FreeACS FreeACS Web Services User Manual

2014R1

Table of Contents

1 Document Introduction

1.1 Document Purpose

The purpose of the document is to explain the FreeACS Web Services interface.

1.2 Document Audience

Developers of FreeACS Web Service Clients and FreeACS Administrators.

1.3 Document History

Version	Editor	Date	Changes
1.0.5	M. Simonsen	23-Mar-09	Initial public version.
1.0.6	M. Simonsen	29-Sep-09	Revised edition, added info about SSL connection + password
1.0.7	M. Simonsen	01-Oct-09	Revised edition
1.1.0	M. Simonsen	19-Oct-09	Complete rework of documentation. Added getUnits() and deleteUnit(), many changes to existing service
1.2.0	M. Simonsen	11-May-10	Added 7 new methods for manipulating Unit Type & Profile, and to return all unitIds for a set of search criteria. User/permissions are now integrated into the services.
1.3.1	M. Simonsen	24-Jun-11	Changed the getUnits-services to allow for more complex searches. Added some restrictions on the usage of unit-id search. Documentation updated.
1.3.1	M. Simonsen	21-Sep-11	Renaming from xAPS to FreeACS
1.3.2	M. Simonsen	14-Dec-11	Updating to 2012R1, removed last chapter (SSL tips)
1.4.3	M. Simonsen	04-Apr-12	Revised edition, removed OPP-references, rewrote some passages, added XML-appendix, improved chapter 3 (General stuff)
2013R1	M. Simonsen	01-Mar-13	Updated to latest version
2014R1	M.Simonsen	28-Aug-14	Updated to FreeACS version

1.4 Acronyms and Abbreviations

Acronym	Explanation
ACS	Auto Configuration Server.
APS	Automatic Provisioning System.
FreeACS	Owera's eXtensible APS with advanced features such as Service Windows, Job Control and Smart Groups.

FreeACS Module	The FreeACS consists of several independently running Modules. The Modules may run on separate hosts in the network.
North Side	Common term describing the part of the FreeACS which includes all FreeACS Modules that provide management interfaces.
South Side	Common term describing the part of the FreeACS which includes all FreeACS Modules that provide CPE communication protocols.
Core	Common term describing the part of the FreeACS which includes all FreeACS Modules that provide neither management interfaces nor CPE communication protocols.
CPE	Customer Premises Equipment. Used in this document to refer to a single physical device. Same as the term "Device".
FreeACS Administrator	Employee of a Hosting Provider managing network infrastructure. Typically the person deploying, configuring and running the FreeACS.
FreeACS Operator	Employee of a Hosting/Virtual Provider managing the provisioned deployment of CPEs through the FreeACS Web Interface.
FreeACS Data Model	How the FreeACS determines which Parameter values and other changes to send to CPEs. Implemented as a database schema with additional processing logic on top in the various FreeACS Modules.
Parameter	Each individual configuration setting is represented in the FreeACS Data Model as a Parameter. A Parameter consists of a name and usually (but not always) a value.
Unit	A dataset in the FreeACS database consisting of Parameter values relating to a single CPE. This dataset may extend beyond the Parameter values actually sent to the CPE, as some Parameter values may only be useful or needed by the FreeACS itself. Also, the dataset may represent only a subset of all the configurable settings in the CPE. For these reasons, it is important to distinguish the term "Unit" from the terms "CPE" and "Device".
Profile	Dataset stored in the FreeACS containing Parameter values shared by multiple Units of the same Unit Type. A Unit is always assigned to a single profile. Multiple Profiles may be created for a Unit Type.
Unit Type	Units that represent CPEs of the same model share a common definition of that CPE model named Unit Type. The Unit Type definition is a list of Parameter names only, as the Unit Type never contains any Parameter values (values are stored in the Unit and/or Profile).
TR-069	Industry standard provisioning protocol used by the FreeACS to read and write configurations from and to the CPEs, in addition to handle upgrades.

2 Introduction

FreeACS Web Services is part of the North Side of FreeACS and its sole purpose is to be a integration interface for other customer specific systems like a BSS or OSS or something else.

FreeACS Web Services interface is designed with emphasis on a re-usable object model. Upon building a Web Service Client you will get the object model from the WSDL and it will be fairly compact. The flip-side of this re-usability is that the content of each object in the services must be carefully documented, hence this documentation.

So far there are nine services in this interface:

- addOrChangeUnit
- deleteUnit
- getUnits
- addOrChangeUnittype
- deleteUnittype
- getUnittype
- addOrChangeProfile
- deleteProfile
- getProfile

For most operators/user, only the first three methods are of interest. The methods allows you to add or change a unit as well as the unit parameters. Deletion of parameters is possible within the addOrChange** service. The addOrChangeUnit method is by far the most important when it comes to basic integration with another system. The delete service is not crucial, but there can perhaps be situations where you would like to control the existence of units from another system. The last service is somewhat more important, as it gives a possibility to get information about one or more units. This information could then form the basis for a decision about add/change/delete Units or Unit Parameter.

3 General stuff

All the services are fairly equal when it comes to input to them. Instead of repeating a lot throughout each section, we'll gather information in this chapter that is considered recommended knowledge for all services.

3.1 Login

To access the services a login is required. Logins are managed and created using Web or Shell. The only user available at startup (before any users are created) is 'admin' with the password 'xaps', and this user will always have access to all parts of the system. We expect the administrators of FreeACS to change the admin password. You can transmit the password in cleartext or as a digest (using SHA-1)[1]. Using this last option and SSL should give you a secure service.

For each login there is set of permissions. These permissions restrict which Unit Types and Profiles you can access. The reason behind this schema is to able to offer a container-feature of FreeACS, so that some users can access Unit Type A and B, while others can access Unit Type C and D.

3.2 Unit Type/Profile

A Unit Type and a Profile are concepts in FreeACS to contain Units. A Unit is always located within one and only one Profile, and one Profile is located within one and only one Unit Type.

Many services require that the Unit Type and Profile are stated explicitly. This may seem to be in conflict with the Login permissions, since a Login implicitly also refers to Unit Types and/or Profiles. Yes, there can be a conflict - that is when the explicitly stated Unit Types and/or Profiles do no match the Login permissions. In that case a SOAP Fault will be returned.

3.3 Unit Id

The Unit id is the unique identifier in FreeACS, e.g. there can be only one Unit in FreeACS for each Unit id. The format of the Unit id varies according to the provisioning protocol:

Protocol	Format	Comment	Example
TR-069	`<OUI>-<ProductClass-<SerialNumber>`	OUI is a 6 digit hexadecimal number to identify the organization ProductClass is often the model name of the device SerialNumber is most often the MAC address	`123456-Modelname-123456789012`
HTTP/TFTP	`<SerialNumber>`	SerialNumber is a unique alfanumerical number. If not universally unique, it should be prefix'ed with a Product-name.	`123456HJK1234`

[1]The SHA-1 digest should be formatted as an alphanumerical upper case string. To make sure you're on the right track, your digest algorithm should convert 'test' into 'A94A8FE5CCB19BA61C4C0873D391E987982FBBD3'

3.4 Serial Number – A unique value

Some years ago, the serial number was the only alternative to reference a unit, except using the Unit-id. Over time FreeACS have adapted to be able to reference a unit using any kind of unique value. However, changing the FreeACS Web Services is something we are very reluctant to do, since it most likely will cause all integrations to be undone. Therefore, we simply treat Serial Number as a "Unique Value" when it comes to searches, so searching for Serial Number = '<customer-id>' will work quite all right as long as the customer-id is actually set on the correct unit.

3.5 Parameter

A parameter is a tuple of information which specify a certain attribute or property of a device or the FreeACS. The allowed parameter names are defined within a Unit Type in FreeACS and cannot be created from these services. The only thing that you can do is to read and write actual values to/from the various units.

Parameter description:

	Comment	**Example**
Name	A name which uniquely identifies this Parameter. The Parameter name **must** be defined on the Unit Type in FreeACS before used in any of these services.	`Device.DeviceInfo.SerialNumber`
Value	The value could be a string up till 1024 characters. The characters could be of any kind, expect very exotic. Remember to URL-encode if necessary (all values will be URL-decoded on the server side).	`Hello World`
Flags	The flags is information to tell the server how to use the Parameter, but can also be used by the client. The flags in use depends upon the service: - addOrChangeUnitRequest: The flags can be either AC (AddChange) or D (Delete). If omitted, AC is default. - addOrChangeUnittypeRequest: The flags can be either AC (AddChange) or D (Delete). If omitted, AC is default. - addOrChangeProfileRequest: The flags can be either AC (AddChange) or D (Delete). If omitted, AC is default. - getUnitsRequest: The flags can be either EQ (Equal) or NE (Not Equal), LT (Less Than), LE (Less or Equal), GE (Greater or Equal) or GT (Greater Than). If omitted EQ is default. Optionally you can add another flag to indicate the type of comparison: TEXT or NUMBER. TEXT is default. Examples: "EQ,NUMBER" or "GT,TEXT" - getUnitsResponse: The flags can be either P (Profile) or U (Unit). Never omitted.	`AC`

3.6 Keyroot

Each parameter that represents an attribute or property of the device always starts it's name with a **`Keyroot`**. This **`Keyroot`** can be either **`Device`** or **`InternetGatwayDevice`**, depending upon the device. Usually a device that can be or should be connected to public IP is a **`InternetGatewayDevice`**, otherwise it is a **`Device`**. This is not a rock solid rule, and the best way to find out is to check the parameters defined in the Unit Type in FreeACS Web or FreeACS Shell.

Examples of parameter names:

```
Device.DeviceInfo.SerialNumber
```

```
InternetGatewayDevice.DeviceInfo.SerialNumber
```

3.7 Exceptions

In case the service for some reason fails it will throw an exception which will be shown as a SOAP Fault. The goal is that this fault message should contain enough information to make it possible to correct the input to the service.

3.8 Color codes

Each input field is in blue color if it is required. Optional input fields are green.

3.9 XML

The recommended way to make the correct XML (SOAP-requests/Web Services Requests) is to generate a Web Service Client based on the WSDL which is always supplied together with this document.

If you do not have the appropriate tools to generate the correct XML, you may read the WSDL and check out the appendix which shows correct XML for some chosen services.

The input/output shown in the examples are not supposed to be interpreted as XML tags, although they do have almost the same name. The difference can be uppercase/lowercase is some cases. The main idea with the input/output is to show what kind of objects/data the various services require.

4 addOrChangeUnit – "add" use case

NB! This chapter is mostly for adding devices not manufactured by Ping Communication. For Ping Communication devices, the adding of new devices is most likely already done through the use of the staging procedure.

This chapter focuses on "add" use cases. This is the case where the FreeACS database has no information whatsoever about the Unit. It is not an add use case if you do not know the Unit id, you must positively know that the unit is not found in the database. If you do not know and run the service, the effect will then be add **or** change. That may not be problematic, it depends on the context and what you are trying to do.

To add a Unit there are two pieces of information that is absolutely necessary:

- Unit id
- Secret (known as ACS-password, provisioning password)

The Unit id has already been described, but the Secret has not, since it's something that only matters if you're going to add a unit. Without the secret there will be no communication between the device and FreeACS (and hence no provisioning), since the communication is not secured.

The secret can be an alphanumerical string of any reasonable length. It is important to understand that the secret that is added to FreeACS (using this service) must match the secret that already exists in the device! There is no room for "generating" a secret into FreeACS, but it must come from a file from the device factory.

Protocol	Parameter name	Example
TR-069	<Keyroot>.X_OWERA-COM.Secret	ASecretStoredInTheDevice
HTTP/TFTP	<Keyroot>.X_OWERA-COM.Secret	ASecretStoredInTheDevice

With this two pieces of information in place, the FreeACS is ready to establish contact with the devices. As already mentioned, a Serial Number can be very useful as well, so if that is an alternative, add the Serial Number as a parameter.

4.1 use case 1: add Unit, known Unit id + secret

This is the use case where you have received a so-called lot-file, with MAC-addresses, secrets and above all: Unit ids. A typical scenario would be that you make a registration page for your customers, they supply the MAC-address found on their device into the web page then the registration server checks with the Taiwan-file and finds the Unit id and the secret. You can of course add as many parameters to the unit as you like, for example information you gather from your customer.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.unittype.name	NPA201E	The Unit Type must be present in FreeACS

	Unit.profile.name	Default	The Profile must be present in FreeACS
	Unit.unitId	123456-Modelname-123456789012	This value must be unique within FreeACS and correspond to the protocol.
	Unit.parameters[0].name	InternetGatewayDevice.X_OWERA-COM.Secret	The parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[0].value	ASe-cretStoredInTheDevice	The actual value of the parameter. This will be stored as a Unit Parameter.
	Unit.parameters[1].name	InternetGatewayDevice.DeviceInfo.SerialNumber	The parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[1].value	123456789012	The actual value of the parameter. This will be stored as a Unit Parameter.
Output	Unit.unitId	123456-Modelname-123456789012	The Unit Id

5 addOrChangeUnit – "change" use cases

A Unit can be changed in many ways:

- Add a Parameter
- Change a Parameter
- Delete a Parameter
- Change the Profile

The only thing you cannot do with a Unit is to change it's connection to a Unit Type and you cannot change the Unit id. If what you want is to move the Unit to a new Unit Type, you will need to delete the Unit all together and add it into the new Unit Type. To actually change the Unit id; delete the Unit and add a new Unit.

5.1 use case 2: change a Unit, known Unit id

When you change a unit you usually know the Unit id. The Unit id is a universal key in FreeACS, so in this case you don't need any more information to identify the Unit.

The profile is changed by simply setting the name of the Profile to something other than the existing Profile. However, the new Profile value must be defined in FreeACS (refer to chapter 3.1).

The Parameters can take on various flags values, as described in chapter 3.4. If you decide to add a Parameter use "AC", in case of deletion use "D". Omitting the flags value will be interpreted as "AC". Naturally you can add as many parameters as you like.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	This unit must be present in FreeACS
	Unit.unittype.name	NPA201E	The Unit Type must be present in FreeACS
	Unit.profile.name	NotDefault	We assume that the Unit was previously in Default Profile. This value will move the Unit to "NotDefault" Profile.
	Unit.parameters[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	The parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[0].value	SIPUser	The actual value to be added/changed for this Unit
	Unit.parameters[1].name	Device.VoiceService.1.VoiceProfile.1.SIP.ProxyServer	The Parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[1].flags	D	The Parameter will be deleted from this Unit
Output	Unit.unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	

5.2 use case 3: change a Unit, unknown Unit id

If you only know the Serial Number (and not the Unit id), and you know that this Serial Number uniquely identifies the Unit (since it's a MAC address or something similar), then you can use the Serial Number. The Unit object has a special field for a Serial Number to be used just in this case (refer to chapter 3.4)

Changing the profile and parameters is like the previous use case.

New in 2011 Release 2: The serial number value can be any value that uniquely identifies the unit (password, ip address, username, etc)

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.serialNumber	123456789012	This serial Number must be found in FreeACS and should represent only one Unit.
	Unit.unittype.name	NPA201E	We assume that the Serial Number we're searching for is defined in a Unit within this Unit Type.
	Unit.profile.name	Default	We assume that the Unit is in Default Profile, hence no profile-move.
	Unit.parameters[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	The parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[0].value	SIPUser	The actual value to be added/changed for this Unit
	Unit.parameters[1].name	Device.VoiceService.1.VoiceProfile.1.SIP.ProxyServer	The Parameter name must be defined within the corresponding Unit Type in FreeACS.
	Unit.parameters[1].flags	D	The Parameter will be deleted from this Unit
Output	Unit.unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	

6 deleteUnit

This service is relatively straight forward - it simply deletes a Unit from FreeACS. The service does not support multiple deletes.

6.1 use case 1: delete a Unit, known Unit id

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	This unit must be present in FreeACS
Output	deleted	true	Will return true if unit was found and deleted. If unit was not found, it will return false.

6.2 use case 2: delete a Unit, unknown Unit id

New in 2011 Release 2: The serial number value can be any value that uniquely identifies the unit (password, ip address, username, etc)

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.serialNumber	123456789012	This serial Number must be found in FreeACS and should represent only one Unit.
	Unit.unittype.name	NPA201E	We assume that the Serial Number we're searching for is defined in a Unit within this Unit Type.
Output	deleted	true	Will return true if unit was found and deleted. If unit was not found, it will return false.

7 getUnits

7.1 Introduction

TMTOWTDI (There's more than one way to do it - or pronounced "Tim Toady") is a famous Perl slogan. This slogan fits this service well. There is a few limitations to the service, but otherwise it's almost like anything you put into it will result in 0 or more Units.

7.1.1 Searching for one single Unit

You can retrieve information about a single Unit using one out these methods:

- UnitId-search: Search for a single Unit using complete Unit id
- UniqueValue-search: Search for a single Unit using complete Serial Number or any other unique value for a unit.

7.1.2 Searching for many Units

The second group of searches are those which are expected to return 0 or more Units. You can use one of these methods or combine the methods freely.

- Simple-search: Search for many Units using Unit Type and/or Profile
- Value-search: Search for many Units using partial Unit id or a (partial) value
- Parameter-search: Search for many Units using Parameters
  - Specify search with one out of six kinds of operators (specify as flags):
    - EQ (Equals) (default)
    - NE (Not Equals)
    - LT (Less Than)
    - LE (Less or Equals)
    - GE (Greater or Equals)
    - GT (Greater Than)
  - A parameter value can be of partial type (see chapter 7.1.5), but only EQ and NE operators apply to partial values
  - A parameter value can be NULL to symbolize the absence of this parameter
  - A parameter can be interpreted as a special type (specify as flags)
    - TEXT (default)
    - NUMBER
  - See chapter 3.5 on how to specify the flags
  - You may repeat the same parameter in the search

Two limitations:

1. If you use a Parameter search you must also specify a Unit Type or a Profile
2. You cannot combine the value-search and the parameter-search.

7.1.3 Preference

Unit Type and Profile information will always be taken into account in all searches. If not provided, the search will be applied within the allowed Unit Types and Profiles.

The priority among the other search criteria are as follows:

1. If Unit Id is detected in the input the search will be a unit-id search. The response will be 0 or 1 unit.
2. If a Serial Number is detected in the input, the search will be a value-search. The response will be 0, 1 (unique value) or more units (not-unique and partial value)
3. If a Parameter is detected in the input, the search will be a parameter-search.

7.1.4 Return limit

The result could range from 0 to 50 hits. The 50 hits limit is specified to avoid a situation where your search hits thousands of Units and the response time goes sky rocketing. To signal that more units were found than returned, the output contains a boolean named "moreUnits". It is not feasible to actually return the number of units in an output field, since that would hit the performance of the query and it is not possible to return the next 50 units (50-100). To overcome this limitation - use the getUnitIds service (see next chapter).

7.1.5 Partial searches

To specify a partial id/value we use a mix of SQL and Regexp syntax (to avoid trouble using % in XML)

- "*" - 0 or more characters of any kind
- "_" - 1 character of any kind

An example search criteria is "Joe*" which would hit on any string that starts with "Joe". You can mix the usage of * and _ as you like. Escape usage of such special SQL characters by using \. In other words the search criteria "Joe*" will return all strings that match "Joe*" exactly.

7.1.6 The result set

The service always returns an array of Units. This array can be 0 or more long. Each unit in the array contains a Unit id, Serial Number, Profile, Unit Type and a set of parameters. If you are somewhat familiar with FreeACS you would know that both the Profile and the Unit Type contains parameters, each of it's own kind. But these parameters are not returned in this service. The reason: They would clutter the result set with too much useless or confusing information. Thus, both Unit Type and Profile are represented in the result set only by their names.

But there is more to be said about the Profile, because there is an interesting principle about Profiles and Units: A Profile parameter is inherited by the Units, unless overridden by a Unit parameter. This is important, because many of the parameters that a Unit holds is not located on the Unit level, but found on the Profile level. To make it simple for the user of this interface, we decided to return all parameters, both Profile and Unit parameters in the list of parameters for each Unit. The flags will be set to either "P" or "U" to show from where the parameter was taken.

7.1.7 Miscellaneous

If a Profile name is specified, Unit Type name must also be specified.

The Profile and Unit Type objects are made with lists of parameters (check with WSDL), but they are never used/populated. The intention is to be able to provide services in the future which focuses on these concepts without breaking the existing interface. Whether that will succeed remains to be seen.

7.2 use case 1: search for a single Unit, known Unit id

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	This unit should be present in FreeACS
Output	Unit[0].unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	The Unit id
	Unit[0].serialNumber	123456789012	The Unit's Serial Number
	Unit[0].profile.name	Default	The Unit's Profile
	Unit[0].unittype.name	NPA201E	The Unit's Unit Type
	Unit[0].parameter[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	A Unit parameter name
	Unit[0].parameter[0].value	SIPUser	A Unit parameter value
	Unit[0].parameter[0].flags	U	The flag indicated that this is a Unit Parameter
	Unit[0].parameter[1].name	Device.DeviceInfo.SerialNumber	A Unit parameter name
	Unit[0].parameter[1].value	123456789012	A Unit parameter value
	Unit[0].parameter[1].flags	U	The flag indicated that this is a Unit Parameter
	moreUnits	false	No more units found

7.3 use case 2: search for a Unit(s), using a parameter value

New for 2011 Release 2: The serialNumber does not really have to be a serialNumber. Any unit parameter value will do (e.g. ip-adress, mac-address, username, etc). If Unit Type name is omitted, the search will be performed among all Unit Types. If the "serial number" value supplied is not unique or even a partial value (containing % or _), the result may be more than 1 unit (up until 50). However, the response will contain the serial number (if it exists) in the same field. This is done to avoid breaking the interface of the service.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.serialNumber	123456789012	Use any kind of unique value (ip-address, mac-address, username, password, serialnumber)
	Unit.unittype.name	NPA201E	The Unit Type must be present in FreeACS
Output	Unit[0].unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	The Unit id
	Unit[0].serialNumber	123456789012	The Unit's Serial Number
	Unit[0].profile.name	Default	The Unit's Profile
	Unit[0].unittype.name	NPA201E	The Unit's Unit Type
	Unit[0].parameter[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	A Unit parameter name
	Unit[0].parameter[0].value	SIPUser	A Unit parameter value
	Unit[0].parameter[0].flags	U	The flag indicated that this is a Unit Parameter
	Unit[0].parameter[1].name	Device.DeviceInfo.SerialNumber	A Unit parameter name

	Unit[0].parameter[1].value	123456789012	A Unit parameter value
	Unit[0].parameter[1].flags	U	The flag indicated that this is a Unit Parameter
	moreUnits	false	No more units found

7.4 use case 3: search for many Units, using various criteria

This example only show one Unit in return, but there could be a list of up to 50 units in a list, all complete with both unit parameters and profile parameters.

New for 2011 Release 2: No longer possible to use unit-id in this search, it will then be interpreted as a use case 1 search.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unit.unittype.name	NPA201E	The Unit Type must be present in FreeACS
	Unit.parameter[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	The parameter name must be present in the Unit Type NPA201E (mentioned above)
	Unit.parameter[0].value	Joe%	The value to search for must start with "Joe".
	Unit.parameter[0].flags	NE,TEXT	Not Equal flag (see chapter 3.5). This will turn the search into find all Units where AuthUserName do not start with "Joe".
	Unit.parameter[1].name	Device.DeviceInfo.Uptime	The parameter name must be present in the Unit Type NPA201E (mentioned above)
	Unit.parameter[1].value	1440	Represents in this case 24h = 1440 minutes

	Unit.parameter[1].flags	GT,NUMBER	Greater Than (see chapter 3.5). This should prompt the search to only return the units with uptime higher than 24h.
Output	Unit[0].unitId	dd37cbd4-d4a3-3d66-8958-5d9cf30635f5	The Unit id
	Unit[0].serialNumber	123456789012	The Unit's Serial Number
	Unit[0].profile.name	Default	The Unit's Profile
	Unit[0].unittype.name	NPA201E	The Unit's Unit Type
	Unit[0].parameter[0].name	Device.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUserName	A Unit parameter name
	Unit[0].parameter[0].value	JoeDoe	A Unit parameter value
	Unit[0].parameter[0].flags	U	The flag indicated that this is a Unit Parameter
	Unit[0].parameter[1].name	Device.DeviceInfo.Uptime	A Unit parameter name
	Unit[0].parameter[1].value	1441	A Unit parameter value
	Unit[0].parameter[1].flags	U	The flag indicated that this is a Unit Parameter
	moreUnits	false	No more units found

8 getUnitIds

This methods is the cousin of the getUnits service. The difference is this:

- no limitation on the result, all unitIds that match the search will be returned
- only unitId will be returned in the result
- higher performance for large result sets

The idea of this unit is to be able to retrieve a list of unitIds. To get more information about a certain unit, you need to run the getUnits service.

9 addOrChangeUnittype

The methods mentioned so far in this document (Unit-methods) are intended for integration between an FreeACS customer requiring a limited/standard integration. It is then assumed that add/change/delete operations on Unittype/Profile is something that will be done manually (using FreeACS Web or FreeACS Shell).

However, some customers will need a more tight integration, since creating a Unittype/Profile will be repeated by their customers. In general this kind of integration will only be necessary with customer that is not an end-customers, that is FreeACS will be part of another system which in turn will be sold to an end-customer. We recommend against using these methods if you're not this kind of customer, because a tighter integration also will hurt more every time the these methods are subject to change. This is not to say changes will happen frequently, but it is unlikely that we'll be able to keep all these interfaces stable over years.

If you create a unittype using a login which is not an admin user, the unittype will be created

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unittype.name	MobilePhone	The name of the unittype
	Unittype.matcherId	000000	The matcherId may be important if protocol is OPP. Check with Owera in this case to set it correctly.
	Unittype.description	A mobile device	A description
	Unittype.vendor	Nokia	The manufacturer of the phone.
	Unittype.protocol	TR-069	Can be TR-069, OPP or N/A. This decision will impact on which format is accepted in the unitIds within this unittype.
	Unittype.parameters[0].name	Device.DeviceInfo.SerialNumber	The name of the unittype parameter
	Unittype.parameters[0].value	SAR	The flag of the unittype parameter. Must be R, RW or X. Additionally allowed: S (Serachable), D (Displayable in FreeACS Web), I (Inspection), A (Always-Read), C (Confidential)
	Unittype.parameters[0].flags	AC	Indicates whether the parameter is to be add or changed (AC) or deleted (D).
Output	Will return the unittype object (but will include all data from FreeACS, not only the data changed (if this is a change request) in this call)		

10 deleteUnittype

The deletion of a Unit Type is intentionally difficult. A Unit Type is the most basic concept in FreeACS and many other concepts derive/depend upon Unit Type. So before a Unit Type can be deleted, all other dependcies must be deleted. Some things are deleted automatically by this service, like Syslog Events, Groups and Unit Type Parameters. However, Softwares, Jobs and Permissions will not be deleted automatically - these must be deleted via FreeACS Shell or FreeACS Web. Profiles and Units must also be deleted separately, but that can be done using these services.

The service is relatively simple to use. If deletion fails it will usually return a SoapFault, but to be 100% sure, check the isDeleted-flag in the output.

	Object	**Example value**	**Comment**
Input	`Login.username`	`admin`	Username
	`Login.password`	`xaps`	Password
	`unittypeName`	`MobilePhone`	`The name of the unittype`
Output	isDeleted	true	Will return false if the delete operation (in SQL) did not return 1 row deleted.

11 getUnittypes

getUnittypes will return all the Unit Types you are allowed to see. If you specify a particular Unit Type you will only receive that one.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unittype.name	MobilePhone	The name of the unittype
Output	Unittype.name	MobilePhone	The name of the unittype
	Unittype.matcherId	000000	The matcherId may be important if protocol is OPP. Check with Owera in this case to set it correctly.
	Unittype.description	A mobile device	A description
	Unittype.vendor	Nokia	The manufacturer of the phone.
	Unittype.protocol	TR-069	Can be TR-069, OPP or N/A. This decision will impact on which format is accepted in the unitIds within this unittype.
	Unittype.parameters[0].name	Device.DeviceInfo.SerialNumber	The name of the unittype parameter
	Unittype.parameters[0].value	SAR	The flag of the unittype parameter. Must be R, RW or X. Additionally allowed: S (Serachable), D (Displayable in FreeACS Web), I (Inspection), A (Always-Read), C (Confidential)
	Unittype.parameters[0].flags		

12 addOrChangeProfile

Profile is a rather simple concept, only requiring a name. The ability to set common parameter values for all units within a profile is the most interesting aspect of a Profile. Some may also find it useful to restrict permissions to a specific Profile.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unittype.name	MobilePhone	The name of the unittype
	Profile.name	Default	The name of the profile
	Profile.parameters[0].name	Device.ManagementServer.URL	The name of the Unit Type parameter (the parameter must already exist in the Unit Type)
	Profile.parameters[0].value	http://xaps.com/tr069	The value of the parameter.
	Profile.parameters[0].flags	AC	Indicates whether the parameter is to be add or changed (AC) or deleted (D).
Output	Will return the profile object (but will include all data from FreeACS, not only the data changed (if this is a change request) in this call)		

13 deleteProfile

The deletion of a profile is relatively simple. Just keep in mind that you may need to delete Groups and Permissions and Units associated with this particular Profile first. Profile Parameters will be deleted automatically.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unittype.name	MobilePhone	The name of the unittype
	Profile.name	Default	The name of the profile
Output	isDeleted	true	Will return false if the delete operation (in SQL) did not return 1 row deleted.

14 getProfiles

getProfile will return all the Profile you are allowed to see. If you specify a particular Profile you will only receive that one.

	Object	Example value	Comment
Input	Login.username	admin	Username
	Login.password	xaps	Password
	Unittype.name	MobilePhone	The name of the unittype
	Profile.name	Default	The name of the profile
Output	Profile.name	Default	The name of the unittype
	Profile.parameters[0].name	Device.ManagementServer.URL	The name of the unittype parameter
	Profile.parameters[0].value	http://xaps.com/tr069	The value of the parameter.
	Profile.parameters[0].flags		

15 Redirection/Troubleshooting

The FreeACS WS Server offers a redirection feature. This is mostly meant for troubleshooting, but could probably be useful in other cases as well. The whole idea with redirecting is to be able to catch the XML to/from the server and store it to a file. This is important since the XML is often lost when working with Web Service framework. Comparing various XML output can be effective to pinpoint where things went wrong.

The URL to use for WS-clients is http://<host>:<port>/xapsws/redirect

The XML will be logged to xaps-ws-xml.log, or if otherwise configured in xaps-ws-logs.properties.

Make sure to configure the correct URL to redirect to the FreeACSWS service in xaps-ws.properties.

16 Appendix

16.1 Add/Change unit (chapter 4.1/5.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <AddOrChangeUnitRequest xmlns="http://xapsws.owera.com/">
      <login xmlns="">
        <username>user</username>
        <password>pass</password>
      </login>
      <unit xmlns="">
        <unitId>000000-TR069TestClient-000000000001</unitId>
        <profile>
          <name>Default</name>
        </profile>
        <unittype>
          <name>NPA201E</name>
        </unittype>
        <parameters>
          <parameterArray>
            <item>

<name>InternetGatewayDevice.Services.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthPa
ssword</name>
              <value>freyrsSipPassword</value>
            </item>
            <item>

<name>InternetGatewayDevice.Services.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUs
erName</name>
              <value>freyrsSipUsername</value>
              <flags>AC</flags>
            </item>
            <item>
              <name>System.X_OWERA-COM.Secret</name>
              <value>0123456789012345</value>
            </item>
          </parameterArray>
        </parameters>
      </unit>
    </AddOrChangeUnitRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

16.2 Change unit (chapter 5.2)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <AddOrChangeUnitRequest xmlns="http://xapsws.owera.com/">
      <login xmlns="">
        <username>user</username>
        <password>pass</password>
      </login>
      <unit xmlns="">
        <serialNumber>012345678902</serialNumber>
        <profile>
          <name>Default</name>
        </profile>
        <unittype>
          <name>NPA201E</name>
        </unittype>
        <parameters>
          <parameterArray>
            <item>

<name>InternetGatewayDevice.Services.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthPa
ssword</name>
              <value>freyrsSipPassword</value>
            </item>
            <item>

<name>InternetGatewayDevice.Services.VoiceService.1.VoiceProfile.1.Line.1.SIP.AuthUs
erName</name>
              <value>freyrsSipUsername</value>
              <flags>AC</flags>
            </item>
            <item>
              <name>System.X_OWERA-COM.Device.MAC</name>
              <value>012345678902</value>
            </item>
          </parameterArray>
        </parameters>
      </unit>
    </AddOrChangeUnitRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

16.3Search using unit-id (chapter 7.2)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <GetUnitsRequest xmlns="http://xapsws.owera.com/">
      <login xmlns="">
        <username>user</username>
        <password>pass</password>
      </login>
```

```
    <unit xmlns="">
      <unitId>002194-NPA201E-012345678901</unitId>
    </unit>
  </GetUnitsRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

16.4Search using unique value (chapter 7.3)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <GetUnitsRequest xmlns="http://xapsws.owera.com/">
    <login xmlns="">
      <username>user</username>
      <password>pass</password>
    </login>
    <unit xmlns="">
      <serialNumber>ohHAII%</serialNumber>
    </unit>
  </GetUnitsRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

16.5Search using parameter-search (chapter 7.4)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <GetUnitsRequest xmlns="http://xapsws.owera.com/">
    <login xmlns="">
      <username>user</username>
      <password>pass</password>
    </login>
    <unit xmlns="">
      <profile>
        <name>Default</name>
      </profile>
      <unittype>
        <name>NPA201E</name>
      </unittype>
      <parameters>
        <parameterArray>
          <item>
            <name>System.X_OWERA-COM.Device.MAC</name>
            <value>00219401CA29</value>
            <flags>EQ</flags>
```

```
          </item>
        </parameterArray>
      </parameters>
    </unit>
  </GetUnitsRequest>
 </soapenv:Body>
</soapenv:Envelope>
```