Fusion TR-069 Server
User Manual

2013R1

Table of Contents

1 Document Introduction

1.1 Document Purpose

The document should teach an operator to install, run and monitor the Fusion TR-069 Server.

1.2 Document Audience

The readers are expected to have knowledge about how to set up a standard Java Enterprise Application Server (like Jboss). Apart from that, they should understand the basic concept of provisioning and have some general knowledge about Fusion. 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

1.4 Acronyms and Abbreviations

Acronym	Explanation
APS	Automated Provisioning System
Fusion	Owera’s eXtended APS with advanced features such as authentication, encryption, syslog, diagnostics
ACS	Auto-Configuration Server
CPE	Customer Premises Equipment
TR-069	A communication protocol between ACS and CPE. Works on top of HTTP(S), using SOAP.
EAR	Enterprise Application Archive – represents a package which contains the whole server before deployment.
J2EE	Java 2 Platform, Enterprise Edition

1.5 References

Document

[1]	Fusion Product Description
[2]	Fusion Installation
[3]	Fusion Shell User Manual

2 Introduction

Before reading this introduction, we expect you to understand the basic concepts of Fusion and that Fusion is set up accordingly ([1] and [2]). This document will focus on how to monitor and run 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 Fusion that also communicates with CPEs, but through other protocols.

The CPEs must be configured with a URL which will point 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 is focused on performance alone, then it should be possible to provision 10-15 million devices per day. If emphasis is more 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 shared secret for all the CPEs. That shared 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 Fusion 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. We have documented how to set up SSL on JBoss in [2].

4 Propertyfiles

4.1 xaps-tr069-logs.properties

The log property file is self-documented and should be fairly 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

# 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

# Discovery Mode will only run if a device is not recognized in the database.
# If you want to force discovery mode, because you want to update the unit type
# parameters in the database, then set this parameter to true. This will cause
# all units to run GPN + GRM, and update all unittype parameters found for that
# particular unit. Needless to say, this is a major performance killer. Never
# run this in production, unless in dire need! Discovery.mode must be true for
# this to work. Default value is false.
discovery.force = false

# 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
# parameter values upon every TR-069 session initiated. This is costly for the
# device, and some device may not handle this very well.
#
# configtargetfile (set configtargetfile to a file name)
#
# The usage of target file name in the Download method is really not
# necessary and is not required by the specification. However, one
# type of devices has seen to require this file name specified. Thus
# the file name is set to "test.sts" if the quirk is activated.
#
# 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.
#
# 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 = configtargetfile,parameterkey
quirks.SpeedTouch 780@6.2.29.2 = configtargetfile,parameterkey,termination
quirks.P-2602HW-F3 = parameterkey
quirks.Thomson TG784 = configtargetfile
quirks.Thomson TG789vn = configtargetfile
quirks.HydrogenHA = xmlcharfilter,prettyprint,unitdiscovery


# --- 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 Reload and Various Controls

The reload property at the top control the reloading of this property file. The goal is to be able to make changes to this property file while the server is running.

Next, decide whether you want authentication or not. Recommended setting is "digest", but it's 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 Fusion database with all necessary information to then provision the device. Actually, both unittype, unittype parameters, profile, unit and the secret unit parameter is 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 set discovery.force = true, then you will run the discover mode over and over again, always updating the list of unittype parameters. This can be useful if the CPE changes its list of parameters through a software update. This setting hurts the performance tremendously and should never ever be used in production.

If you have the Fusion Job Control Server module, you can use the job functionality of Fusion. In that case 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.

The syslog server address is only important if you're running TR-statistics. To setup such a job, you must run change and run certain scripts in Fusion Shell. Read more about this in [3].

Shell daemons are used whenever you make a job of SHELL type. This kind of job triggers the run of a shell script upon contact between device and server. Thus the TR-069 server must have a pool of shell daemons to run.

4.2.2 Quirks

A quirk is an adaptation of the server behaviour to fit a violation of the spec or to support and old version of the spec. These quirks are usually found during the interoperability phase run by Owera, so normally you shouldn't concern yourself with these details. 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 module.

4.2.3 Database

The database settings are fairly common, but the url setting could be complex, depending on which database you are connecting to. If connecting to MySQL we have used this kind of URL: jdbc:mysql://xaps-c.owera.com:3306/xaps and driver is com.mysql.jdbc.Driver. The database connection pool properties are pretty simple: decide the maximum number of database connections and decide how many milliseconds a connection can be in use.

5 Software and script download

5.1 Software download

To initiate a software download do the following:

1. Add the software file to Fusion. To do so use either Fusion Web or Fusion Shell to import the software file. You will be asked to enter the version number. Make absolutely sure that this version number is the same as the version number the software/CPE will report back once it is installed and applied on the CPE.
2. Changed the desired software version to point to the version number of the software file recently added to Fusion. The desired software version is set in this parameter: System.X_OWERA-COM.DesiredSoftwareVersion.

Caution: The desired software version MUST be the same string as reported from the CPE once the device has upgraded itself, if not the CPE will download the software again and again. The device reports the software version in the parameter InternetGatewayDevice.DeviceInfo.SoftwareVersion or Device.DeviceInfo.SoftwareVersion.

As always the settings can be done either on the unit, profile or job, but settings on unit level is usually not recommended.

If for some reason you do not want to serve the software file from Fusion you can decide to serve it from another location. To do so, just make the software available on a standard URL, and set the URL in this parameter: System.X_OWERA-COM.SoftwareURL. Make sure that the URL does not contain any thing like & or ?, since CPEs generally don't like these signs.

5.2 Script download

In TR-069 terminology a script is called "configuration". Such a "configuration" can be sent to a CPE and applied there. We believe that the term "script" is better, since the possibility for configuration download is to allow the CPEs to download a file (not a software image) and do whatever it likes with it. Many vendors see this as an opportunity to make available a large number of commands which currently is not supported in TR-069.

The concept of script download is very much the same as for software download, but with different system parameters.

```
Device-X.OWERA-COM.ScriptURL
Device-X.OWERA-COM.DesiredScriptVersion
```

5.3 Both software and config download

If both the DesiredSoftwareVersion and the DesiredScriptVersion is different from the parameters reported in the Inform, then we would have to do both of them. However, software download goes first. Next goes script download.

5.4 *Software upgrade wizard (Fusion 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.

5.5 *Software upgrade using Fusion Shell*

Everything you can do in Fusion Web, you can do in Fusion Shell. However, something is better in Fusion 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 Fusion 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 & Fusion 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 be STUN clients and be able to receive a special "trigger"-message from the STUN server. The Fusion STUN Server is of course in contact with Fusion, so that it will send those trigger messages whenever requested.

7.1 Installation

The installation of Fusion STUN Server is similar to all the other modules of Fusion: Just install the EAR file on a J2EE-server. In JBoss this translates into copying this xapsstun.ear file into the "deploy" directory and placing the property files into the "conf" directory.

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

# Fusion database connection
db.xaps = xaps/xaps@jdbc:mysql://localhost:3306/xaps
#db.xaps = xaps/xaps@jdbc:oracle:thin:@//localhost:1521/orcl

# 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
```

Set the primary IP and the db.xaps property, then you're ready to run.