Fusion SPP 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 SPP 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
1.1.1	M. Simonsen	29-Sep-10	Updated to latest version of Fusion SPP Server
1.3.1	M. Simonsen	12-Dec-11	Updated to latest version
1.4.8	M. Simonsen	20-Feb-13	Upaated 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
EAR	Enterprise Application Archive – represents a package which contains the whole server before deployment.
JEE	Java latform, Enterprise Edition

1.5 References

Document
[1] Fusion Product Description
[2] Fusion Installation
[3] Fusion Web 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 SPP Server is responsible for the communication with CPEs, and communicates over HTTP, HTTPS, TFTP and Telnet protocol. There are other servers in Fusion that also communicates with CPEs, but through other protocols (Syslog & TR-069).

The CPEs must be configured with a URL which will point to this server. When this is in place, the SPP 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.

2.1 CPE Interoperability

So far only Linksys SPA2102 and Ping Communication devices (NPA201E, RGW208EN, IAD208AN) has been run using SPP. However, the support for Linksys SPA2102 should guarantee a whole range of devices from Linksys/Cisco.

If other you want to run other devices in SPP, please contact Ping Communication for an interoperability test phase. It may very well turn out that a patch is needed for SPP to work, since these protocols are proprietary and will vary from one device vendor to another. However, since Linksys has made a public available document on HTTP/TFTP-provisioning, it could very well be that this "standard" has become relatively widespread. In that case the range of devices supported by SPP may be even greater than anticipated.

3 Propertyfiles

3.1 xaps-spp.properties

```
# *** Fusion Simple Provisioning Protocol Server (Fusion SPP) configuration file
***

# --- Parsing of vital information ---
# The request from a device usually contains vital and important information. You
# can configure how to parse & find this information - even though the device has
# never been provisioned by Fusion before. Currently Fusion will search for 4 types
of
# information:
#
# - serialnumber (a must for provisioning)
# - mac (a must for syslog)
# - softwareversion (a must for firmware upgrade)
# - modelname (a must for provisioning & discovery mode)
#
# Depending upon the request-protcol, we can search for information in various
# places:
#
# 1. In the request-filename - many devices are able to request a URL which
# contains a mac or serialnumber
# 2. In the request-headers - many devices provide information in the header
#(only HTTP)
# 3. In the request-parameters - many devices provide information in the
# request-parameters (only HTTP)
#
# To specify a search, create a property name and a pattern to search for.
# The property name is created like this (BNF-notation):
#
# propertyname = paramname.requestinfo
# paramname = mac | serialnumber | softwareversion | modelname
# requestinfo = requ | reqh | reqp
# requ = reqfile.index
# reqh = reqheader.header.index
# reqp = reqparam.param.index
# header = A request header name
# param = A request parameter name
# index = A unique index, made to separate two otherwise identical property names
#
# Another way of explaining the format is the reqexp-like syntax:
#
# mac|serialnumber|softwareversion|modelname.req(file|header.<header-name>|
param.<param-name>).<index>
#
# The pattern to search for is a reqexp-pattern. Remember to enclose the
# information your'e after in parenthesis (only one set of parenthesis in the
# pattern).
#
# Some examples which have been tested with SPA2102 (Linksys ATA) over HTTP
# mac.reqfile.1 = ([0-9a-fA-F:]+).cfg
# serialnumber.reqheader.user-agent.1 = \((\w+)\)
# softwareversion.reqheader.user-agent.1 = (\d+\.\d+\.\d+)
# modelname.reqheader.user-agent.1 = /(\w+-[^-]+)
#
# Some examples which have been tested with SPA2102 (Linksys ATA) over TFTP
# (filename example: /FM500JA31360/000E08172240/SPA-2102/5.2.10)
# serialnumber.reqfile.1 =    /([^/]+)/[^/]+/[^/]+/[^/]+^
# mac.reqfile.1 =             /[^/]+/([^/]+)/[^/]+/[^/]+^
# modelname.reqfile.1 =       /[^/]+/[^/]+/([^/]+)/[^/]+^
# softwareversion.reqfile.1 = /[^/]+/[^/]+/[^/]+/([^/]+)^

# You may specify as many patterns as you like, the server will try to match
# until it finds a hit for that particular information. Read the xaps-http.log
# (in debug-mode) carefully, it will print all headers/parameters and the filename
# (url), to help you find what you're searching for.
```

```
serialnumber.reqfile.1 =     /([^/]+)/[^/]+/[^/]+/[^/]+$
mac.reqfile.1 =              /[^/]+/([^/]+)/[^/]+/[^/]+$
modelname.reqfile.1 =        /[^/]+/[^/]+/([^/]+)/[^/]+$
softwareversion.reqfile.1 = /[^/]+/[^/]+/[^/]+/([^/]+)$
mac.reqfile.2 = ([0-9a-fA-F:]+).cfg
serialnumber.reqheader.user-agent.2 = \((\w+)\)
softwareversion.reqheader.user-agent.2 = (\d+\.\d+\.\d+)
modelname.reqheader.user-agent.2 = /(\w+-[^-]+)

# --- Provisioning output ----
# Each device may have it's own way of printing the provisioning data and we
# link a class responsible for producing the output to each modelname.
# Specify like this:
# <modelname>.output = <classname>
# The classes are described below
# Default class is SPA2102

# classname: SPA2102
# Fusion is currently tested with one device, the SPA2102 (a standard Linksys
# ATA adapter)
# According to documentation for this device, the SPA2102 uses the same protocol
# for provisioning as all the Sipura (SPA/Cisco/Linksys) products, so you may try
# this also for other SPA products that 2102.

SPA-2102.output = SPA
NPA201E.output = Pingcom

# -- Software upgrade output --
# A software upgrade is usually performed by returning a software-download-url
# to the client (CPE), and then allow the client to connect to this URL and
# download the binaries. For some clients this process is too cumbersome, and
# we want to allow a direct response in terms of binaries with no extra round-
# trips. In order to set up this, you need to specify a software upgrade job
# like usual. Like the property above, we use the modelname to the specify
# which type of upgrade output you prefer. Format:
# <modelname>.upgrade-output = Regular|Staging
# Regular is the normal behaviour, while Staging will give you a binaries directly
# if a software upgrade process.
HA01.upgrade-output = Staging

# --- Various controls ---

# Discovery will automatically create the unit in Fusion database.
# No parameters will be created; we have no information about the device upon
# contact. The device will be placed in the Default profile in the unittype
# corresponding to the modelname (if such can be found). Remember to specify a
# modelname pattern above. Discovery can be true or false. Default is false.
discovery = true

# 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 50. The number does not say anything about the KB used of the link,
# since this depends upon many factors.
concurrent.download.limit = 50


# ---- TFTP ----
# Decide the port to run the TFTP server. Default is 1069. (Default IANA standard
# is 69)
tftp.port = 1069

# --- Telnet ---
# Specify the maximum number of telnet clients allowed. If too many are
# allowed, it may cause problems for both the server (performance-wise)
# and the connection to your server
telnet.max-clients = 10

# 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 process is idle (no telnet-session active)
# Default is 60 minutes.
telnet.rescan = 60


# --- Database ---

# xAPS database connection
db.xaps.url = xaps/xaps@jdbc:mysql://localhost:3306/xaps
# Max connections. Default is 10.
db.xaps.maxconn = 40

# 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
```

3.1.1 Parsing of vital information

The configuration needed to set up parsing of vital information may seem very confusing at first. The reason for making such a configuration is simply that we don't know on beforehand how the device transmits its basic information to the server. Depending upon protocol and device, it may transmit information using the URL to the server, some HTTP-header or an HTTP-query string. Let's look at an example:

```
serialnumber.reqfile.1 =    /([^/]+)/[^/]+/[^/]+/[^/]+$
```

This example tells the server to parse the "reqfile" (short for request-file or URL) of the incoming request. Such a URL could be

http://xaps.pingcom.net/xapsspp/ABC/123/Product/1.0
In this case we parse and matches serialnumber = ABC. The parenthesis in the configuration denotes the "phrase" or "word" we're parsing for. The parsing format is (as usual) regular expression. This format is common for such purposes and not explained here.

It is possible to specify more than one way to search for serialnumber (or any other type of information), just index every parsing-command with a number.

All of this (and more) is explained in the property file it self.

3.1.2 Provisioning Output

Each device needs a particular format of the output. This is something we cannot fully configure, but we can connect a certain model name to an output formatter. The model name is retrieved in the parsing of input information. You need to contact Ping Communication to specify the correct output formatter. As a first attempt, always try using "SPA", since this is close to a de-facto standard in among HTTP/TFTP provisioning schemes.

3.1.3 Software upgrade output

In some very special cases, we can set the server to work with extreme simplicity: Upon connection to Fusion SPP server, the server can hand out a new firmware image which can be instantly download and applied on the device. This is a use case taken from Ping Communications Staging server, and may not apply elsewhere.

3.1.4 Various Controls

If you set discovery = true, new devices can be added to Fusion automatically (upon first contact). Their secret/serialnumber/mac is added to the database.

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.

3.1.5 TFTP

TFTP Server is a part of SPP module. You need to decide which port to run on.

3.1.6 Telnet

Telnet provisioning is described in [3]. Here you need to specify how often you will rescan groups participating in a job, to see if new devices should be on the "connect list". The max-clients properties specifies how many simultaneous connections to devices you will set up. The whole provisioning is controlled by starting a TELNET-job.

3.1.7 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.

4 Software and script download

When we say “software download”, we talk about a new firmware to the CPE. This is by far the most important feature of the system, so it is worth spending a few pages on this issue.

4.1 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 (for IAD) and Device.DeviceInfo.SoftwareVersion (for ATA).

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.

4.2 Both software and config download

If both the DesiredSoftwareVersion and the DesiredConfigVersion 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 config download.

4.3 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.

4.4 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.

5 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**).

5.1 *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.

5.2 *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.

5.3 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.