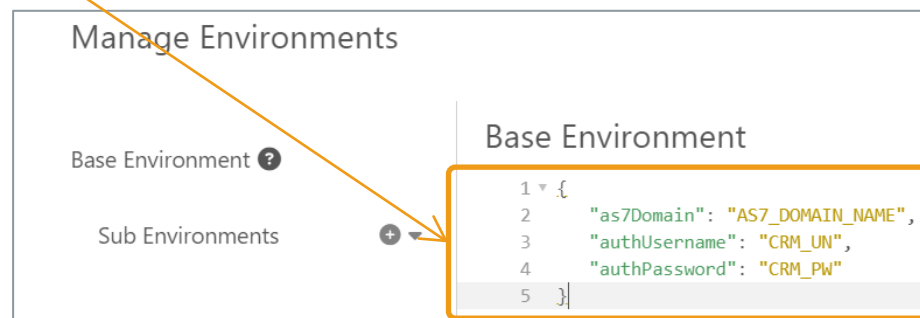
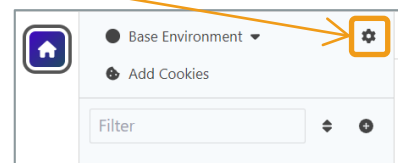
► Configure the environment variables:

- > At "Base Environment" click the "Gear" icon
- > Configure the essential environment variables

```
1 {  
2   "as7Domain": "AS7_DOMAIN_NAME",  
3   "authUsername": "CRM_UN",  
4   "authPassword": "CRM_PW"  
5 }
```

← Insert the real domain name
← Insert the real username
← Insert the real password

- > Click button: Close



CONFIGURE THE FIRST GET REQUEST AS TEMPLATE

- ▶ Configure the first request as template and apply the values of the basic environment variables.

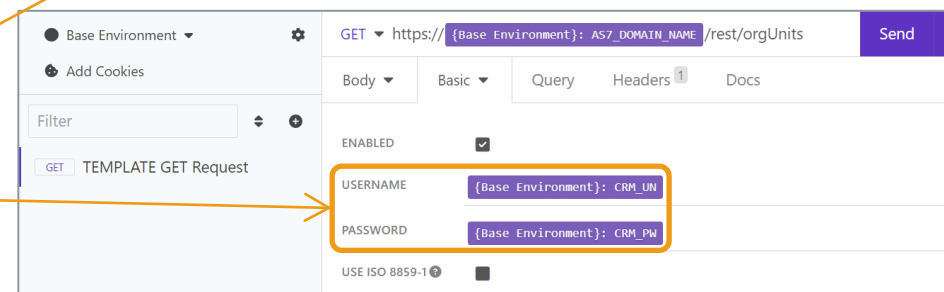
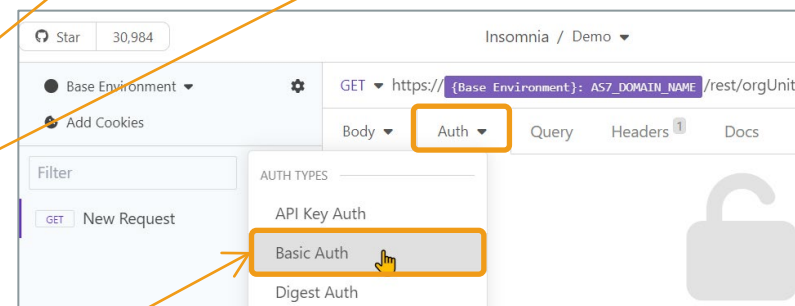
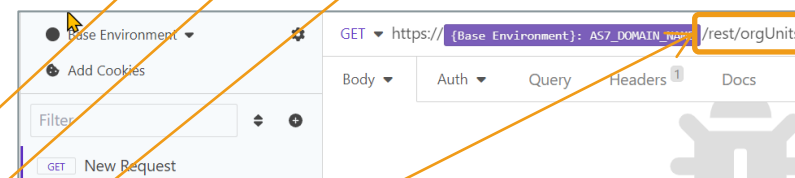
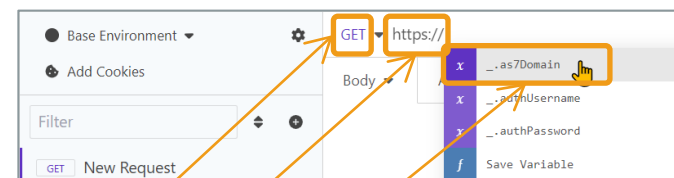
- ▶ Later duplicate this template for further requests and you have all done for the interaction with the anSwitch V7.

a. Configure the GET request URL

- > Configure the request:
 - ▶ Select the request type: GET
 - ▶ Start with the Url: https://
 - ▶ Insert the domain name from the environment variable: Ctrl + Space → Select "_as7Domain"
 - ▶ Continue the path and REST object, e.g.: /rest/orgUnits

b. Configure the authentication.

- ▶ Use the environment variables
 - > Select tab "Auth" :
 - ▶ Select: Basic Auth
 - ▶ Insert the authentication values from the environment variables: Ctrl + Space → Select "_authUsername", "_authPassword"
 - > Fire your first request → click button: Send



CONFIGURE THE FIRST POST REQUEST AS TEMPLATE

► Configure a POST request as template.

a. Make a copy of the GET request template

- > At "TEMPLATE GET Request":
 - Select: Duplicate
- > Rename the request: TEMPLATE POST Request

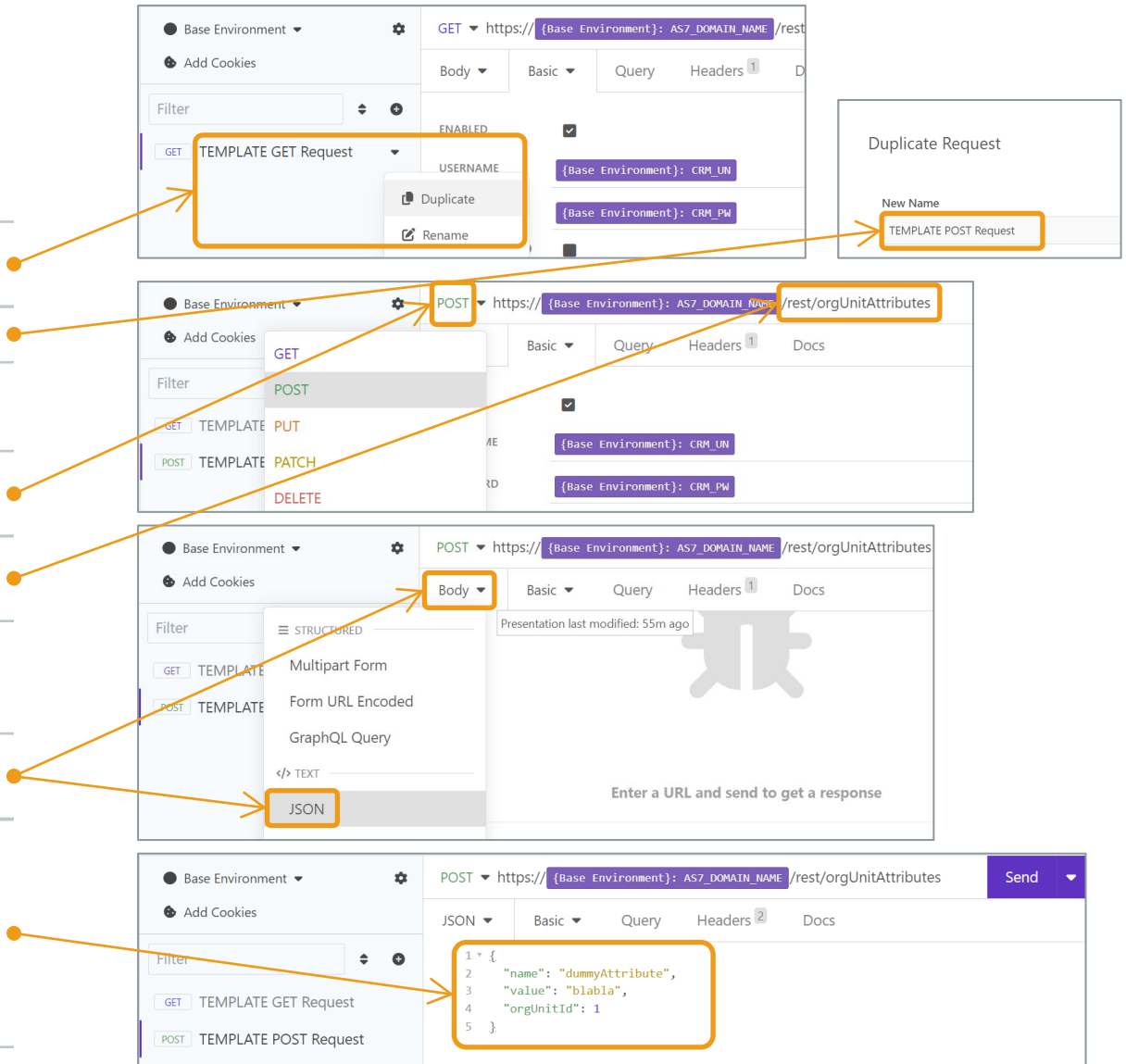
b. Configure the POST request URL

- > Configure the request:
 - Select the request type: POST
- > Continue the path and REST object to be created, e.g.:
/rest/orgUnitAttributes

c. Configure the body of the POST.

- > Select tab "Body":
 - Select: JSON
- > Insert the body contents in JSON notation, e.g.:

```
1 {  
2   "name": "dummyAttribute",  
3   "value": "blabla",  
4   "orgUnitId": 1  
5 }
```



9 EXAMPLE: CREATE PBX & PBX EXTENSION

EXAMPLE OVERVIEW: CREATE PBX & PBX EXTENSION

► Goal:

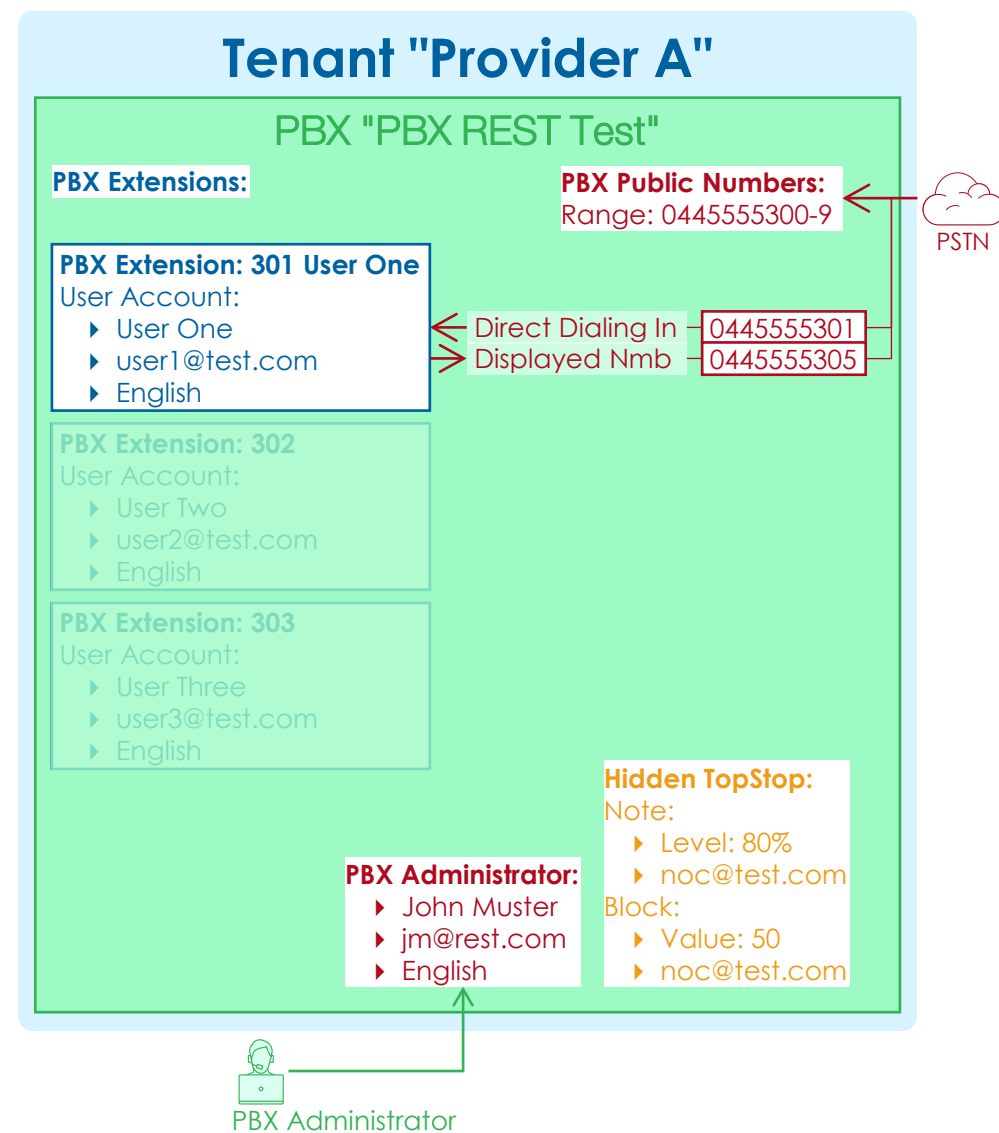
- Create a basic PBX and its PBX Extensions.

► Data to prepare:

- Values used in the example, see picture on the right side.
- ☐ Have ready the OrgUnit ID of the parent Tenant.
- ☐ Unique PBX Name
- ☐ Description of the PBX
- ☐ Hidden TopStop
 - ☐ Value, note/block, email addresses
- ☐ Public numbers and number ranges
- ☐ User account for the PBX Administrator
 - ☐ Email address, first & last name, language
- ☐ PBX Extensions & internal Number & DDI
 - ☐ Private number, display name, DDI, displayed public number
- ☐ User account for the PBX Member
 - ☐ Email address, first & last name, language

► Assumptions:

- Inherited from any parent OrgUnit
 - CTI basic configuration
 - PBX pricelists
 - PBX limits, e.g. extension types, channels
- Access to the PSTN via the system default routes and gateways.

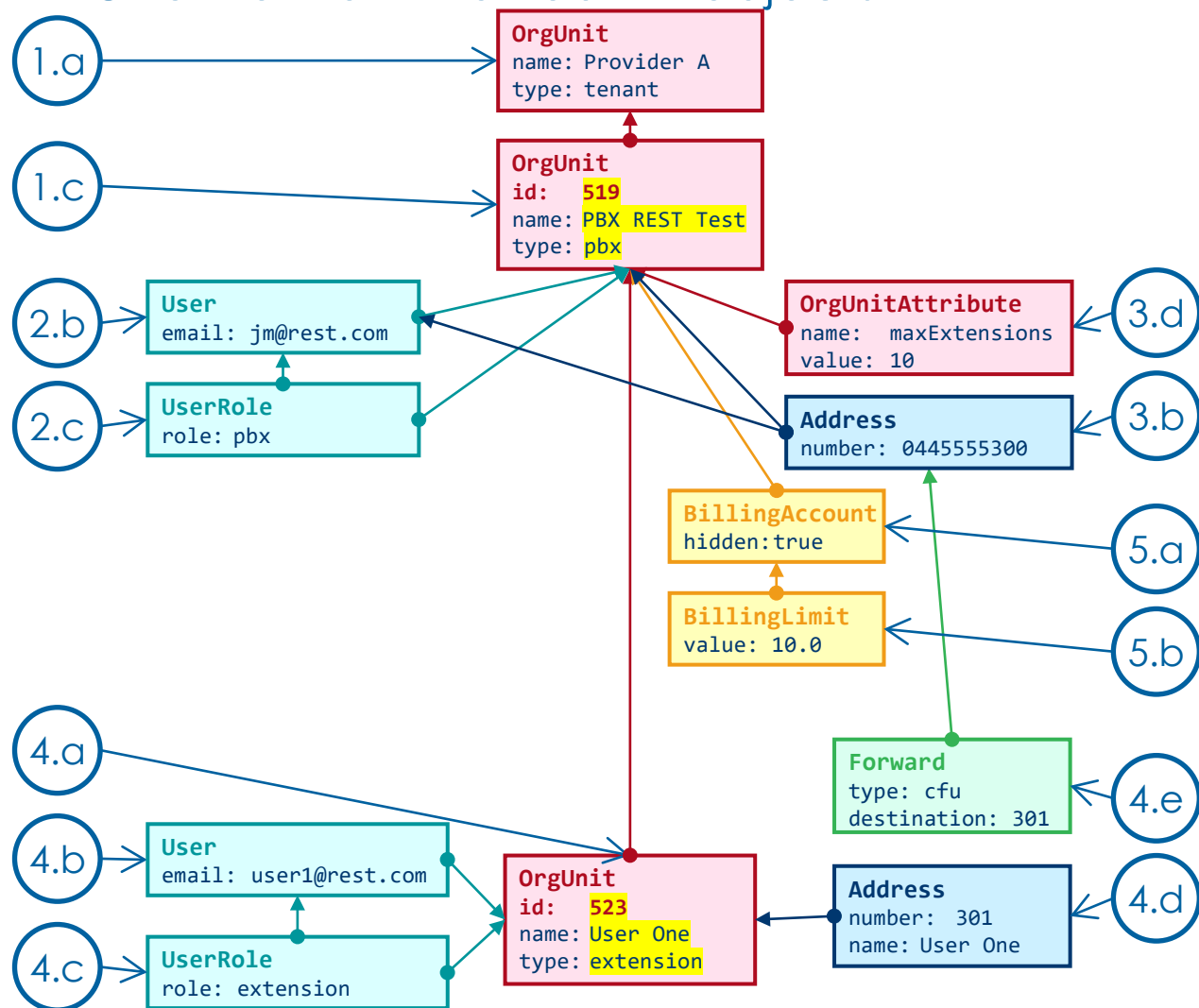


EXAMPLE OVERVIEW: CREATE PBX & PBX EXTENSION

► Overview of the execution steps:

1. Create PBX "PBX REST Test".
 - a. GET: Get OrgUnit ID of Tenant "Provider A"
 - b. GET: Check if new PBX name "PBX REST Test" is already used
 - c. POST: Create PBX "PBX REST Test"
2. Create PBX Administrator.
 - a. GET: Check if a user with email "jm@rest.com" already exists.
 - b. POST: Create user account of "jm@rest.com"
 - c. POST: Assign role "PBX Administrator" to user jm@rest.com
3. Assign PBX public numbers, limits, etc.
 - a. GET: Check if an address with number "0445555300" already exists.
 - b. POST: Create the public numbers "0445555300-9".
 - c. GET: Get data of all public numbers of the PBX.
 - d. POST: Create the PBX limitations, e.g. channel limits.
4. Create PBX extension with internal number, user account and DDI.
 - a. POST: Create PBX Extension of "User One".
 - b. POST: Create user "User One".
 - c. POST: Assign the user role.
 - d. POST: Assign the private number 301.
 - e. POST: Assign DDI & displayed public number for 301.
5. Create hidden TopStop.
 - a. POST: Create the TopStop billing account for the PBX.
 - b. POST: Assign the TopStop limit conditions.

► Overview of involved DB objects



EXAMPLE CODE: CREATE PBX

1. Create PBX "PBX REST Test"

a. GET: Get OrgUnit ID of Tenant "Provider A"

GET https://DOMAIN_REST/rest/orgUnitsorgUnits?where=name.like('Provider A')&properties=id

No body

a. Response

200 OK

```
{
  "orgUnits": [
    {
      "id": 25
    }
  ]
}
```

← OrgUnit ID of the tenant, save for later use!

b. GET: Check if new PBX name "PBX REST Test" is already used.

GET https://DOMAIN_REST/rest/orgUnitsorgUnits?where=name.like('PBX REST Test')&properties=id

No body

b. Response

200 OK

```
{
  "orgUnits": []
}
```

The expected result is "empty".
If not, then a PBX with this name exists.

c. POST: Create OrgUnit ID of PBX "PBX REST Test"

POST https://DOMAIN_REST/rest/orgUnits

```
{
  "name": "PBX REST Test",
  "type": "pbx",
  "description": "PBX of company REST",
  "parentId": 25
}
```

← Mandatory value

c. Response

200 OK

```
{
  "name": "PBX REST Test",
  "id": 519,
  "type": "pbx",
  "parentId": 25,
  "description": "PBX of company REST"
}
```

← OrgUnit ID of the PBX, save for later use!

EXAMPLE CODE: CREATE PBX ADMINISTRATOR

2. Create PBX Administrator

a. GET: Check if a user with email "jm@rest.com" already exists.

GET

[https://DOMAIN_REST/rest/users?where=email.like\(jm@rest.com\)&properties=id](https://DOMAIN_REST/rest/users?where=email.like(jm@rest.com)&properties=id)

No body

a. Response

200 OK

```
{
  "orgUnits": []
}
```

The expected result is "empty".
If not, then a user with this email exists.

```
{
  "orgUnits": [
    {
      "id": 111
    }
  ]
}
```

If there is an ID, then check if this user really is the expected user. If yes, then go directly to 2.c)

← User ID of the PBX administrator to be, save for later use!

b. POST: Create user account of "jm@rest.com"

POST https://DOMAIN_REST/rest/users

```
{
  "firstName": "John",
  "lastName": "Muster",
  "email": "jm@rest.com",
  "language": "en",
  "orgUnitId": "519"
}
```

← Set no password, this the user shall do on his first login.

b. Response

200 OK

```
{
  "name": null,
  "id": 111,
  "language": "en",
  "password": null,
  "pin": null,
  "orgUnitId": "519",
  "email": "jm@rest.com",
  "pinFails": 0,
  "pinBlocked": 0,
  "passwordFails": 0,
  "passwordBlocked": 0,
  "firstName": "John",
  "lastName": "John",
  "sendEmailVoicemail": false
}
```

← Remember the user ID for assigning the role.

EXAMPLE CODE: CREATE PBX ADMINISTRATOR

2. Create PBX Administrator

c. POST: Assign role "PBX Administrator" to user jm@rest.com.

POST https://DOMAIN_REST/rest/userRoles

```
{  
  "userId": "111",  
  "role": "pbx",  
  "orgUnitId": "519"  
}
```

← Mandatory value!

c. Response

200 OK

```
{  
  "userId": "111",  
  "role": "pbx",  
  "orgUnitId": "519"  
}
```

EXAMPLE CODE: ASSIGN PBX PUBLIC NUMBERS, LIMITS, ETC.

3. Assign PBX public numbers, limits, etc.

- a. GET: Check if an address with number "0445555300" already exists.

```
GET https://DOMAIN_REST/rest/addresses?where=number.like('0445555300')&properties=id
```

No body

- b. POST: Create public number of " 0445555300 ".
‣ 3.b) must be done for every single public number!
‣ Make sure always to use the same epoch for "validFrom".
Even when you later add additional public numbers.

```
POST https://DOMAIN_REST/rest/addresses
```

```
{
  "number": "0445555300",
  "orgUnitId": 519,
  "validFrom": 1699875180000
}
```

← Insert the epoch of the real date/time.

- c. GET: Get data of all public numbers of the PBX.

```
GET https://DOMAIN_REST/rest/addresses?where=orgUnitId.eq(519)
```

No body

- a. Response

200 OK

```
{
  "addresses": []
}
```

The expected result is "empty".
If not, then a number with this number exists.

```
{
  "addresses": [
    {
      "id": 720
    }
  ]
}
```

If there is an ID, then check if this address really is assigned to this PBX.

- b. Response

200 OK

```
{
  "name": null,
  "id": 728,
  "number": "0445555300",
  "orgUnitId": 519,
  ...
  "validFrom": 1699875180000,
  "validUntil": 9223372036854775807,
  ...
}
```

← Remember the address ID for:
‣ Assigning the DDI
‣ Deleting the number

EXAMPLE CODE: ASSIGN PBX PUBLIC NUMBERS, LIMITS, ETC.

3. Assign PBX public numbers, limits, etc.

- ▶ Several PBX parameters are configured as OrgUnit attributes, e.g. attributes with name:
 - ▶ maxExtension
 - ▶ maxServiceExtension
 - ▶ maxExternalChannels
 - ▶ timezone
 - ▶ dateformat
 - ▶ timeformat

- ▶ Process configuring an OrgUnit attribute:
 1. Check if the OrgUnit attribute for the PBX already exists.
 2. Depending on the result apply a POST for creating the attribute or a PUT for changing the attribute value.

d. GET: Check the existence of a PBX OrgUnit attribute.

GET
[https://DOMAIN_REST/rest/orgUnitAttributes?where=name.like\('maxExtensions'\).and\(orgUnitId.eq\(519\)\)](https://DOMAIN_REST/rest/orgUnitAttributes?where=name.like('maxExtensions').and(orgUnitId.eq(519)))
 No body

d. Response

200 OK	
{ "orgUnitAttributes": [] }	The OrgUnit attribute doesn't exist yet. → Do 3.e) for creating the attribute.
{ "name": "maxExtensions", "value": "5", "id": 746, "orgUnitId": 519 }	The OrgUnit attribute exists. → Do 3.f) for changing the value of the attribute.

e. POST: Create a PBX OrgUnit attribute.

POST https://DOMAIN_REST/rest/orgUnitAttributes
 {
 "name": "maxExtensions",
 "value": 5,
 "orgUnitId": 519
}

e. Response

200 OK	
{ "name": "maxExtensions", "value": "5", "id": 746, "orgUnitId": 519 }	If there is an ID, then check if this address really is assigned to this PBX.

f. PUT: Change a PBX OrgUnit attribute value.

PUT https://DOMAIN_REST/rest/orgUnitAttributes/746
 {
 "value": 10
}

d. Response

200 OK	
	Check the modified OrgUnit attribute data with a GET .../orgUnitAttributes/746

EXAMPLE CODE: PBX EXTENSION, INTERNAL NUMBER, DDI

4. Create PBX extension with internal number, user account and DDI.

- ▶ Every PBX Extension must be created individually!

a. POST: Create PBX Extension of "User One"

POST https://DOMAIN_REST/rest/orgUnits

```
{
  "name": "User One",
  "type": "extension",
  "parentId": 519
}
```

← Mandatory value

b. POST: Create user account "User One"

POST https://DOMAIN_REST/rest/users

```
{
  "firstName": "User",
  "lastName": "One",
  "email": "user1@rest.com",
  "password": "secret",
  "language": "en",
  "orgUnitId": 523
}
```

← Set a password !
This enables us to activate the auto provisioning of the phone types anDesktop and anConnect.

▶ Process configuring an OrgUnit attribute:

1. Create the PBX Extension
2. Assign the user account.
3. Assign the private number.

a. Response

200 OK

```
{
  "name": "User One",
  "id": 523,
  "type": "extension",
  "description": null,
  "parentId": 519
}
```

← Remember the extension name for the phone configuration.
← Remember OrgUnit ID of the PBX Extension for:
▶ User account and role

b. Response

200 OK

```
{
  "name": null,
  "id": 113,
  "language": "en",
  "password": "crypt:XEH7Z17q33VmOw==",
  "orgUnitId": 523,
  "pin": null,
  "email": "user1@rest.com",
  "pinFails": 0,
  "pinBlocked": 0,
  "passwordFails": 0,
  "passwordBlocked": 0,
  "firstName": "User",
  "lastName": "One",
  "sendEmailVoicemail": false
}
```

← Remember the user ID for :
▶ The user role
▶ Private number
▶ Terminal
← Note: The password is encrypted!

EXAMPLE CODE: PBX EXTENSION, INTERNAL NUMBER, DDI

4. Create PBX extension with internal number, user account and DDI.

c. POST: Assign role "PBX Administrator" to user user1@rest.com

POST https://DOMAIN_REST/rest/userRoles

```
{
  "userId": "113",
  "role": "extension",
  "orgUnitId": "523"
}
```

← Mandatory value!

c. Response

200 OK

```
{
  "id": 235,
  "orgUnitId": 523,
  "userId": 113,
  "role": "extension"
}
```

d. POST: Assign the private number 301

POST https://DOMAIN_REST/rest/addresses

```
{
  "number": "301",
  "name": "User One",
  "orgUnitId": 523,
  "userId": "113",
  "scopeOuld": "519"
}
```

← Name to display.
 ← OrgUnit ID of the new PBX Extension
 ← scopeOuld: OrgUnit ID of the PBX or a Department of this PBX.

d. Response

200 OK

```
{
  "name": "User One",
  "id": 743,
  "number": "301",
  "orgUnitId": 523,
  "userId": 113,
  ...
  "scopeOuld": 519,
  ...
}
```

← Remember the address ID for:

- ▶ Assigning the location of an associated terminal
- ▶ Obtaining the list of VoiceMail messages

e.

EXAMPLE CODE: PBX EXTENSION, INTERNAL NUMBER, DDI

4. Create PBX extension with internal number, user account and DDI.

e. POST: Assign DDI to 301.

- ▶ DDI is just an ordinary call forwarding unconditional CFU.

POST https://DOMAIN_REST/rest/forwards

```
{
  "addressId": 729,
  "destination": 301,
  "type": "cfu"
}
```

← Mandatory value.

PUT: Assign displayed public number 0445555305 to 301.

PUT https://DOMAIN_REST/rest/addresses/743

```
{
  "publicAddrId": 733
}
```

← Address ID of public number 0445555305

e. Response

200 OK

```
{
  "id": 664,
  "type": "cfu",
  "destination": "301",
  "delay": 0,
  "timetableId": 0,
  "addressId": 729,
  "inactive": false,
  "timePattern": null
}
```

← Remember the address ID for:

- ▶ Assigning the DDI
- ▶ Deleting the number

Response

200 OK

```
{
  "name": "User One",
  "id": 743,
  "number": "301",
  "orgUnitId": 523,
  "userId": 113,
  ...
  "publicAddrId": 733
  ...
  "scopeOuld": 519,
  ...
}
```

Check the modified address data with a

EXAMPLE CODE: CREATE HIDDEN PBX TOPSTOP

5. Create hidden PBX TopStop.

a. POST: Assign note and block conditions.

a. POST: Create the TopStop billing account for the PBX.

POST https://DOMAIN_REST/rest/billingAccounts

<pre>{ "orgUnitId": 519, "type": "charge", "resetTime": 1701385200000, "resetMonthly": true, "hidden": true, "flags": 18 }</pre>	<p>← PBX OrgUnit ID</p> <p>← Mandatory value.</p> <p>← Insert the current epoch date/time</p> <p>← Mandatory value.</p>
--	--

GET: The date of the billing account.

GET https://DOMAIN_REST/rest/billingAccounts/48

a. Response

200 OK

<pre>{ "value": 0.0, "id": 48, "type": "charge", "hidden": true, "units": null, "resetDaily": false, "resetMonthly": true, "resetTime": 1701385200000, "orgUnitId": 519 }</pre>	<p>← Remember the billing account ID for: ▸ Assigning note and block level</p>
---	--

Response

200 OK

```
{
  "value": 0.0,
  "id": 48,
  "type": "charge",
  "hidden": true,
  "units": null,
  "resetDaily": false,
  "resetMonthly": true,
  "resetTime": 1701385200000,
  "orgUnitId": 519
}
```


EXAMPLE CODE: CREATE HIDDEN PBX TOPSTOP

5. Create hidden PBX TopStop.

a. POST: Create the TopStop billing account for the PBX.

POST https://DOMAIN_REST/rest/billingAccounts

<pre>{ "orgUnitId": 519, "type": "charge", "resetTime": 1701385200000, "resetMonthly": true, "hidden": true, "flags": 18 }</pre>	<p>← PBX OrgUnit ID</p> <p>← Mandatory value.</p> <p>← Insert the current epoch date/time</p> <p>← Mandatory value.</p>
--	--

a. Response

200 OK

<pre>{ "value": 0.0, "id": 48, "type": "charge", "hidden": true, "units": null, "resetDaily": false, "resetMonthly": true, "resetTime": 1701385200000, "orgUnitId": 519 }</pre>	<p>← Remember the billing account ID for:</p> <ul style="list-style-type: none">▶ Assigning note and block limits <p>← Take the currency type from the assigned pricelist.</p>
---	--

b. POST: Assign the TopStop limit conditions.

POST https://DOMAIN_REST/rest/billingLimits

<pre>{ "value": 10.0, "enabled": true, "blocking": false, "billingAccountId": 48, "notifyMail": "noc@rest.com" }</pre>	<p>← For a blocking limit set the value to true.</p>
--	--

b. Response

200 OK

<pre>{ "value": 10.0, "id": 40, "enabled": true, "blocking": false, "disabled": false, "billingAccountId": 48, "notifyMail": "noc@rest.com", "notifyTime": 0 }</pre>	<p>← Remember the billing limit ID for:</p> <ul style="list-style-type: none">▶ Adjusting the value▶ Adjusting the email address▶ Blocking/de-blocking the limit
--	--

10 EXAMPLE: ASSIGN & PROVISION PHONES

EXAMPLE OVERVIEW: ASSIGN & PROVISION PHONES

► Goal:

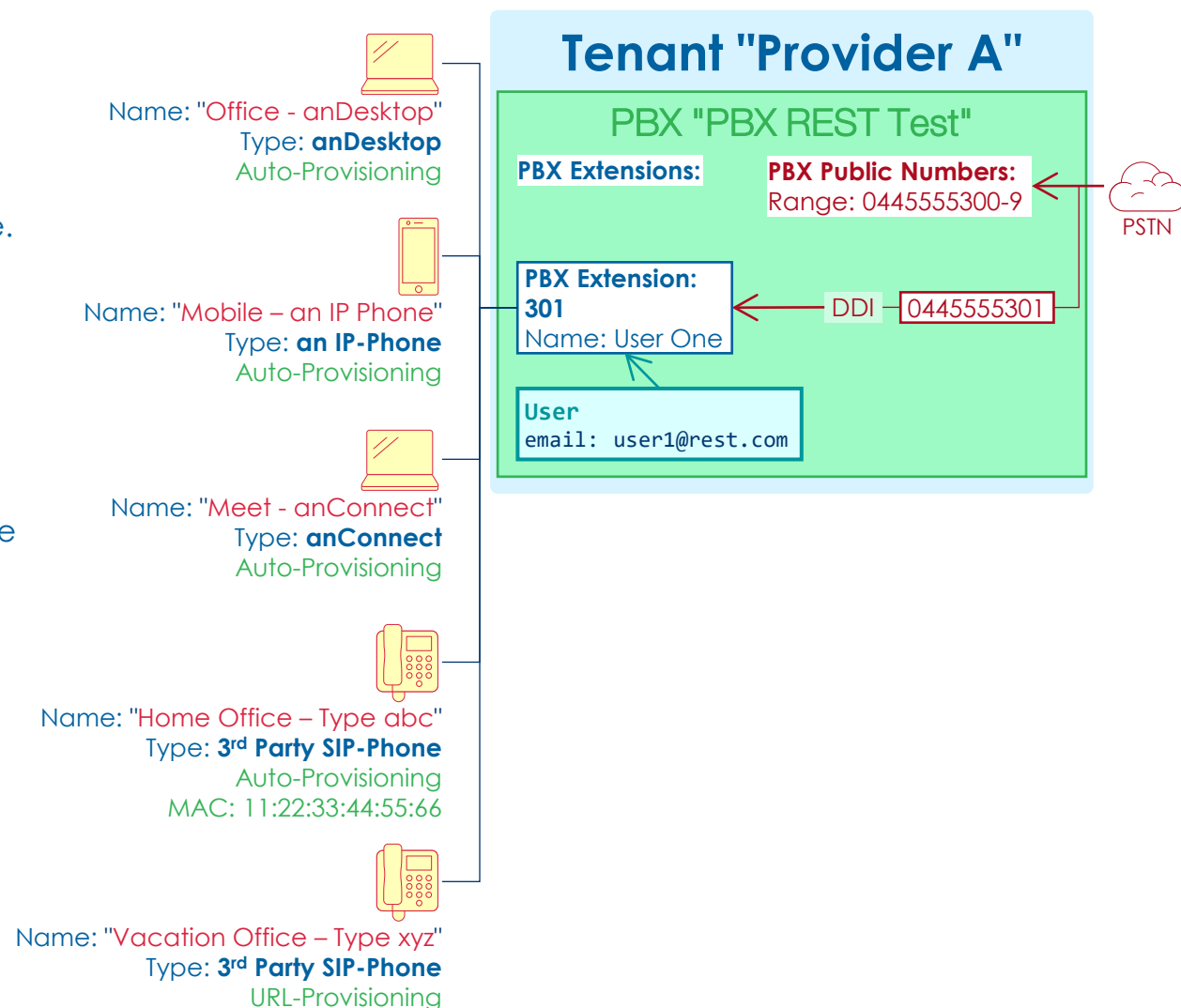
- Assign different phone types to a PBX Extension.

► Data to prepare:

- Values used in the example, see picture on the right side.
- ☐ Have ready the type of phone.
- ☐ Phone Name
- ☐ For 3rd party SIP-Phone provisioning have ready:
 - ☐ For the device configuration interface, e.g. Web administration.
 - ☐ Username for admin and eventually user
 - ☐ Password
- ☐ For auto-provisioning the MAC address of the phone device.
- ☐ ...

► Assumptions:

- On the anSwitch V7 system level the phone manufacturer redirection service is configured.
- On the system, tenant or PBX level the basic CTI URL is configured.
- On the tenant or PBX level an optional other name for the anDesktop is configured.
- The PBX Extension has assigned a user account.

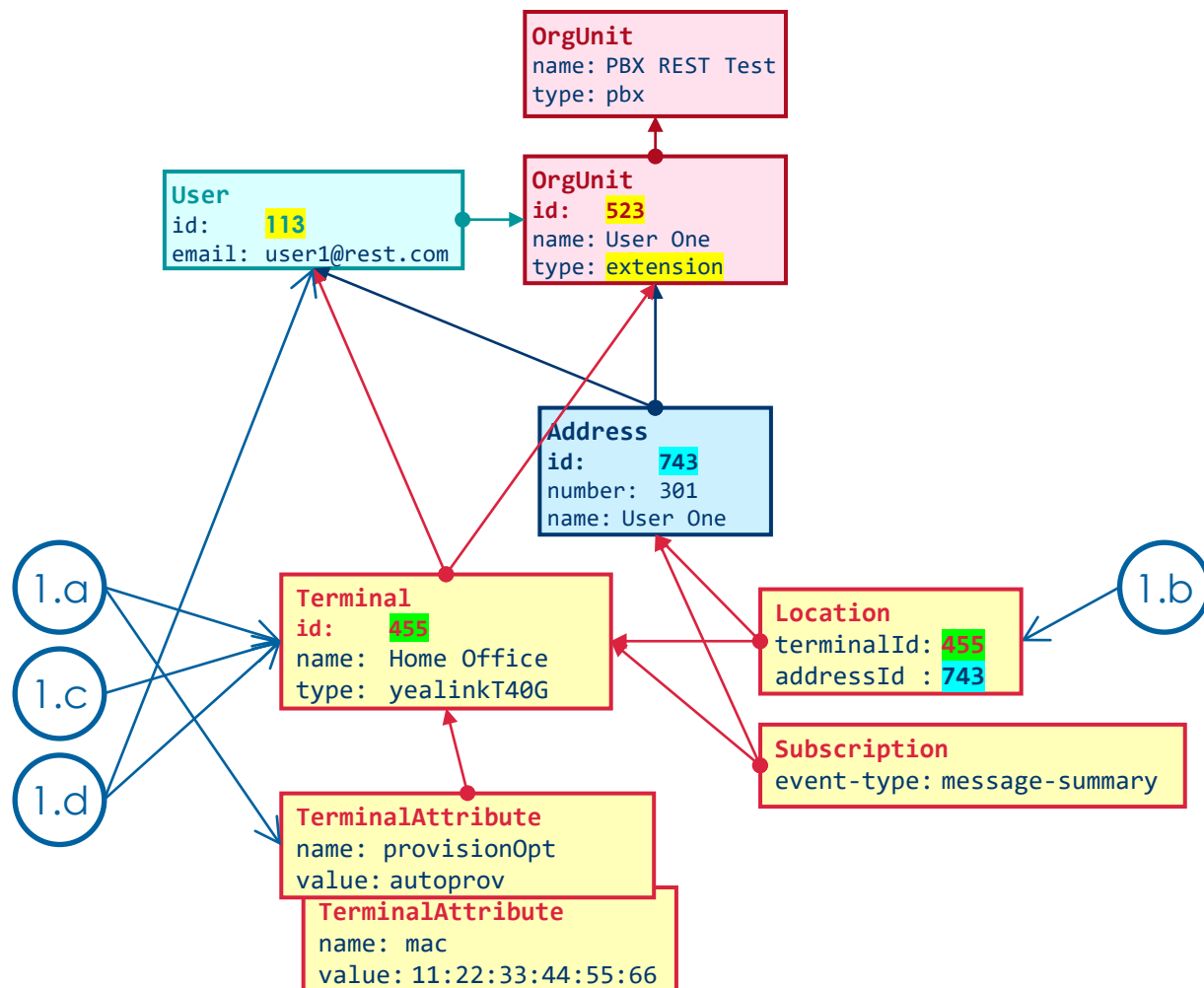


EXAMPLE OVERVIEW: ASSIGN & PROVISION PHONES

► Overview of the execution steps:

1. Assign a phone type to an address.
 - For each phone type that is assigned to a PBX Extension an own terminal must be created.
 - For each terminal, an own location must be created.
 - For each terminal type different types of terminal attributes must be created.
 - Examples for:
 - Terminal type: **anDesktop**
 - Terminal type: **an IP-Phone**
 - Terminal type: **anConnect**
 - Terminal type: **3rd Party SIP-Phone, Auto-Provisioning**
 - Terminal type: **3rd Party SIP-Phone, URL-Provisioning**
- a. POST: Create the terminal and its needed terminal attributes.
- b. POST: Create the terminal location.
- c. POST: Create message subscriptions.
2. Provision the phone.
 - a. PUT: Make ready the provisioning the phone.
 - b. PUT: Send an email to the user with installation instructions.

► Overview of involved DB objects



TERMINAL TYPE
"ANDESKTOP"

EXAMPLE CODE: TERMINAL TYPE "ANDESKTOP"

1. Assign a phone type "anDesktop" to an address.

a. POST: Create the terminal.

POST https://DOMAIN_REST/rest/terminals

```
{
  "name": "Office - anDesktop",
  "type": "cti",
  "orgUnitId": 523,
  "userId": 113,
  "password": "fsdfgsdfgf98ug00hzHJBS",
  "username": "453lkjnMN43"
}
```

← Mandatory value
 ← OrgUnit ID of the PBX Extension.
 ← **Always** create a **new** random password
 ← **Always** create a **new** random username

a. Response

200 OK

```
{
  "name": "Office - anDesktop",
  "id": 297,
  "type": "cti",
  "password": "crypt:OxsnNa_fTXqmO2OjjyNXKKF0qR",
  "username": "453lkjnMN43",
  "barred": false,
  "orgUnitId": 523,
  "userId": 113,
  "emergencyLocationId": 0,
  "sendAoc": false,
  "sendsMusicOnHold": false,
  "specialArrangement": false,
  "gateway": false,
  "noGui": false,
  "markedDelete": false
}
```

← Remember the terminal ID for:

- ▶ Terminal attributes
- ▶ Location
- ▶ Subscription

GET: Check the terminal data.

GET https://DOMAIN_REST/rest/terminals/297

variant:

GET https://DOMAIN_REST/rest/terminals?=&where=name.like('Office - anDesktop')

EXAMPLE CODE: TERMINAL TYPE "ANDESKTOP"

1. Assign a phone type "anDesktop" to an address.

a. POST: Create the needed terminal attributes

- ▶ Mandatory terminal attributes for this terminal type:

POST https://DOMAIN_REST/rest/terminalAttributes

```
{
  "terminalId": 297,
  "name": "provisionOpt",
  "value": "anDesktop"
}
```

← Mandatory value

```
{
  "terminalId": 297,
  "name": "oneTimeKey",
  "value": CtrbNZRvX4Q9QUzuLffy339db124
}
```

← Any random string

GET: Check all terminal attributes of this terminal

GET [https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq\(297\)](https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(297))

b. POST: Create the terminal location.

POST https://DOMAIN_REST/rest/locations

```
{
  "terminalId": 297,
  "addressId": 743
}
```

200 OK

```
{
  ...
  "id": 289,
  "terminalId": 297,
  "addressId": 743,
  ...
}
```

EXAMPLE CODE: TERMINAL TYPE "ANDESKTOP"

2. Provision the phone.

a. PUT: Make ready the provisioning the phone.

PUT https://DOMAIN_REST/rest/terminals/297.provision

```
{}
```

← The body must be empty!

c. Response

200 OK

```
{
  "provisioningOption": "anDesktop",
  "status": 200,
  "ctiDeviceID": "term.297",
  "windowsDownloadURL": "https://IP-ADDR/cti/app/update/belasAnDesktop-latest.exe",
  "macOSDownloadURL": "https://IP-ADDR/cti/app/update/belasAnDesktop-latest.pkg",
  "setupURL": "https://IP-ADDR/cti/app/redirect/ancall?data=ancall%3Asetup%...."
}
```

b. PUT: Send an email to the user with installation instructions.

PUT https://DOMAIN_REST/rest/terminals/297.mail

```
{}
```

← The body must be empty!

200 OK

No body

TERMINAL TYPE
"AN IP-PHONE"

EXAMPLE CODE: TERMINAL TYPE "AN IP-PHONE"

1. Assign a phone type "an IP-Phone" to an address.

a. POST: Create the terminal.

POST https://DOMAIN_REST/rest/terminals

```
{
  "name": "Mobile – an IP-Phone",
  "type": "anIpPhone",
  "orgUnitId": 523,
  "userId": 113,
  "password": "fsdfgsdfgf98ug00hzHJBS",
  "username": "fvg56AS056"
}
```

← Mandatory value
← OrgUnit ID of the PBX Extension.

← **Always** create a **new** random password
← **Always** create a **new** random username

a. Response

200 OK

```
{
  "name": "Mobile – an IP-Phone",
  "id": 113,
  "type": "anIpPhone",
  "password": "crypt:OxsnNa_fTXqmO2OjjyN",
  "username": "fvg56AS056",
  "barred": false,
  "orgUnitId": 523,
  "userId": 113,
  "emergencyLocationId": 0,
  "sendAoc": false,
  "sendsMusicOnHold": false,
  "specialArrangement": false,
  "gateway": false,
  "noGui": false,
  "markedDelete": false
}
```

← Remember the terminal ID for:

- ▶ Terminal attributes
- ▶ Location
- ▶ Subscription

GET: Check the terminal data.

GET https://DOMAIN_REST/rest/terminals/299

variant:

GET https://DOMAIN_REST/rest/terminals?=&where=name.like('Mobile – an IP-Phone')

EXAMPLE CODE: TERMINAL TYPE "AN IP-PHONE"

1. Assign a phone type "an IP-Phone" to an address.

a. POST: Create the needed terminal attributes

- ▶ Mandatory terminal attributes for this terminal type:

POST https://DOMAIN_REST/rest/terminalAttributes

```
{
  "terminalId": 308,
  "name": "provisionOpt",
  "value": "anipphone"
}
```

← Mandatory value

```
{
  "terminalId": 308,
  "name": "oneTimeCode",
  "value": "2970b736a5911789"
}
```

← Any random string

GET: Check data of all terminal attributes of this terminal

GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(308)

b. POST: Create the terminal location.

POST https://DOMAIN_REST/rest/locations

```
{
  "terminalId": 308,
  "addressId": 743
}
```

b. Response

200 OK

```
{
  ...
  "id": 300,
  "terminalId": 308,
  "addressId": 743,
  ...
}
```

EXAMPLE CODE: TERMINAL TYPE "AN IP-PHONE"

1. Assign a phone type "an IP-Phone" to an address.

c. POST: Create message subscriptions "message-summary".

POST https://DOMAIN_REST/rest/subscriptions

```
{
  "terminalId": 808,
  "eventType": "message-summary",
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301
}
```

← Mandatory value
← Terminal ID
← Privat phone number
← Terminal ID
← Privat phone number

POST: Create message subscriptions "as-feature-event".

POST https://DOMAIN_REST/rest/subscriptions

```
{
  "terminalId": 808,
  "eventType": "as-feature-event",
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301
}
```

← Mandatory value
← Terminal ID
← Privat phone number
← Terminal ID
← Privat phone number

GET: Check data of all subscriptions of this terminal

GET https://DOMAIN_REST/rest/subscriptions?=&where=terminalId.eq(808)

c. Response

200 OK

```
{
  "id": 928,
  "eventType": "message-summary",
  "terminalId": 455,
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301,
  ...
}
```

200 OK

```
{
  "id": 926,
  "eventType": "message-summary",
  "terminalId": 455,
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301,
  ...
}
```

EXAMPLE CODE: TERMINAL TYPE "AN IP-PHONE"

2. Provision the phone.

a. PUT: Make ready the provisioning the phone.

PUT https://DOMAIN_REST/rest/terminals/308.provision

{}

← The body must be empty!

c. Response

200 OK

```
{
  "provisioningOption": "an IP-Phone",
  "status": 200,
  "link": "csc:otc:2970b736a5911789@CLID_AS7POC2*",
  "accountID": "otc@CLID_ASV7",
  "password": "2970-b736-a591-1789"
}
```

b. PUT: Send an email to the user with installation instructions.

PUT https://DOMAIN_REST/rest/terminals/308.mail

{}

← The body must be empty!

200 OK

No body

TERMINAL TYPE
"ANCONNECT"

EXAMPLE CODE: TERMINAL TYPE "anConnect"

1. Assign a phone type "anConnect" to an address.

a. POST: Create the terminal.

POST https://DOMAIN_REST/rest/terminals

```
{
  "name": "Meet - anConnect",
  "type": "anConnect",
  "orgUnitId": 523,
  "userId": 113,
  "password": "rtzeZIIUZ7843sdfKL",
  "username": "gK4dg89345mwIP"
}
```

← Mandatory value
 ← OrgUnit ID of the PBX Extension.
 ← **Always** create a **new** random password
 ← **Always** create a **new** random username

a. Response

200 OK

```
{
  "name": "Meet - anConnect",
  "id": 310,
  "type": "anConnect",
  "password": "crypt:OxsnNa_fTXqmO2OjyN",
  "username": "gK4dg89345mwIP",
  "barred": false,
  "orgUnitId": 523,
  "userId": 113,
  "emergencyLocationId": 0,
  "sendAoc": false,
  "sendsMusicOnHold": false,
  "specialArrangement": false,
  "gateway": false,
  "noGui": false,
  "markedDelete": false
}
```

← Remember the terminal ID for:

- ▶ Terminal attributes
- ▶ Location
- ▶ Subscription

GET: Check the terminal data.

GET https://DOMAIN_REST/rest/terminals/310

variant:

GET [https://DOMAIN_REST/rest/terminals?=&where=name.like\(Meet - anConnect\)](https://DOMAIN_REST/rest/terminals?=&where=name.like(Meet - anConnect))

EXAMPLE CODE: TERMINAL TYPE "ANCONNECT"

1. Assign a phone type "anConnect" to an address.

- a. POST: Create the needed terminal attributes
- ▶ None

b. POST: Create the terminal location.

POST https://DOMAIN_REST/rest/locations

```
{  
  "terminalId": 309,  
  "addressId": 743  
}
```

b. Response

200 OK

```
{  
  ...  
  "id": 302,  
  "terminalId": 309,  
  "addressId": 743,  
  ...  
}
```


EXAMPLE CODE: TERMINAL TYPE "ANCONNECT"

1. Assign a phone type "anConnect" to an address.

c. POST: Create message subscription "dialog".

POST https://DOMAIN_REST/rest/subscriptions

```
{
  "terminalId": "318",
  "eventType": "dialog",
  "notifierId": "743",
  "notifierNumber": "301",
  "subscriberAddrId": "743",
  "subscriberNumber": "301"
}
```

← Mandatory value
← Terminal ID
← Privat phone number
← Terminal ID
← Privat phone number

c. Response

200 OK

```
{
  "id": "931",
  "eventType": "dialog",
  "terminalId": "455",
  "notifierId": "743",
  "notifierNumber": "301",
  "subscriberAddrId": "743",
  "subscriberNumber": "301",
  ...
}
```

GET: Check data of all subscriptions of this terminal

GET https://DOMAIN_REST/rest/subscriptions?=&where=terminalId.eq(318)

200 OK

```
{
  ...
}
```

EXAMPLE CODE: TERMINAL TYPE "ANCONNECT"

1. Assign a phone type "anConnect" to an address.

c. PUT: Make ready the provisioning the phone.

PUT https://DOMAIN_REST/rest/terminals/310

```
{  
  "type": "anConnect"  
}
```

→ Mandatory value

c. Response

200 OK

```
{  
  ...  
}
```

PUT https://DOMAIN_REST/rest/terminals/310.provision

```
{}
```

Not available yet!
← The body must be empty!

d. PUT: Send an email to the user with installation instructions.

PUT https://DOMAIN_REST/rest/terminals/310.mail

```
{}
```

← The body must be empty!

200 OK

No body

TERMINAL TYPE

"3RD PARTY SIP-
PHONE, AUTO-
PROVISIONING"

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

- ▶ Provisioning a "3rd Party SIP-Phone" via the REST API can be a tedious work, as every manufacturer has different parameters that need to be configured.
 - ▶ The type of parameters may also differ from phone type to phone type of the same manufacturer.
 - ▶ The different parameters are represented in the "Terminal Attributes".
- ▶ The following instructions are just a guideline.

Best Practice

1. Provision only 3rd party SIP-phone types that can be provisioned via the Portal UI
→ You can evaluate all needed terminal attributes and their values.
2. Assign via the Portal UI the desired phone type and read out all generated Terminal Attributes.
 - a) Give this phone a unique name that you can search via the REST API, e.g.: "test phone xyz".
 - b) Get the terminal ID and type of the phone "test phone xyz"
`GET https://DOMAIN_REST/rest/terminals?=&where=name.like('test phone xyz')&properties=id,type`
 - c) Get all terminal attributes that were generated for the phone "test phone ab"
`GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(320)`
 - d) Use the parameter "type" and **all** listed terminal attributes for your provisioning.

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

- ▶ For this REST API example a Yealink T40 G is chosen.
- ▶ Assign a phone Yealink T40 G with name "test phone Yealink T40G" to a PBX Extension the REST API has access rights to.

a. GET: Get the terminal type and ID of this test phone.

GET https://DOMAIN_REST/rest/terminals?=&where=name.like(test phone Yealink T40G')&properties=id,type

--	--

b. GET: Get the terminal attributes of this test phones

GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(320)

--	--

a. Response

200 OK

```
{
  "terminals": [
    {
      "id": 320,
      "type": "yealinkT40G"
    }
  ]
}
```

← Remember this type for:
▶ The phone type creation via REST API.

b. Response

200 OK

```
{
  "terminalAttributes": ...
}
```

The whole list of terminal attributes are returned for this Yealink T40G with auto-provisioning.

Example: Yealink T40G	Terminal Attribute Name	Value	Remark
Mandatory terminal attributes for Auto-Provisioning	provisionOpt	"autoprov"	Provisioning type
	macAuth	"true"	Mandatory due to the provisioning type
	mac	"11:22:33:44:55:66"	MAC address of the device
Terminal attributes needed for the correct working of this phone type.	webAdminname	"admin"	Admin username
	webAdminPassword	"crypt:ejlHpabqU7"	Encrypted admin password
	webUsername	"user"	User username
Note: The names and values can change according the selected phone type.	fkey1	"line/743//301 User One"	Attributes with instructions for configuring the first configurable function key of the phone.
	attrDisplayFirstPageOnly	"false"	
	attrPhoneFirstPageOnly	"false"	

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

1. Assign a phone type "3rd Party SIP-Phone, Auto-Provisioning" to an address.

- a. POST: Create the terminal Yealink T40G.

POST https://DOMAIN_REST/rest/terminals

```
{
  "name": "Home Office – Yealink T40G",
  "type": "yealinkT40G",
  "orgUnitId": 523,
  "userId": 113,
  "password": "fsdfgsdfgf45rvbug00hzHJBS",
  "username": "abh56AS056"
}
```

← Mandatory value
 ← OrgUnit ID of the PBX Extension.
 ← **Always** create a **new** random password
 ← **Always** create a **new** random username

- a. Response

200 OK

```
{
  "name": "Home Office – Yealink T40G",
  "id": 321,
  "type": "yealinkT40G",
  "password": "crypt:ejlHpabqU7PPcvfqAob_Jt",
  "username": "abh56AS056",
  "barred": false,
  "orgUnitId": 523,
  "userId": 113,
  "emergencyLocationId": 0,
  "sendAoc": false,
  "sendsMusicOnHold": false,
  "specialArrangement": false,
  "gateway": false,
  "noGui": false,
  "markedDelete": false
}
```

- ← Remember the terminal ID for:
- ▶ Terminal attributes
 - ▶ Location
 - ▶ Subscription

- b. GET: Check terminal data

GET https://DOMAIN_REST/rest/terminals/321

variant:

GET https://DOMAIN_REST/rest/terminals?=&where=name.like(Home Office – Yealink T40G')

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

1. Assign a phone type "3rd Party SIP-Phone, Auto-Provisioning" to an address.

- b. POST: Create the mandatory terminal attributes for auto-provisioning.

POST https://DOMAIN_REST/rest/terminalAttributes

{ "terminalId": 321, "name": "provisionOpt", "value": "autoprov" }	← Mandatory value!
{ "terminalId": 321, "name": "macAuth", "value": "true" }	← Mandatory value!
{ "terminalId": 321, "name": "mac", "value": "66:55:44:33:22:11" }	← Insert the MAC address of the device

GET: Check data of all terminal attributes of this terminal.

GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(321)

- b. POST: Create the terminal attributes for the **correct working** of this phone type

POST https://DOMAIN_REST/rest/terminalAttributes

{ "terminalId": 325, "name": "webAdminname", "value": "admin" }	← Mandatory value!
{ "terminalId": 325, "name": "webAdminPassword", "value": "mySecretPw" }	← Insert a secure password!
{ "terminalId": 325, "name": "webUsername", "value": "user" }	← Mandatory value!
{ "terminalId": 325, "name": "fkey1", "value": "line/743//301 User One" }	← The value composes: "line/<ADDRESS_ID>//<PRIVATE_NUMBER> <EXTENSION NAME>"
{ "terminalId": 325, "name": "attrDisplayFirstPageOnly", "value": "false" }	
{ "terminalId": 325, "name": "attrPhoneFirstPageOnly", "value": "false" }	

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

1. Assign a phone type "3rd Party SIP-Phone, Auto-Provisioning" to an address.

c. POST: Create the terminal location.

POST https://DOMAIN_REST/rest/locations

```
{  
  "terminalId": 321,  
  "addressId": 743  
}
```

c. Response

200 OK

```
{  
  ...  
  "id": 314,  
  "terminalId": 321,  
  "addressId": 743,  
  ...  
}
```


EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

1. Assign a phone type "3rd Party SIP-Phone, Auto-Provisioning" to an address.

d. POST: Create message subscriptions "message-summary".

POST https://DOMAIN_REST/rest/subscriptions

```
{
  "terminalId": 321,
  "eventType": "message-summary",
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301
}
```

← Mandatory value
 ← Terminal ID
 ← Privat phone number
 ← Terminal ID
 ← Privat phone number

POST: Create message subscriptions "as-feature-event".

POST https://DOMAIN_REST/rest/subscriptions

```
{
  "terminalId": 321,
  "eventType": "as-feature-event",
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301
}
```

← Mandatory value
 ← Terminal ID
 ← Privat phone number
 ← Terminal ID
 ← Privat phone number

GET: Check data of all subscriptions of this terminal

GET https://DOMAIN_REST/rest/subscriptions?=&where=terminalId.eq(321)

d. Response

200 OK

```
{
  "id": 944,
  "eventType": "message-summary",
  "terminalId": 321,
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301,
  ...
}
```

200 OK

```
{
  "id": 945,
  "eventType": "message-summary",
  "terminalId": 321,
  "notifierId": 743,
  "notifierNumber": 301,
  "subscriberAddrId": 743,
  "subscriberNumber": 301,
  ...
}
```

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, AUTO-PROVISIONING"

2. Provision the phone.

a. PUT: Make ready the provisioning the phone.

PUT https://DOMAIN_REST/rest/terminals/321.provision

{}

← The body must be empty!

c. Response

200 OK

```
{
  "provisioningOption": "an IP-Phone",
  "status": 200,
  "link": "csc:otc:2970b736a5911789@CLID_AS7POC2*",
  "accountID": "otc@CLID_ASV7",
  "password": "2970-b736-a591-1789"
}
```

b. PUT: Send an email to the user with installation instructions.

PUT https://DOMAIN_REST/rest/terminals/321.mail

{}

← The body must be empty!

200 OK

No body

TERMINAL TYPE
"3RD PARTY SIP-
PHONE, URL-
PROVISIONING"

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, URL-PROVISIONING"

- ▶ Provisioning a "3rd Party SIP-Phone" via the REST API can be a tedious work, as every manufacturer has different parameters that need to be configured.
 - ▶ The type of parameters may also differ from phone type to phone type of the same manufacturer.
 - ▶ The different parameters are represented in the "Terminal Attributes".
- ▶ The following instructions are just a guideline.

Best Practice

1. Provision only 3rd party SIP-phone types that can be provisioned via the Portal UI
→ You can evaluate all needed terminal attributes and their values.
2. Assign via the Portal UI the desired phone type and read out all generated Terminal Attributes.
 - a) Give this phone a unique name that you can search via the REST API, e.g.: "test phone xyz".
 - b) Get the terminal ID and type of the phone "test phone abc"
`GET https://DOMAIN_REST/rest/terminals?=&where=name.like('test phone abc')&properties=id,type`
 - c) Get all terminal attributes that were generated for the phone "test phone ab"
`GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(321)`
 - d) Use the parameter "type" and **all** listed terminal attributes for your provisioning.

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, URL-PROVISIONING"

- ▶ Provisioning a "3rd Party SIP-Phone" via "URL-Provision" is almost identical to the "Auto-Provisioning".
 - ▶ The only difference lies in the provisioning type terminal attributes.
- This example shows just the differences in the terminal attributes.

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, URL-PROVISIONING"

- ▶ For this REST API example a Yealink T40 G is chosen.
- ▶ Assign a phone Yealink T40 G with name "test phone Yealink T40G" to a PBX Extension the REST API has access rights to.

a. GET: Get the terminal type and ID of this test phone.

GET https://DOMAIN_REST/rest/terminals?=&where=name.like(test phone Yealink T40G')&properties=id,type

--	--

b. GET: Get the terminal attributes of this test phones

GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(322)

--	--

a. Response

200 OK

```
{
  "terminals": [
    {
      "id": 322,
      "type": "yealinkT40G"
    }
  ]
}
```

← Remember this type for:
▶ The phone type creation via REST API.

b. Response

200 OK

```
{
  "terminalAttributes": ...
}
```

The whole list of terminal attributes are returned for this Yealink T40G with URL-provisioning.

Example: Yealink T40G	Terminal Attribute Name	Value	Remark
Mandatory terminal attributes for URL-Provisioning	provisionOpt oneTimeCode	"manual" "e62a44f54b95cc14"	Provisioning type Last created one time code
Terminal attributes needed for the correct working of this phone type.	webAdminname webAdminPassword webUsername	"admin" "crypt:ejlHpabqU7" "user"	Admin username Encrypted admin password User username
Note: The names and values can change according the selected phone type.	fkey1 attrDisplayFirstPageOnly attrPhoneFirstPageOnly	"line/743//301 User One" "false" "false"	Attributes with instructions for configuring the first configurable function key of the phone.

EXAMPLE CODE: TERMINAL TYPE "3RD PARTY SIP-PHONE, URL-PROVISIONING"

1. Assign a phone type "3rd Party SIP-Phone, URL" to an address.

a. POST: Create the terminal Yealink T40G.
→ Equal as with auto-provisioning

b. POST: Create the mandatory terminal attributes for URL-provisioning.

POST https://DOMAIN_REST/rest/terminalAttributes

{ "terminalId": 325, "name": "provisionOpt", "value": "manual" }	← Mandatory value!
{ "terminalId": 325, "name": "oneTimeCode", "value": "e62a44f54b95cc14" }	← Requirement of the value: <ul style="list-style-type: none"> ▶ Minimum length: 16 char ▶ Random string composed of: <ul style="list-style-type: none"> ▶ Small letters: [a-z] ▶ Digits: [0-9]

GET: Check data of all terminal attributes of this terminal.

GET https://DOMAIN_REST/rest/terminalAttributes?=&where=terminalId.eq(325)

1 c. – d. & 2. a. - .b. These steps are equal to auto-provisioning.

b. POST: Create the terminal attributes for the **correct working** of this phone type

POST https://DOMAIN_REST/rest/terminalAttributes

{ "terminalId": 325, "name": "webAdminname", "value": "admin" }	← Mandatory value!
{ "terminalId": 325, "name": "webAdminPassword", "value": "mySecretPw" }	← Insert a secure password!
{ "terminalId": 325, "name": "webUsername", "value": "user" }	← Mandatory value!
{ "terminalId": 325, "name": "fkey1", "value": "line/743//301 User One" }	← The value composes: "line/<ADDRESS_ID>//<PRIVATE_NUMBER> <EXTENSION NAME>"
{ "terminalId": 325, "name": "attrDisplayFirstPageOnly", "value": "false" }	
{ "terminalId": 325, "name": "attrPhoneFirstPageOnly", "value": "false" }	

11 EXAMPLE: DELETE A PBX OR PBX EXTENSION

EXAMPLE CODE: DELETE A PBX OR PBX EXTENSION

- ▶ Delete a PBX or PBX Extension.
 - ▶ **No** REST request exists that deletes a PBX or PBX Extension with all its associated instances of the available objects.
 - ▶ Deleting just the OrgUnit of a PBX leaves a lot of garbage in the database DB.
 - ▶ Unwanted side effects of garbage in the DB database cannot be ruled out.
 - ▶ Finding all associated instances and deleting them one by one needs a lot of code in a customer's CRM application.

Best Practice

Delete a PBX or PBX Extension via the Portal UI.

The screenshot displays two sections of the Portal UI. The top section, titled 'PBX List', features a search bar with 'Name: test' and a table with columns: Name, Description, Member of, and E-Mail. A single row is visible: 'PBX REST Test' with description 'PBX of company REST', member 'Provider A', and email 'jm@rest.com'. The bottom section, titled 'Extensions: PBX REST Test', includes a 'Dashboard' link and a search bar with 'Extension: 301'. It contains a table with columns: Extension, Displayed Name, Department, Call Distribution Mode, PBX User, Dial In Number of, Displayed Public Number, Voicemail Status, and Registration Status. A single row is visible: '301' with name 'User One', department, 'All Phones', user 'User One', number '0445555301', public number '0445555305', and both status indicators are green.

Name	Description	Member of	E-Mail
PBX REST Test	PBX of company REST	Provider A	jm@rest.com

Extension	Displayed Name	Department	Call Distribution Mode	PBX User	Dial In Number of	Displayed Public Number	Voicemail Status	Registration Status
301	User One		All Phones	User One	0445555301	0445555305	●	●

12 EXAMPLE: USEFUL PBX & TERMINAL COMMANDS

EXAMPLE CODE: FORCE A PHONE TO RE-DOWNLOAD ITS CONFIG

► Force a phone to re-download its configuration

- A re-download may become necessary for a set of terminals, e.g. its phone template changed.
- Proposed process:
 1. Send a GET request with the desired query to obtain the list of terminals matching.
 2. Send the notify request for each of them one by one.

a. PUT: Send a notification to the phone to re-load its

PUT https://DOMAIN_REST/rest/terminals/310.notify

{}

← The body must be empty!

a. Response

200 OK

No body

Check if the device reloads its configuration

EXAMPLE CODE: LIST ALL TERMINALS OF A PBX EXTENSION

- ▶ Check all terminals assigned to a PBX Extension OrgUnit.

- ▶ GET: List terminals of a PBX

GET [https://DOMAIN_REST/rest/terminals?=&where=orgUnitId.eq\(523\)](https://DOMAIN_REST/rest/terminals?=&where=orgUnitId.eq(523))

Response

200 OK

```
{
  "terminals": [
    {
      "name": "Mobile – an IP-Phone",
      ...
    },
    {
      "name": ""Vacation Office – GRANDSTREAM GRP2612",
      ...
    }
  ]
}
```

EXAMPLE CODE: SET THE TIME ZONE OF A PHONE

► Set the Time Zone of a Phone.

- The default time zone is defined by the PBX.
- If a phone needs a different time zone, then a terminal attribute "timezone" must be present or created.

► POST: Create a time zone terminal attribute

POST https://DOMAIN_REST/rest/terminalAttributes

```
{
  "terminalId": 321,
  "name": "timezone",
  "value": "timezone01"
}
```

Response

200 OK

```
{
  "name": "timezone",
  "value": "timezone01",
  "id": 1182,
  "terminalId": 321
}
```

► PUT: Assign a different time zone

PUT https://DOMAIN_REST/rest/terminalAttributes/1182

```
{
  "value": "timezone07"
}
```

Response


200 OK

No body

EXAMPLE CODE: SET THE RINGING TONES OF A PHONE

► Set the ringing tones of a Phone.

- The default time zone is defined by the anSwitch V7 system.
- If a phone needs a different ringing tones , then a terminal attribute "alertInternal" or "alertExternal" must be present or created.
- Check via Portal UI page "Phone related features" if the phone type supports the setting of ringing tones.



Phone Notification	
Source	Notification
Internal Call	Notification 1
External Call	

► POST: Create the ringing tone terminal attributes

POST https://DOMAIN_REST/rest/terminalAttributes

```
{
  "terminalId": "321",
  "name": "alertInternal",
  "value": "1"
}
```

```
{
  "terminalId": "321",
  "name": "alertExternal",
  "value": "1"
}
```

Response

200 OK

```
{
  "name": "alertInternal",
  "value": "1",
  "id": "2081",
  "terminalId": "321"
}
```

```
{
  "name": "alertExternal",
  "value": "1",
  "id": "2082",
  "terminalId": "321"
}
```

► PUT: Assign a different internal ringing tone

PUT https://DOMAIN_REST/rest/terminalAttributes/2081

```
{
  "value": "7"
}
```

Response

200 OK

No body

EXAMPLE CODE: GET THE VOICE MAIL MESSAGES

► Get the Voice Mail Messages of a PBX Extension.

- The VoiceMail messages are associated to the Address ID of the PBX Extension.

a. GET: Get all Voice Mail Messages of a PBX Extension

```
GET https://DOMAIN_REST/rest/ ↗
      audioFiles?where=addressId.eq(743).and(type.like('message'))
```

No body

b. GET: The audio file of a VoiceMail message

```
GET https://DOMAIN/rest/audioFiles/134.wav
```

No body

a. Response

200 OK

```
{
  "audioFiles": [
    {
      "number": "301",
      "time": 1654692548289,
      "duration": 4760,
      "name": "User One",
      "id": 134,
      "type": "message",
      "addressId": 743,
      "newMessage": true
    },
    ... Further audio file IDs
  ]
}
```

← Message ID of the audio file to download.

b. Response

200 OK

Binary code

The body of the 200 OK response contains the audio-stream as binary code that must be saved to a file with extension *.WAV.

EXAMPLE CODE: GET A CALL RECORDING ORDERED VIA CSTA

- ▶ A CTI application orders a call recording via the CSTA protocol and download the recorded file via REST API.

- A CTI application orders a call recording with the CSTA method "recordMessage".
- In the response of the anSwitch V7 CTI server the message ID of the recorded audio file is contained, e.g.

```
1 <RecordMessage>
2   <callToBeRecorded>
3     <callID>xxx</callID>
4     <deviceID>dddd</deviceID>
5   </callToBeRecorded>
6   <messageID>123456</messageID>
7 </RecordMessage>
```

- GET: When the call is finished the CTI application can download the recorded audio file.

```
GET https://DOMAIN_REST/rest/ ↗
    audioFiles?where=addressId.eq(123456)
```

No body

c. Response

200 OK

Binary code

The body of the 200 OK response contains the audio-stream as binary code that must be saved to a file with extension *.WAV.

EXAMPLE CODE: UPLOAD MUSIC ON HOLD FOR A PBX

► Upload Music on Hold MoH audio file for a PBX.

- The default MoH is defined by the anSwitch V7 system.

a. POST: Create a new audio-file instance.

POST https://DOMAIN_REST/rest/audioFiles

```
{
  "type": "static",
  "addressId": 0
}
```

b. PUT: Upload the MoH audio file.

PUT https://DOMAIN_REST/rest/audioFiles/247.wav

Set the headers:

- content-type: audio/wave
- content-length: 1234567890 → size of the audio file.

c. POST: Create a MoH OrgUnit attribute for the PBX.

POST https://DOMAIN_REST/rest/orgUnitAttributes

```
{
  "orgUnitId": 519,
  "name": "musicOnHoldId",
  "value": "247"
}
```

- If a PBX needs a different MoH, then a OrgUnit attribute "musicOnHoldId" must be present or created for the PBX.

a. Response

200 OK

```
{
  "id": 247,
  "type": "static",
  "addressId": 0,
  ...
}
```

← This is the audio file ID.

- It must be contained in the uploaded audio file name.
- It is the reference for the PBX MoH audio file.

b. Response

200 OK

No body

c. Response

200 OK

```
{
  "name": "musicOnHoldId",
  "value": "139",
  "id": 778,
  "orgUnitId": 519
}
```

EXAMPLE CODE: MANAGE CALL FORWARD CF

► Manage Call Forwards CF.

- Every call forward of an PBX Extension is configured in an own instance of object "Forwards" (like the attributes of a OrgUnit or terminal).

a. GET: Get the instance ID of private number 301

GET https://DOMAIN_REST/rest/addresses?where=number.like('301').and(orgUnitId.eq(523))&properties=id

No body

b. GET: List all Call Forwards CF of private number 301

GET https://DOMAIN_REST/rest/forwards?where=addressId.eq(743)

No body

c. POST: Set new active CFU to 0876543219 for private number 301

POST https://DOMAIN_REST/rest/forwards

```
{
  "addressId": 743,
  "type": "cfu",
  "destination": "0876543219",
  "delay": 0,
  "inactive": false
}
```

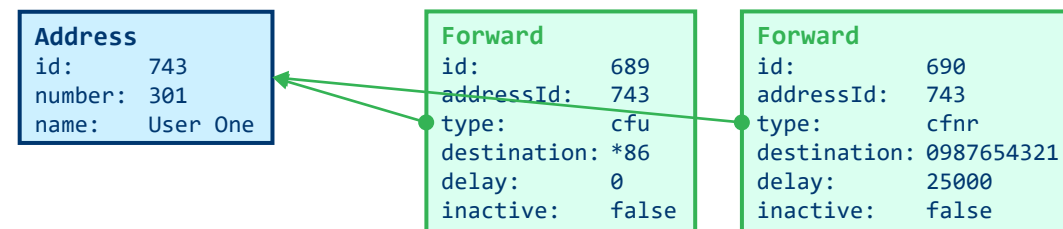
- ← Select the desired CF type
- ← Any dialable number (*86 for the own VoiceMail Box)
- ← For CFNR define the delay in milliseconds.
- ← Activate the CF

d. PUT: Inactivate CFU to 0876543219 for private number 301

PUT https://DOMAIN_REST/rest/forwards/743

```
{ "inactive": true }
```

- ← Deactivate the CF



a. Response

200 OK

```
{
  "id": 743
}
```

- ← Remember for configuring the call forwards.

b. Response

200 OK

Empty response or list of already configured CFs.

- Check if the desired CF is already configured:
 - If not, then the CF instances must be created.
 - If yes, then remember its instance-id.

c. Response

200 OK

```
{
  "id": 692,
  "type": "cfu",
  "destination": "0876543219",
  "delay": 0,
  "inactive": false,
  "timetableId": 0,
  "addressId": 743,
}
```

- ← Remember for configuring the call forwards.

d. Response

200 OK

```
{ ..., "inactive": true, ... }
```

13 EXAMPLE: MANAGE CONTACTS

EXAMPLE OVERVIEW: MANAGE CONTACTS

► Goal:

- Manage the contacts of a PBX

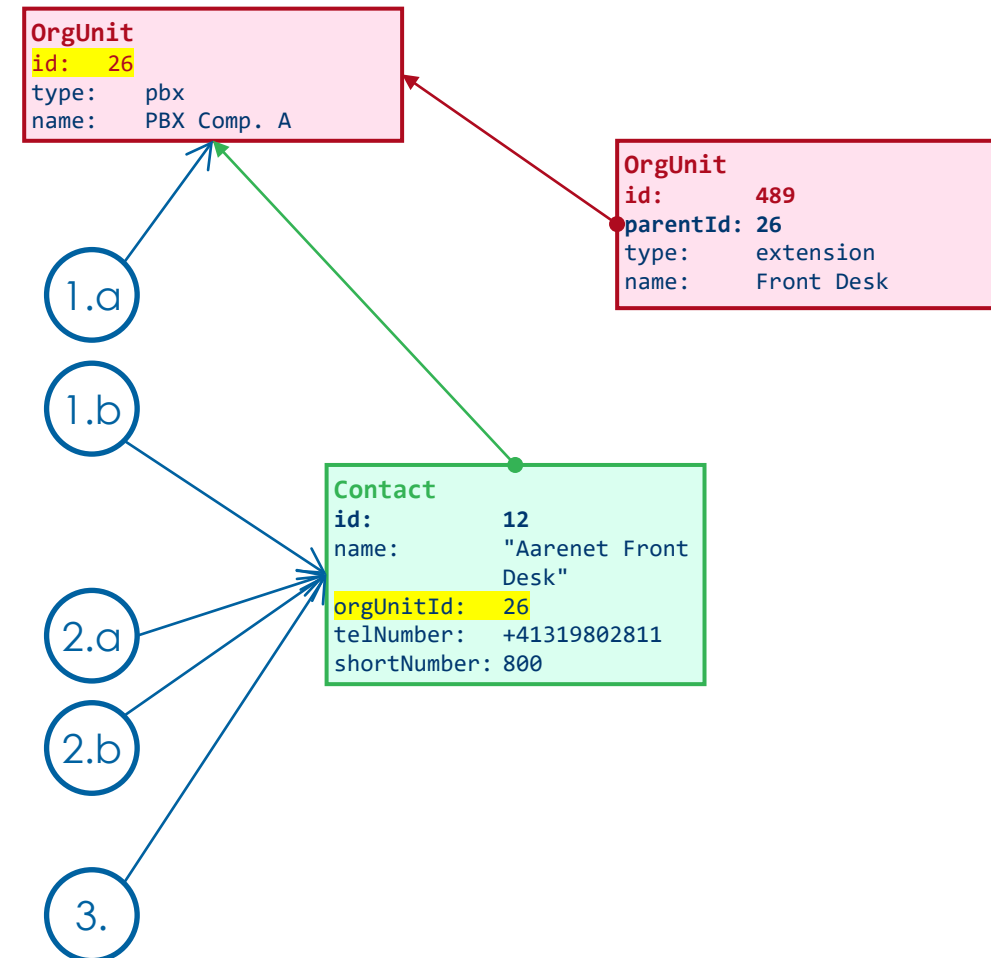
► Data to prepare:

- ☐ Name of the of the PBX
- ☐ Name of the PBX extension
- ☐ Data of new contact
 - ☐ Name
 - ☐ Number
 - ☐ Short number

► Overview of the execution steps:

1. Get the last 5 CDRs of "Front Desk"
 - a. GET: Get the OrgUnit ID of extension with name "PA-PBX-0AX89001".
 - b. POST: Create new contact with short number "Aarenet Front Desk".
2. Manage contacts
 - a. PUT: Change a property of a contact, e.g. number
 - b. DELETE: Delete a contact
3. Search contacts

► Overview of involved DB objects



EXAMPLE CODE: CREATE A CONTACT FOR A PBX

1. Create a Contact for a PBX.

a. GET: Get the OrgUnit ID of PBX with name "PA-PBX-0AX89001".

GET https://DOMAIN_REST/rest/orgUnits?where=name.like('PA-PBX-0AX89001')&properties=id

No body

a. Response

200 OK

```
{
  "orgUnits": [
    {
      "id": 26
    }
  ]
}
```

← OrgUnit ID of the PBX, save for later use!

b. Get: Check if contact "Aarenet Front Desk" already exists for

GET https://DOMAIN_REST/rest/contacts?where=name.like('Aarenet Front Desk').and(orgUnitId.eq(26))

No body

b. Response

200 OK

```
{
  "contacts": []
}
```

→ Empty brackets [] → this contact doesn't exist.

c. POST: Create new contact with short number "Aarenet Front Desk".

POST https://DOMAIN_REST/rest/contacts

```
{
  "orgUnitId": 26,
  "name": "Aarenet Front Desk",
  "telNumber": "+41 31 980 28 11",
  "shortNumber": "800"
}
```

→ + is not a valid character
→ Use 00 instead or the valid international prefix.

c. Response

200 OK

```
{
  "name": "Aarenet Front Desk",
  "type": null,
  "orgUnitId": 26,
  "telNumber": "+41 31 980 28 11",
  "telNumberNormalized": "41319802811",
  "shortNumber": "800"
}
```

← New contact ID

→ The normalized number will be dialed when the short number is dialed.
If a "+" was configured it is stripped and the destination cannot be found.

EXAMPLE CODE: CHANGE, DELETE OR SEARCH CONTACTS

2. Manage a contact

a. PUT: Change a property of a contact, e.g. number

PUT https://DOMAIN_REST/rest/ contacts/12	
<pre>{ "telNumber": "0041319802811", }</pre>	

GET https://DOMAIN_REST/rest/ contacts/12	
No body	

b. DELETE: Delete a contact

DELETE https://DOMAIN_REST/rest/ contacts/12	
No body	

a. Response

200 OK	
	There is no return of the changed instance. → Re-read the instance for checking the new value.

b. Response

200 OK	
	There is no return of the deleted instance. → Re-read the instance for ching if it is deleted.

3. Search contacts

a. Get: Search the contacts for names that start with "Bo".

GET https://DOMAIN_REST/rest/ contacts?where=name.like('Bo%').and(orgUnitId.eq(26))	
No body	

b. Get: Search the contacts for names that start with "Bo" and the phone number starts with "079".

GET https://DOMAIN_REST/rest/ contacts?where=name.like('Bo%').and(telNumber.like('079%')).and(orgUnitId.eq(26))	
No body	

14 EXAMPLE: LIST CDR OF A PBX EXTENSION

EXAMPLE OVERVIEW: LIST CDRS OF A PBX EXTENSION

► Goal:

► Get the last 5 CDRs of user "Front Desk"

► Data per call to be provided:

- Id of the CDR
- Destination number
- Connection date/time
- Call release date/time
- Charge

► Data to prepare:

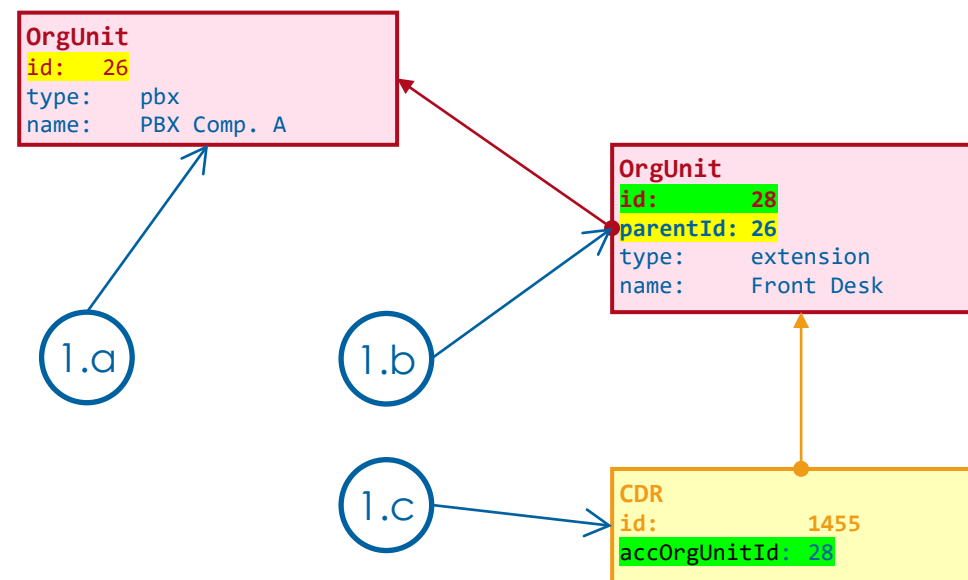
- ☐ Name of the of the PBX
- ☐ Name of the PBX extension

► Overview of the execution steps:

1. Get the last 5 CDRs of "Front Desk"

- GET: Get the OrgUnit ID of extension with name "PA-PBX-0AX89001".
- GET: Get the OrgUnit ID of extension with name "Front Desk".
- GET: Get 5 CDRs of the PBX extension "Front Desk".

► Overview of involved DB objects



EXAMPLE CODE: LIST CDRS OF A PBX EXTENSION

1. Get the last 5 CDRs of "Front Desk"

a. GET: Get the OrgUnit ID of PBX with name "PA-PBX-0AX89001".

```
GET https://DOMAIN_REST/rest/orgUnits?=&where=name.like('PA-PBX-0AX89001')
```

No body

a. Response

200 OK

```
{
  "orgUnits": [
    {
      "name": "PA-PBX-0AX89001",
      "id": 26,
      "type": "pbx",
      "description": "Yupi Inc",
      "parentId": 25
    }
  ]
}
```

← OrgUnit ID of the PBX, save for later use!

b. GET: Get the OrgUnit ID of extension with name "Front Desk" of PBX "PA-PBX-0AX89001".

```
GET https://DOMAIN_REST/rest/orgUnits?=&where=name.like('Front Desk').and(parentId.eq(26))
```

No body

b. Response

200 OK

```
{
  "orgUnits": [
    {
      "name": "Front Desk",
      "id": 28,
      "type": "extension",
      "description": "",
      "parentId": 26
    }
  ]
}
```

← OrgUnit ID of the PBX extension, save for later use!

EXAMPLE CODE: LIST CDRS OF A PBX EXTENSION

1. Get the last 5 CDRs of "Front Desk"

c. GET: Get 5 CDRs of the PBX extension "Front Desk"

GET

`https://DOMAIN_REST/rest/cdrs?where=accOrgUnitId.eq(28)&limit=5&ascending=id&properties=id,destNumber,timeConnect,timeEnd,chargePublic`

No body

Note

For details about the available CDR fields, see the training documentation:

"Rating & Call Detail Record CDR"
(doc id: training_as7_706_sys_rating_cdr)

Note

Do not use this type of obtaining CDRs for billing purposes!

→ Large queries may block the database!
→ Use the CSV formatted CDR files for this task.

c. Response

200 OK

```
{
  "cdrs": [
    {
      "id": 2420,
      "destNumber": "330",
      "timeConnect": 1692364701534,
      "timeEnd": 1692364704230,
      "chargePublic": 0.0
    },
    {
      "id": 2427,
      "destNumber": "0123456789",
      "timeConnect": 1694436798192,
      "timeEnd": 1694436813831,
      "chargePublic": 103.007845565
    },
    {
      "id": 2429,
      "destNumber": "0123456789",
      "timeConnect": 1694436900075,
      "timeEnd": 1694436913697,
      "chargePublic": 103.00683370333333
    },
    {
      "id": 2430,
      "destNumber": "0123456789",
      "timeConnect": 1694437569320,
      "timeEnd": 1694437591917,
      "chargePublic": 103.01133616166666
    },
    {
      "id": 2432,
      "destNumber": "300",
      "timeConnect": 1695044262452,
      "timeEnd": 1695044269466,
      "chargePublic": 0.0
    }
  ]
}
```

EXAMPLE CODE: OTHER CDR COLLECTING COMMANDS

- ▶ Get the CDRs of multiple OrgUnits.

```
GET https://DOMAIN_REST/rest/cdrs?where=accOrgUnitId.in(28,32)
```

No body

- ▶ Get all CDR's of PBX from last month.
 - ▶ Transform the start and end dates to UNIX timestamp in milliseconds.
(<https://www.unixtimestamp.com/>)
 - ▶ timeStart 1.10.2023 00:00 → Timestamp: 1696111200'000
 - ▶ timeEnd 1.11.2023 00:00 → Timestamp: 1698793200000

```
GET https://DOMAIN_REST/rest/cdrs?where=accOrgUnitId.eq(28).and(timeStart.ge(1696111200000)).and(timeEnd.le(1698793200000))
```

No body

LAST PAGE

Date	Doc-ID	Description	Changes
8.12.2022	application_information_as7_api_rest_e2.4	Preliminary published	
28.12.2023	application_information_as7_api_rest_e30	V7.14.0; Sections and examples have been revised and supplemented.	Complete overdone