FreeACS TR-069 Server
User Manual

2014R1

Table of Contents

1 Document Introduction

1.1 Document Purpose

The document should teach an operator to configure and run the TR-069 Server.

1.2 Document Audience

The audience should understand the basic concept of provisioning and have some general knowledge about FreeACS. The readers will probably be operators of the server and future developers and testers of the server.

1.3 Document History

Version	Editor	Date	Changes
2.2.1	M. Simonsen	23-Mar-09	Updated to latest version of TR-069 Server
2.3.0	M. Simonsen	03-Apr-09	Revised edition
2.3.2	M. Simonsen	30-Jun-09	Revised edition
2.3.3	M. Simonsen	12-Nov-09	Revised edition
2.3.10	M. Simonsen	28-Sep-10	Revised edition
2.4.1	M. Simonsen	07-Nov-10	Handles repeating jobs and TR-statistics
2.4.5	M. Simonsen	17-Mar-11	Revised edition
2.6.2	M. Simonsen	13-Dec-11	Changed some chapters
2013R1	M. Simonsen	25-Feb-13	Updated to latest version
2014R1	M. Simonsen	05-Aug-14	Updated to latest version

1.4 References

Document
[1] FreeACS Installation

2 Introduction

Before reading this introduction, we expect you to understand the basic concepts of FreeACS and that it is set up accordingly to [1]. This document will focus on how to configure the TR-069 Server.

The TR-069 Server is responsible for the communication with CPEs, and communicates over TR-069 protocol. There are other servers in FreeACS that also communicates with CPEs, but through other protocols (SPP Server supports HTTP/TFTP).

The CPEs must be configured with a URL which points to this server. When this is in place, the TR-069 Server will fulfil its obligation: Provision parameters to the CPEs, read parameters from the CPEs, update the database with data from the CPE and upgrade the firmware/software and configuration.

The server should be able to handle millions of devices every day. The exact performance varies with many factors, but if the configuration of the server and the provisioning focuses on performance alone, it should be possible to provision 10-15 million devices per day. If emphasis is on job control the figure will be substantially lower, although still in the millions.

3 Security

3.1 Authentication

The server can be run with no authentication, basic authentication and digest authentication, the latter recommended for TR-069 devices. To change the settings, look in the "propertyfile" chapter. In addition to changing the property file you need to create a secret for each of the CPEs. That secret is specific for each CPE and the parameter name is:

```
System.X_OWERA-COM.Secret
```

This value is something that is populated in the CPE (from the factory) and the FreeACS will need to know this value (in the database) to get it working.

You could run authentication without using SSL, but that would open up the possibility for a man-in-the-middle attack.

4 Propertyfiles

4.1 xaps-tr069-logs.properties

The log property file is self-documented and should be easy to edit. The main points is that there is defined 6 different logs in TR-069 Server.

4.2 xaps-tr069.properties

```
# *** xAPS TR-069 Server Configuration file ***

# --- Various controls ---

# Allowed values are "none", "basic" and "digest". Digest authentication
# is default, and it is the most secure way to communicate with the devices.
# Combining this with SSL-setup, will give you a very secure provisioning.
auth.method = digest

# Will require username/password to download a Firmware/Config file/etc
# using the TR-069 Download method. FreeACS will instruct the CPE to use
# the ACS-username/password in the HTTP basic/digest challenge. Default
# is false, since this is a change introduced in version 3.1.0. It will become
# default within a few releases.
file.auth.used = false


# Discovery Mode can be set to true if you want to automatically add a new
# unittype and unit. This mode is violating the security of the system,
# because it allows unknown units to connect and then changes will be performed
# in the database. So use this option with caution, preferably when you want to
# add a new unittype to the system. Default is false.
discovery.mode = false

# Comma separated black-list (if discovery.mode is true) - units with
# ACS-username containing these strings will be blocked.
discovery.block =

# concurrent download limit will limit the number of concurrent downloads
# allowed from this provisioning server. This is done to conserve bandwidth.
# This will override jobs/servicewindows if necessary, thus postponing the
# download to later. Default is 1000000 (virtually no limit).
concurrent.download.limit = 1000000


# --- Quirks ---
#
# unitdiscovery (perform full unit discovery for every unit)
#
# If the supported parameters for a certain unittype changes a lot, it will
# make sense to discover the capabilities of every unit every time. Instead of
# doing an elaborate and complex discovery of the unit, we simply ask for all
# parameters values upon every TR-069 session initiated. This is costly for the
# device, and some device may not handle this very well.
#
# parameterkey (do not return parameterkey)
#
# TR-069 specifies a parameter key which the ACS could set to the CPE and
# retrieve if and only if a change (SetParameterValue) was executed
# successfully. This is important to verify that a change was ok. However some
# devices do not return this parameter key as they should, hence som of
# the verification of a change is compromised.
#
# termination
#
# The termination quirk will requires the session to terminate using
# Empty(ACS) - Empty(CPE) - Empty(ACS) as the final methods. This is according
```

```
# to the original specification of TR-069. From amendment 1 it was decided
# that a final Empty(ACS) was enough, and this is the default behavior.
#
# prettyprint
#
# The device may not format the XML requests nicely. This quirk will make
# sure the conversation log will be easier to read. The formatting will
# be done even if the XML contains illegal characters. The reason to avoid
# this quirk is performance and perhaps unnecessary.
#
# xmlcharfilter
#
# Some times the device will output XML which contains invalid XML characters.
# This quirk filters such characters before XML parser receives the stream.
# The reason to avoid this quirk is performance and perhaps unnecessary.
#
# ignorevendorconfigfile
#
# Establish which "vendor config files" (could be any kind of file really,
# but TR-069 terminology is "config") are installed on the device
# and furthermore, whether a new "vendor config file" should be uploaded to
# the device. To support this, the firmware MUST be able to answer a request
# for "InternetGatewayDevice.DeviceInfo.VendorConfigFile." object in a
# GetParameterValue request. In case no vendor config file exists, the
# device MUST NOT return an error, simply return a list of 0 parameters.
# This behavior is really standard TR-069 (since many years back), but
# asking for an object is still something that some units may have trouble
# with, hence the possibility to turn off this feature.
#
# nextlevel0ingpn
#
# Some devices doesn't support the usage of <NextLevel>false</NextLevel>
# in the GetParameterNames-request. Instead they may support
# <NextLevel>0</NextLevel>.
#
# Specify quirks like this:
#
# quirks.<unittypename>[@<version>] = <quirkname>(,<quirkname>)*
#
# If you specify quirks for a version, then quirks specified for the unittype
# only is ignored all together (for that particular version of course). This
# way you can make default quirks for a unittype, and then only specify a few
# versions that have different quirks. Examples:

quirks.SpeedTouch 780 = parameterkey
quirks.SpeedTouch 780@6.2.29.2 = parameterkey,termination
quirks.P-2602HW-F3 = parameterkey
quirks.HydrogenHA = xmlcharfilter,prettyprint,unitdiscovery
quirks.freecwmp = prettyrprint,xmlcharfilter

# --- Database ---

# xAPS database connection
db.xaps.url = xaps/xaps@jdbc:mysql://localhost:3306/xaps

# Max connections. Default is 100.
db.xaps.maxconn = 100

# Syslog database connection
# Default is to place syslog on the same database as xaps. However, you may
# specify a database placed elsewhere, to relieve the xaps database of
# excessive load from syslogging.
db.syslog = db.xaps


# --- DEBUGGING ONLY, NEVER SET THESE IN PRODUCTION ----
# Test mode is set to true to test specific XMLs found in the tests-folder. Run
# the URL /test to choose which tests to be run. Default is false.
debug.test.mode = false
```

4.2.1 Various Controls

Decide whether you want authentication or not. Recommended setting is "digest", but it is also possible to use "basic" and "none". To use authentication you must also add the "secret"-parameter in the database, as stated in the previous chapter.

Discovery mode should normally be false (which is default). If it is true, then you violate the security model, since we accept the incoming traffic even without the secret parameter in the database. The point with this mode is to allow you to hook up a CPE to the ACS and then auto-populate the FreeACS database with all necessary information, to then provision the device. Actually, both unittype, unittype parameters, profile, unit and the secret unit parameter are created on the fly. Note: If the device does not support basic authentication, discovery mode will not work, since we then have no way to get the secret from the CPE. This whole procedure will only run once; the next time the CPE connects it will perform a standard conversation, although always using basic authentication (never using digest authentication).

If you use the job functionality of FreeACS you can set a certain limit of concurrent downloads. This is useful when the number of CPEs to upgrade might be more than your network can handle.

4.2.2 Quirks

A quirk is an adaptation of the server behaviour to fit a violation of the spec or to support an old version of the spec. Non the less, sometimes it can be useful to tinker with these settings, especially if you get an updated software version for the CPE which has a slightly changed TR-069 client.

4.2.3 Database

The database settings are the same for all FreeACS servers. You must specify a "URL" to a database with the format:

<user>/<pass>@jdbc:mysql://<hostname>:<port>/<dbname>

You can change the number of maximum connections to the database, but it should be substantial.

4.2.4 Debug

By setting the debug.test.mode to "true" you can test a CPE. You must then go to the URL:

http://<hostname>/tr069/test

..and register the device you want to test. Furthermore, you must copy the folder "tests" that you find within the tr069.war into your working folder (/var/lib/tomcat7/) and within this folder in turn, create "results" folder and "modified" folder.

5 Firmware/config file download

The TR-069 server works like this: Check whether a device has the correct (wanted) version of software or config files. If not, then issue a Download method with the correct/new software/config file. The file is by default found in the File storage of FreeACS (in the FreeACS database) and it is identified based on the version number and file type.

For software, the server checks the parameter

- DeviceInfo.SoftwareVersion

And compares this parameter value with the System parameter

- System.X_OWERA-COM.DesiredSoftwareVersion

No Download method is issued if these are equal.

For config files it is somewhat more complicated. The server asks for

- DeviceInfo.VendorConfigFile.

Notice the trailing dot at the end of the parameter name. The device should return the entire object of VendorConfigFile, which could be multiple entries like this:

- Device.VendorConfigFile.1.Version
- Device.VendorConfigFile.1.Name
- Device.VendorConfigFile.2.Version
- Device.VendorConfigFile.2.Name
- …
-

NB! If the device cannot support such a questions, add the "ingnorevendorconfigfile" quirk to the unittype (see chapter 4.2.2) to avoid error messages. You will not be able to download config files to the device.

If the device do support the VendorConfigFile and return it, then you may specify some System-parameters to tell the server which config files should be present. The system-parameters must match the vendor-config-file parameters:

- System.X_OWERA-COM.TR069Script.<Name>.Version = <Version>

The files to be downloaded to the device are found in the FreeACS database (as earlier indicated). However, you may override this behaviour if you specify the URL to be used in the Download method.

For the software: System.X_OWERA-COM.SoftwareURL
For the TR069-script/config-file: System.X_OWERA-COM.TR069Script.<Name>.URL

5.1 Software upgrade wizard (FreeACS Web)

To help you set these parameters, we provide a Software Upgrade Wizard in Fusion Web. This wizard can set a specific unit to upgrade or a specific profile to upgrade. If you have the Job Control Server module, you can also set an upgrade for a particular job. Having this functionality, you can decide to upgrade a particular set of units. A job not only helps

you to choose a set of units, but it is also able to control to progress of the upgrades. If some devices fail, this can trigger a stop in the job, so to avoid sending out a lot of softwares that causes malfunction. There is a lot more to be said about jobs, but that will be covered in another document.

5.2 Software upgrade using FreeACS Shell

Everything you can do in Web, you can do in Shell. However, something is better in Shell. Let's assume you are interested to deploy a new software on a set of units. The set is of course not the same as all the units in the profile, since you could then just set the parameters on the profile. With Shell, you can make a selection like this:

```
>listunits Device.X_OPERATOR-COM.Country = UK > UNITS_IN_UK.txt
```

Then you can use this output file (UNITS_IN_UK.txt) to set the necessary parameters on each unit.

```
> FROMFILE[1] setparam Device-X.OWERA-COM.DesiredSoftwareVersion 2 < UNITS_IN_UK.txt
```

However, as long as you don't use jobs, the change will be uncontrolled; the CPE will apply the change with no regard whether other CPEs fail or not.

6 Tuning parameters

There are some important parameters in the server:

- Maximum number of simultaneously served servlets (**max-servlets**)
- Maximum number of simultaneously served database requests (**max-conn**).

6.1 Max-servlets

Max-servlets is a setting usually found in server.xml (true for Tomcat and JBoss). There are several parameters, as seen in this example from Tomcat:

```
    <Connector acceptCount="5" connectionTimeout="20000" disableUploadTimeout="true"
enableLookups="false" maxHttpHeaderSize="8192" maxSpareThreads="10" maxThreads="10"
minSpareThreads="10" minThreads="10" port="80" redirectPort="8443"/>
```

In this example maxThreads equals 10, and that is the setting controlling max-servlets. In addition it is possible to set acceptCount to a lower number. This setting controls how many servlet requests will be queued up before rejected. Remembering that there will be situations with overload on the server, it is important that the setting is kept so low that the CPU will never go into 80-100% area. If the CPU is kept at a comfortable level at all times, then the server will be able to quickly reject any incoming request whenever the load reaches max-servlets. If the CPU starts to climb to 100% it can increase the problems even more, because more and more CPEs are "hanging" on to the server, waiting for a reply.

6.2 Max-conn

Having decided upon max-servlets, it is time to tune max-conn (found in xaps-tr069.properties). The number of connections to the database is not always something you can decide for yourself, independently of other users of the same database. So in this "shared database environment" you could face a situation where only a limited number of connections are available. Now, if you only are using one single TR-069 server, then there is no question about the max-conn, just set it as high as possible/allowed (no need to set it higher than max-servlets). If there are multiple TR-069 servers, then it could be worthwhile to think a little. Consider this case:

You have 20 database connections available and 4 servers available. One server can easily handle 50% of traffic in terms of CPU/memory.

In this case it makes more sense to divide the connections on 2 servers than on four. To understand why, you need to focus on the probability for getting a connection: A database connection is needed only for a short period during a conversation; therefore multiple clients could reuse the same connection even though they access the TR-069 server at the same time. The probability P for using a database connection during a session, is (to make it easy) the relationship between the length of the session compared to the length of the database request. The chances of two sessions needing one database connection are P^2 and for three connections it is P^3 and so on. The math can be quite complicated when you are going to calculate the likelihood of X connections being enough to cover for 99,999% of the traffic, however, it could be intuitively understood that as the number of connections available increases (linear), the probability of using all of them at the same time decreases rapidly (exponential). The simple conclusion is that a slight increase in the number of connections can easily handle a doubling of the load on the server.

To make things more complicated, the probability P changes as a function of the load on the server. With heavier load, P increases (mostly because of the database). Then the number of required connections will increase rapidly.

The conclusion of this is simple: do not spread your resources too thin, it can generate more problems than you solve. You can follow the usage of the connections on the monitoring page, and tune your system accordingly.

6.3 Both of the parameters

When you understand how to tune each parameter, then you need to look at both of them at the same time. The point is that the database itself may be a bottleneck. If so, then the time consumption for each database call will increase with heavier load (more parallel database access). If so, you will come to a certain point where the number of connections available is so high, that each database access is slowed down to a point where the throughput will no longer increase. You should then lower the number of database connections and lower max-threads so that the traffic into the ACS is no bigger than the database can handle. At this point you should have a perfectly tuned system. If the database is powerful enough, it will cover all your needs. If the database is very powerful (or your ACS-server specs are very poor) you might need several ACS-servers.

6.4 Other important factors

To increase performance, turn off logging to the SYSLOG appender. Another clue is to increase log levels, so very little is written to file.

7 TR-111 & STUN Server

TR-111 is a specification on how to use STUN to trigger an Inform on the CPE behind NAT. In plain English: This is about the server triggering the devices, not the other way around as is usual.

To make this feasible a STUN server is necessary. A STUN server is a server which is in contact with all it's STUN clients all the time. The CPEs must then run STUN clients and be able to receive a special "trigger"-message from the STUN server. The STUN Server is of course in contact with FreeACS, so that it will send those trigger messages whenever requested.

7.1.1 xaps-stun.properties

```
# The Stun server needs 2 network interface to work

# 3478 is the default Stun server port
primary.port = 3478
secondary.port = 3479

# Primary ip. Specify the default interface of your computer
primary.ip = 10.11.10.255
#primary.ip = 85.112.159.52

# A secondary interface is not necessary to run TR-111 operations (it
# uses only parts of the STUN specification). If you want to run this
# server as a regular STUN server then you should also add a secondary
# interface. If no ip is specified, 127.0.0.1 will be used anyway.
#secondary.ip = 85.112.159.55

# Specify the interval between each kick in a group-kick.
# Specify in milliseconds. Default is 1000. If set too low,
# the server will not be able to comply, since only one thread
# is running the requests and may in some circumstance wait
# for reply from the devices. (If not TR-111 is supported)
kick.interval = 1000

# Specify the interval in minutes between each time a job/group is scanned
# for changes in the group (units added or removed). If set too
# low it may cause higher load on server. This rescan will only
# happen if the kick-process is idle (not actively kicking any devices)
# Default is 60 minutes.
kick.rescan = 60

# The kick-feature of the STUN server will try to use the public IP
# of the device in the ConnectionRequestURL, if the ConnectionRequstURL
# is private and UDPConnectionRequestAddress (TR-111) is not defined.
# Default is false
kick.expect-port-forwarding = false


# FreeACS database connection
db.xaps.url = xaps/xaps@jdbc:mysql://localhost:3306/xaps

# Syslog database connection
# Default is to place syslog on the same database as xaps. However, you
# may specify a database placed elsewhere, to relieve the xaps database
# of excessive load from syslogging.
db.syslog = db.xaps

# This is for test-mode only, never used in production. Default is true.
# If set to false, the STUN-server will not start, and it will only support
# ConnectionRequest to the devices (not UDPConnectionRequest)
test.runwithstun = true
```

You must specify the primary IP and the db.xaps.url for this to work. The kick-properties is about how the server should handle a massive kick-operation. The test.runwithstun can be set to false, if you have problems with the bind-operation. In this case, the STUN server will be able to kick the device over the TCP/HTTP connection. This will not work unless you have the device on public IP OR you have set up port forwarding.

Make sure that these parameters are read (found in the FreeACS database) if you want to test/run the TCP/HTTP connection kick (this is not really TR-111):

- ManagementServer.ConnectionRequestURL
- ManagementServer.ConnectionRequestPassword
- ManagementServer.ConnectionRequestUsername

If you want STUN kick to work (over UDP), then the following parameters must be found in the FreeACS database:

- ManagementServer.UDPConnectionRequestAddress

In addition you should/must specify/provision the following parameters for the STUN client to work properly on the CPE:

- ManagementServer.StunEnable = 1
- ManagementServer.StunServerAddress = <IP> or <hostname>
- ManagementServer.StunServerPort = 3478
- ManagementServer.StunMinimumKeepAlivePeriod = 20
- ManagementServer.StunMinimumKeepAlivePeriod = 30

If all this is in place, this may work. Check the logs of the STUN server, especially the fusion-stun-kick-single.log.