

FreeACS Fusion Shell User Manual

2014R1

Table of Contents

FreeACS Fusion Shell User Manual	1
2014R1	1
1 Document Introduction	7
1.1 Document Purpose and Audience	7
1.1.1 Purpose	7
1.1.2 Audience	7
1.2 Document History	8
1.3 Acronyms and Abbreviations	9
2 Introduction	10
2.1 A brief introduction to FreeACS concepts	11
3 Installation	12
4 Options	13
5 Connect & Configure	14
6 Command Introduction	15
6.1 Syntax and arguments	16
6.2 Contexts	17
6.3 File redirection	18
6.3.1 Advanced file redirection - input	18
6.3.2 Advanced file redirection – input/output	19
6.4 Piping	20
6.5 Scripts	21
6.5.1 Call	21
6.5.2 Variables	21
6.5.3 Return	22
6.5.4 Sleep	22
6.5.5 Echo	22
6.5.6 If/Else	23
6.5.7 While	24
6.6 File system commands	26
6.7 Logout/Exit	27
6.8 General help	28
6.9 Tab completion & command history	29

6.10	Important concepts/principles.....	30
7	Command reference	31
7.1	Generic menu	32
7.1.1	Generic.cc.....	32
7.1.2	Generic.unit	33
7.1.3	Generic.syslog	33
7.1.4	Generic.call	34
7.1.5	Generic.setvar	35
7.1.6	Generic.delvar	35
7.1.7	Generic.listvars.....	35
7.1.8	Generic.cat	35
7.1.9	Generic.delosfile	36
7.1.10	Generic.unittypeimport	36
7.1.11	Generic.unittypeexport	36
	Generic.unittypecompleteddelete	37
7.1.12	Generic.logout	37
7.1.13	Generic.dbinfo	37
7.1.14	Generic.userinfo	37
7.1.15	Generic.echo	37
7.1.16	Generic.exit	37
7.1.17	Generic.help.....	37
7.1.18	Generic.ls	38
7.1.19	Generic.sleep	38
7.1.20	Generic.pausescript.....	38
7.1.21	Generic.error.....	38
7.2	Root menu.....	39
7.2.1	Root.listunittypes	39
7.2.2	Root.listunits	39
7.2.3	Root.setunittype	40
7.2.4	Root.delunittype	40
7.2.5	Root.listparams.....	41
7.2.6	Root.listusers	41
7.2.7	Root.setuser.....	42
7.2.8	Root.deluser	42
7.2.9	Root.listperms	43

7.2.10	Root.setperm	43
7.2.11	Root.delperm.....	43
7.2.12	Root.listcertificates.....	44
7.2.13	Root.setcertificate	44
7.2.14	Root.delcertificate	44
7.3	Unittype menu	46
7.3.1	Unittype.listparams	46
7.3.2	Unittype.setparam	47
7.3.3	Unittype.delparam	48
7.3.4	Unittype.listunits.....	48
7.3.5	Unittype.moveunit	49
7.3.6	Unittype.systemparameterscleanup.....	49
7.3.7	Unittype.listfiles	49
7.3.8	Unittype.importfile	50
7.3.9	Unittype.exportfile	51
7.3.10	Unittype.delfile	51
7.3.11	Unittype.listprofiles	51
7.3.12	Unittype.setprofile.....	52
7.3.13	Unittype.delprofile.....	52
7.3.14	Unittype.listgroups	52
7.3.15	Unittype.setgroup	53
7.3.16	Unittype.delgroup	54
7.3.17	Unittype.listjobs.....	54
7.3.18	Unittype.setjob	55
7.3.19	Unittype.deljob	56
7.3.20	Unittype.listsyslogevents	56
7.3.21	Unittype.setsyslogevent.....	56
7.3.22	Unittype.delsyslogevent.....	57
7.3.23	Unittype.listexecutions.....	58
7.3.24	Unittype.setexecution	58
7.3.25	Unittype.generatetc.....	58
7.3.26	Unittype.deltcduplicates.....	59
7.3.27	Unittype.listtc	59
7.3.28	Unittype.showtc.....	59
7.3.29	Unittype.deltc.....	59

7.3.30	Unitttype.exporttcfile	60
7.3.31	Unitttype.exporttcdir	60
7.3.32	Unitttype.importtcfile	60
7.3.33	Unitttype.importtcdir	60
7.3.34	Unitttype.listtesthistory	61
7.3.35	Unitttype.deltesthistory	61
7.4	UnitttypeParameter menu	62
7.4.1	UnitttypeParameter.listvalues	62
7.4.2	UnitttypeParameter.setvalues	62
7.4.3	UnitttypeParameter.delvalues	62
7.4.4	UnitttypeParameter.generateenum	63
7.5	Profile menu	64
7.5.1	Profile.listparams	64
7.5.2	Profile.setparam	64
7.5.3	Profile.delparam	65
7.5.4	Profile.listunits	65
7.5.5	Profile.setunit	66
7.5.6	Profile.delunit	66
7.5.7	Profile.delallunits	66
7.5.8	Profile.moveunit	67
7.6	Unit menu	68
7.6.1	Unit.listallparams	68
7.6.2	Unit.listunitparams	69
7.6.3	Unit.setparam	69
7.6.4	Unit.delparam	70
7.6.5	Unit.kick	70
7.6.6	Unit.inspection	70
7.6.7	Unit.extraction	71
7.6.8	Unit.read	71
7.6.9	Unit.write	71
7.6.10	Unit.refresh	71
7.6.11	Unit.generatetc	71
7.6.12	Unit.deltcduplicates	72
7.6.13	Unit.listtc	72
7.6.14	Unit.showtc	72

7.6.15	Unit.deltc	72
7.6.16	Unit.exporttcfile	73
7.6.17	Unit.exporttcdir	73
7.6.18	Unit.importtcfile	73
7.6.19	Unit.importtcdir	73
7.6.20	Unit.testsetup	74
7.6.21	Unit.enabletest	74
7.6.22	Unit.disabletest	74
7.6.23	Unit.listtesthistory	74
7.6.24	Unit.deltesthistory	75
7.7	Group menu	76
7.7.1	Group.listparams	76
7.7.2	Group.setparam	76
7.7.3	Group.delparam	77
7.7.4	Group.delallparams	77
7.7.5	Group.count	77
7.7.6	Group.listunits	78
7.7.7	Group.listdetails	78
7.8	Job menu	79
7.8.1	Job.listdetails	79
7.8.2	Job.listparams	79
7.8.3	Job.listfailedunits	79
7.8.4	Job.delfailedunits	80
7.8.5	Job.status	80
7.8.6	Job.start	80
7.8.7	Job.pause	80
7.8.8	Job.finish	81
7.8.9	Job.setparam	81
7.8.10	Job.delparam	81
7.8.11	Job.refresh	81

1 Document Introduction

1.1 Document Purpose and Audience

1.1.1 Purpose

The purpose of the document is to explain what FreeACS Shell is and how to use it.

1.1.2 Audience

FreeACS Shell is not at all that intuitive like FreeACS Web. It is a command line interface and requires either

- ⑩ A determined user with some technical experience
- or
- ⑩ A technically experienced user

Users with experience from other shells (like unix-shells, telnet-interfaces, ftp etc) should be able to navigate the shell fairly quickly. Furthermore, recognizing regex is always a good thing. The not so experienced user will have a steep hill to climb – but the document aims to help him/her as well.

1.2 Document History

Version	Editor	Date	Changes
1.4.0	M. Simonsen	01-Mar-09	Initial public version.
1.4.1	M. Simonsen	23-Mar-09	Revised version
1.4.2	M. Simonsen	03-Apr-09	Revised edition
1.4.3	M. Simonsen	25-Jun-09	Revised edition
1.4.4	M. Simonsen	01-Oct-09	Revised edition
1.5.0	M. Simonsen	27-Sep-10	Revised edition
1.5.1	J. Hübenthal	19-Okt-10	Revised edition
1.5.3	M. Simonsen	24-Nov-10	Supports Job repeat/interval and script variables. Added monitoring scripts and commands. Added help. Major refactoring of document
1.5.4	M. Simonsen	07-Feb-11	Bugfix edition for 2011R1-SP1
1.7.3	M. Simonsen	25-May-11	2011R2 edition (added command piping, new options, syslog commands, inbuilt export/import command, ++
2.0.7	M. Simonsen	28-Dec-11	2012R1 edition, updated to latest commands (if/else/while/done)
2.3.18	M. Simonsen	13-Feb-13	2013R1 edition, updated to latest commands
2.3.40	M. Simonsen	20-Aug-14	Revision of this document – changed from Fusion to FreeACS Fusion

1.3 *Acronyms and Abbreviations*

Acronym	Explanation
Fusion	The previous (commercial) brand name of the product.
Module	The product consists of several independently running servers, which we often call "Module". The Modules may run on separate hosts in the network.
CPE	Customer Premises Equipment. Used in this document to refer to a single physical device. Same as the term "Device".
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).
Group	A set of matching criteria used to search for Units. Commonly referred to as Smart Group.
Job	Automates and controls changes to Units within a Group. Partitions the changes over time according to rules to limit network load.

2 Introduction

The Shell is a tool to help you understand, list, change, delete and add elements in the database. Furthermore it is a tool to help you automate tasks which are large or repetitive.

Previously, both xAPS and Fusion has been the brand names of FreeACS. The name change has not been completed throughout the system, hence there will be some references (in filenames, etc.) to "xaps" and "fusion". I hope that this will not confuse too much.

2.1 *A brief introduction to FreeACS concepts*

The system is based on an idea of Unit Types, Profiles and Units. These concepts are part of a hierarchy like this:

- the database can contain many Unit Types - each Unit Type represent a device model
- each Unit Type can contain many Profiles - a Profile is a group of Units
- each Profile can contain many Units -
- each Unit represents one physical device and contains data for one CPE/device

A Unit Type holds the information about which Parameters **can** be configured through FreeACS. A Profile holds the actual Parameter values, values that are similar for a whole group of Units. Correspondingly, the Unit holds the Parameter values that are special for each Unit (CPE/device).

That's it! To get a better understanding of FreeACS, read some of the reference documentation. This document will explain how to use Fusion Shell. As stated earlier the shell should help you to administer and manage the content of the database. The whole point is simply to be able to configure the CPEs in your network in the most efficient way, and the shell will prove to be one powerful tool to help you out in that task. But instead of explaining to you what the shell does it's better to show it.

3 Installation

The installation should be done by running the install script for FreeACS (download from freeacs.com/download).

The shell comes as a JAR file plus two property files. Place the JAR file somewhere of your choosing. Place xaps-shell.properties and xaps-shell-log4j.properties in the same directory. Run this command:

```
>java -jar shell.jar
```

Immediately the shell will ask you for some information, so if it does, "installation" is successful.

If you get this error message during startup:

```
Wed Dec 28 11:01:46 CET 2011 ERROR FileAppender.rollOnTime() LOG Have NOT rolled  
/opt/shell/fusion-shell.log
```

it is because your Linux user is not able to create files and/or directories in the /opt/xaps directory. Trying using "sudo" to overcome the problem. Another possibility is that two or more shell-sessions are running at the same time.

4 Options

To find out which options you have to run shell, run

```
java -jar shell.jar -help
```

This should reveal the short help page
Usage:

```
xapsshell
    Interactive session. Choose DB in application
xapsshell <LOGIN>
    Interactive session. DB chosen using command line arguments
xapsshell <COMMAND-OPTION> [<LOGIN>]
    Session runs according to command-option. May be interactive.
    DATABASE can be omitted only if using the script, help and ? command
    options.

COMMAND-OPTIONS (choose one):
  -script <filename>
    The shell will execute the commands in the file. The commands
    would be the same as used in an interactive session.
  -export ALL|<unitttype-name> [-dir <directory>]
    The shell will export ALL or a specific unit type to files. If you
    want to, you can specify a directory to place the files, otherwise
    it will place the files in a directory with the same name as the
    unitttype.
  -import ALL|<unitttype-name> [-dir <directory>]
    The shell will import ALL or a specific unit type from files. If you
    want to, you can specify a directory to read the files, otherwise
    it will read the files in a directory with the same name as the
    unitttype.
  -delete ALL|<unitttype-name>
    The shell will delete ALL or a specific unit type. Be careful.
  -help
    Shows this message
  -?
    Shows this message
  -<unrecognized option> <value>
    The shell will translate any unrecognized option into a variable.
    Option name will be the variable name and option will be the variable
    value.

LOGIN:
  -user <username>
    The database username, can be taken from xaps-shell.properties
  -password <password>
    The database password, can be taken from xaps-shell.properties
  -url <jdbc-url>
    The database url, can be taken from xaps-shell.properties.
  -fusionuser <username>
    The Fusion user, with a specified set of access permissions within
    FreeACS user database
  -fusionpass <password>
    The FreeACS password
```

5 Connect & Configure

If you choose the interactive database login options (from previous chapter), the first you'll see something similar to this:

```
0. Custom.  
1. xaps/xaps@jdbc:mysql://localhost:3306/xaps
```

The choices you see here are configured in xaps-shell.properties; reconfigure to correct user/password/URL.

Example of xaps-shell.properties:

```
# Available connections  
db.1 = xaps/xaps@jdbc:mysql://localhost:3306/xaps  
db.2 = myuser/mypass@jdbc:mysql://anotherhost:3306/xaps
```

You most definitely need to change the list of databases in xaps-shell.properties. Just try to follow the pattern <user>/<password>@<jdbc-url>. You can of course add more than one database, just increase the counter db.<counter> for each line. Do not skip a number.

If you choose custom (0) login, you will have to answer a series of question to establish the necessary information to log in to the database. Be prepared to enter the username, password and jdbc-url of the database, see example:

```
xapsdb-shell>0  
What is the url to the database?  
xapsdb-shell>jdbc:mysql://localhost:3306/xaps  
What is the username to the database?  
xapsdb-shell>user  
What is the password to the database?  
xapsdb-shell>password  
Wait a moment while some tables are loaded into memory.
```

After you've chosen the database to connect to, you may also specify a so-called "fusion username". If you decide to skip it, you can press ENTER and become admin. The reason for logging in with a specified fusion user is to reproduce a certain behaviour or because you run the shell in "restricted mode" (check xaps-shell.properties).

6 Command Introduction

The shell has over the past years grown into a simple (compared to shells like sh and bash), but yet powerful domain-specific script-language for FreeACS. It can do many kinds of tasks specific to FreeACS like adding a Unit or changing a Profile or starting a Job. This is in itself so useful that you would never ever consider doing a repetitive task in Web now that you are aware of the shell.

In order to make shell really useful some concepts more linked to regular script languages than to FreeACS itself are included. We'll discuss these concepts first, then move on to the specific commands to add, modify and delete contents of FreeACS.

6.1 Syntax and arguments

The syntax of this section goes like this:

Expression	Explanation
<something>	The whole argument should be replaced by the name of this "something"
	Logical "OR".
[<something>]	The "something" argument is optional
()	Used to group an expression
*	An expression can be repeated 0 or more times
Something	The argument should be exactly the string "Something"
=	Equal
!	Not equal
+	An expression can be repeated 1 or more times

Some examples to clarify:

Example	Can be written as
<weekday>	Monday
<A B>	A
A[<weekday>]	ATuesday
[A B] C	C
(A B)*	AAAB
Weekday	Weekday

Arguments will have to be enclosed in double quotes ("an argument") if it contains a space. Otherwise, it will be interpreted as two arguments.

6.2 Contexts

A very basic idea in the shell (and Web) is the idea of contexts. A context will tell the shell which Unit Type, Profile, Unit, Unit Parameter, Group or Job is currently being worked on. That way we do not have to specify all this information for each command that is invoked. The context are presented in the prompt like this:

```
/ut:Testperf/pr:Default/>
```

This prompt informs the user that Unit Type is "Testperf" and Profile is "Default". All commands entered in this context will be invoked on this Unit Type and Profile. Since the shell at all times know the context, the user can at any time enter "help" and expect a list of commands available in that specific context.

Even though the context in this example specifies both Unit Type and Profile, it makes most sense to think of this as a Profile context, since Profile is more "specific" than a Unit Type.

To switch between commands see command reference for "Generic.cc" or type "help cc".

6.3 File redirection

There are basically two types of commands: read & write. The read command produces output and the write command requires input. The shell exploits these characteristics of the commands by employing a simple principle for file redirection. Just like a unix-shell, by using ">", you redirect the output of a command to a file. If you want to append to an existing file, you may use ">>" for file redirection. Let's say you wanted to list all Unit Types, then you could send the output directly to a file like this:

```
/>listunittypes > output-of-list.txt  
/>
```

Now the output of the list command is sent to a file called output-of-list.txt. We could also do the opposite, using the "<", but only for write-commands. One typical example could be to add new Units to the system. Assume you have a file (units.txt) like this:

```
A  
B  
C  
D
```

Each letter now represents a Unit-id. We want to add this to the system as quickly as possible. To do so just run

```
/ut:Testperf/pr:Default/>setunit < units.txt  
[1] The unit was added  
[2] The unit was added  
[3] The unit was added  
[4] The unit was added  
/ut:Testperf/pr:Default/>
```

What happens here is that a write-command can iterate over a input-file and process one line at the time from the file. The file's content is simply added to the command before it is being processed, so the actual commands processed is really:

```
setunit A  
setunit B  
setunit C  
setunit D
```

At this point, it is possible to recognize that it can be fairly easy to add a lot of Units to the database.

6.3.1 Advanced file redirection - input

After coming to terms with file redirection in general there will soon be a need for a smarter way of using the file input than just appending the entire content at the end of the command (like the previous example). What we would like is to be able to place the file input arguments somewhere of our choosing. Going back to the previous example with the file "units.txt", we can now look on how to set a Parameter on each Unit:

```
/ut:Testperf/pr:Default/>un:${1}/setparam Testparameter "A value" < units.txt  
[1] The unit parameter is added.  
[2] The unit parameter is added.
```

```
[3] The unit parameter is added.  
[4] The unit parameter is added.  
/ut:Testperf/pr:Default/>
```

In this command we insert the input from the file at the beginning of the command by referencing the first (and only) column in the units.txt file. Since this input is a Unit id, the context is shifted, then "setparam" command is executed in the Unit context.

To further build on this example, let us assume that "units.txt" contains two columns, one with the Unit id and one with a Unit Parameter value:

```
A 1  
B 2  
C 3  
D 4
```

This file could then be used to set the value directly from file, instead of using a "static" value like in the previous example:

```
/ut:Testperf/pr:Default/>un:${1}/setparam Testparameter ${2} < units.txt  
[1] The unit parameter is changed.  
[2] The unit parameter is changed.  
[3] The unit parameter is changed.  
[4] The unit parameter is changed.  
/ut:Testperf/pr:Default/>
```

In this example arguments from the input file is extracted column by column (for each row). This will give a lot of flexibility when it comes to what kind of file input that can be used.

6.3.2 Advanced file redirection – input/output

It is also possible to do both input and output in one command. Assume we have a file with all names of the Unit Types in "unittypes.txt". We can now list all parameters in all those Unit Types and output to a file:

```
/>ut:${1}/listparams < unittypes.txt > params.txt
```

6.4 *Piping*

Once familiar with the file redirection feature, you may discover that you write to file for the **sole** purpose of using it as input in the next command. In that case you can avoid it all together by using pipe (the character is '|'). Using pipe you can send the output of one command into the next and then quickly build complex commands. This is the same feature as available in standard unix-shell. One example could be:

```
/>listunittypes | ut:${1}/listparams Comment
```

This example shows a listing of all parameters having the name "Comment" for all unittypes. The \${1} is used the same way as for file redirection.

A further enhancement of this command might be

```
/>listunittypes | ut:${1} listparams -c MAC | listunits -u ${1} = %10AE
```

This command will search all unittypes and all parameters named something with "MAC" with the value '10AE' preceded by anything. Notice the introduction of the options "-c" and "-u". The option "-c" will make the output from this command also list the context. The option "-u" will let the command use any context information from the input of the previous command/file-input.

6.5 Scripts

The absolute most useful thing about the shell is the ability to make a script. A script is nothing more than a file with shell commands. These commands should be entered into the script file just as you would enter the commands directly to the shell, even the login commands. You can of course run a script at any time. Such a script can be used to set up a whole range of Units or an entire Unit Type or modify all Profiles, etc. Sometimes you write a script for one occasion, other times you write scripts to use them over and over.

6.5.1 Call

To run a script there are various possibilities – one is already mentioned in the Option chapter: Make a script, then trigger it using the script option. Another possibility is to use the script-command "call". If the script is called "test.xss", then this command will trigger it:

```
/>call test.xss [-c<context>] [-v<variables>]
```

It is possible to make a script which calls upon another script (and so forth). What for? The reasoning is that it can be useful to make small reusable "scriptlets" which can be combined into a larger "master-script" which is custom made.

The context argument is optional and defaults to "the same context as for the callee". In other words: If no context is supplied to the new script it will start with the same context. However, sometimes it can be useful to specify another context for the new script. Specify the context in the same way as in the prompt (ut:MYUNITTYPE/pr:Default/un:MYUNIT).

The variables argument is a comma-separated list of variable names or values that you wish to pass to the new script. A variable name that is known in the callee will be referenced the same way in the called script. Example could be \${a}. If the variable name is not known, it will be passed as a value and will be referenced \${_1}, \${_2}, etc... in the called script.

6.5.2 Variables

A variable can be created using the "setvar" command, like this:

```
/>setvar <variable-name> <expression>
```

To display it you can use this command:

```
/>echo ${<variable-name>}
```

The expression is interpreted and evaluated using Javascript. This is a major benefit for this shell, since all kinds of expression may be evaluated upon setting the variable. However, to assign a string to a variable, you must enclose it with simple quotes ('). The setvar command is also allowed to run without the set-prefix, for better readability in scripts.

Examples:

```
/>var a 'Hello World'
/>echo ${a}
"Hello World"
/>var b 1 + 2
/>echo ${b}
3.0
```

```
/>
```

If you are inside a Unitttype context (that translates to every context except Root context), you may reference a parameter using a variable in this form:

```
/>${_parameter_name}
```

If such a parameter is defined in the Unit Type, the variable will contain the value of that a parameter. If you are in a Unit context, and that parameter happens not to be specified, the value of the variable will be NULL.

Examples:

```
/ut:N/pr:D/un:8/>echo ${ System.X OWERA-COM.Secret}  
5546084FCD4B8E37  
/ut:N/pr:D/un:8/>echo ${ System.X OWERA-COM.Telnet.Username}  
NULL  
/ut:N/pr:D/un:8/>echo ${ System.X OWERA-COM.Telnet.UsernameX}  
System.X_OWERA-COM.Telnet.UsernameX  
/ut:N/pr:D/un:8/>
```

If a regular variable is not resolved to any known variable name or parameter name, the variable name is simply echoed.

You can also access the context information using special variable names:

```
/ut:MYUNITTYPE/pr:Default/un:MYUNIT/>echo ${ ut} ${ pr} ${ un}  
MYUNITTYPE Default MYUNIT  
/ut:MYUNITTYPE/pr:Default/un:MYUNIT/>
```

The special variables are `_ut` (unitttype), `_pr` (profile), `_un` (unit), `_gr` (group), `_jo` (job), `_up` (unitttype parameter) and `_return` (return value from a script).

6.5.3 Return

```
/>return [<value>]
```

The command will return from the script, even if there are more to process in the script. The command is useful for testing of script, or to reuse parts of scripts. Upon return a value may be sent. This value can only be read using a special variable: `${_return}`.

6.5.4 Sleep

```
/>sleep [<number-of-seconds>]
```

The command will halt the script for as many seconds as specified, default is 60. The command is useful to provoke changes in FreeACS a timed manner, usually to test something, but could perhaps also be used to make a gradual change over a large population of objects.

6.5.5 Echo

```
/>echo [<message>|on|off]
```

The command will just print without any error message. You may write whatever you want after the echo command – maybe a warning or a message. If you specify "echo off", your scripts will run without showing all the commands executed. Set to on after finishing the scripts.

6.5.6 If/Else

From version 2.0.0, if/else has been added, which makes the scripts even more flexible. The syntax is

```
if <expression>
elseif <expression>
else
fi
```

You may repeat "elseif" 0 or more times, as you would expect. "else" is also optional. An expression is interpreted and evaluated by a Javascript-engine, so you will have the same capabilities here as you have in regular Javascript programming. Let's look at an example:

In this example we are looking for a particular parameter called ConnectionRequestURL with the format/value mentioned in the script-comment. Then we want to extract the IP-address of that string and return from the script.

```
# We assume a unit-context, where we can find the ConnectionRequestURL
# with a value like this:
# http://192.168.100.10:9699/conn req url

var param ' InternetGatewayDevice.ManagementServer.ConnectionRequestURL'
if '${param}' eq 'NULL'
    return 'NULL'
elseif '${param}'.substr(5,6) eq 's'
    var ip '${param}'.substr(8)
else
    var ip '${param}'.substr(7)
fi
var ip '${ip}'.substr(0, '${ip}'.indexOf(':'))
return '${ip}'
```

First we assign a value to a variable called "param". The main reason for doing so, is simply readability. In the first if-statement we test for equality between this variable and 'NULL'. Remember (from the variable sub-chapter) that if a variable name matches a parameter name, the script will retrieve the value of that parameter. In case that parameter has no value, the script will return 'NULL'.

If the parameter actually has a value, the script uses the Javascript-command "substr" to check if the URL starts with "https" or "http". Then the "substr" command is used to assign a value into the variable \${ip}. Lastly the ip-variable is manipulated using the Javascript command "indexOf" and the IP-address is found and returned.

Because > and < are use for file redirection, we have been forced to exchange all comparison operators in Javascript with a literal substitutes. Thus "eq" is used instead of "=". Here is the list of all comparison operators:

New operator	Textual	Old operator
lt	Less Than	<
le	Less Than Or Equal	<=
eq	Equal	=
ne	Not Equal	!=
ge	Greater Than Or Equal	>=
gt	Greater Than	>

Additionally || cannot be used (already used for piping). Therefore, this must be replaced with "or".

6.5.7 While

Another construct as of version 2.0.0 is the while/done. This is the only loop-construct of this shell, and for-loops or do-while loops has to be constructed using while-loops. The syntax of the loop is

```
while <expression>
...<commands>...
done
```

The simplest expression in a while loop is this

```
while true
echo Eternal loop
done
```

This kind of loop doesn't make much sense without the break-command, which allows the loop to be terminated.

```
while true
echo One loop
break
done
```

Combining the break statement with an if statement which turns true after a certain number of loops, can make some sense to this kind of loop. In that case we're very close to a regular for-loop.


```
echo off
var index 0
while ${index} lt 10
    var index ${index} + 1
    echo Loop number ${index}
done
echo on
```

This loop will go 10 rounds before exiting as is similar to a for-loop construct. Another very useful expression allowed by this shell is to run while over a file:

```
while foo.txt
    echo ${1} ${2}
done
```

The while loop will read line by line from the file, and each argument in this line will be available using \${1} for the first argument, \${2} for the second, etc.

Lastly, the command "continue" can be used to start the loop over, before coming to the end:

```
echo off
var index 0
while true
    var index ${index} + 1
    if ${index} le 10
        continue
    fi
    echo Loop number ${index}
    if ${index} ge 20
        break
    fi
done
echo on
```

6.6 *File system commands*

Since the script files are found on the file system, the shell provides a few very simple file system commands to browse the file system.

```
/>cat <filename>
```

Lists the content of the file specified. You should always use / as the file separator, it will work even if you run in a Windows environment.

```
/>ls [<directory>]
```

Lists the content of the directory. Default is the current directory (.). None of the options of ls are supported, but the output is similar to that of ls -l, but also influenced by the DOS command "dir".

```
/>delosfile <filename>
```

A command provided to make some cleanup (since it's quite easy to create a lot of files with the file redirection feature).

It is not a goal to provide a transparent shell which provides access to native commands from either a linux/unix/dos shell – to do so would be to break the platform compatibility of this shell. Instead we aim to provide a tiny subset of commands which is just enough to browse files.

6.7 *Logout/Exit*

You can change which database to work on, without exiting the shell. Just enter this command:

```
/>logout
```

Then you will get the same list of choices as at startup of the shell.

To completely exit the shell, run 'exit' command.

6.8 General help

When you log in you will see the “general help” of FreeACS Fusion Shell. That is a kind of short summary of how to work with the commands available in the shell:

[illegible]

The most interesting thing about this is just these two commands: help and exit. Help has already been mentioned, but it should be repeated: whenever assistance is needed, enter "help". There will most certainly be many times when the number of arguments needed for a command is forgotten. The detailed help for each command will also provide examples and comments explaining the particular command. As mentioned earlier the help is context sensitive. However, all the commands listed in from chapter 6.4 and onwards are generic commands. These can be invoked everywhere, and are also listed in each help.

The exit command will simply quit the shell. A script doesn't need an exit command – but if you use it – expect the termination of the shell even if it's a script called by another script. Instead use return if it's the termination of that specific script which is required.

6.9 *Tab completion & command history*

Shell now supports tab completion. The tab completion works to fill out long parameter names and commands. Since some terminals struggle to handle backspace/delete, this feature is very helpful and time saving.

Shell also supports command history, just hit arrow up key to re-run old commands. You can dump the command history by the command 'history' or a single '!' and re-run a command by using !<history-number>:

```
/>ls
...Some output...
/>!1 (will run 'ls' again)
```

6.10 *Important concepts/principles*

The set commands will do both add and change. The commands are designed to be run twice with the same input, something which is essential for scripts.

The deletion commands will offer some level of cascade deletion, but not a complete cascade. The principle is that when deleting a unit, all unit parameters will be deleted as well, when deleting a profile, all profile parameter will be deleted, and so on. The reason for not offering full cascade is that it should not be easy to delete too much. Take a look at the -delete option in chapter 4.

7 Command reference

The next chapters contains all the help text found in FreeACS Shell. This is mostly for reference, but it can be beneficial to quickly skim through the commands, to get an idea of the richness of the commands available. You should browse through the chapter, to see which commands are available.

7.1 *Generic menu*

List of commands available:

- ⑩ cc
- ⑩ unit
- ⑩ syslog
- ⑩ call
- ⑩ setvar
- ⑩ delvar
- ⑩ listvars
- ⑩ cat
- ⑩ delosfile
- ⑩ unittypeimport
- ⑩ unittypeexport
- ⑩ unittypecompleteddelete
- ⑩ logout
- ⑩ dbinfo
- ⑩ userinfo
- ⑩ echo
- ⑩ exit
- ⑩ help
- ⑩ ls
- ⑩ sleep
- ⑩ pausescript
- ⑩ error

7.1.1 **Generic.cc**

Syntax: cc <context>

Comment: Change to another context. The context change is fairly robust, so you can make context changes directly from, say a unit context to a group context, if only the group specified is within the same unit type.

Arguments:

- ⑩ **<context>**: Specify context to change to.
Syntax: .. or /(<type>:<contextname>/*

Use .. to go one step back in the context hierarchy.. Types are:

- 'ut' (Unit Type)
- 'pr' (Profile)
- 'un' (Unit)
- 'gr' (Group)
- 'jo' (Job)
- 'up' (Unit Type Parameter)

Use a single '/' to specify root. If context name contains whitespace, enclose the entire context with quotes: "/ut:Test UT/".

Examples:

- ⑩ cc ..
- ⑩ cc /
- ⑩ cc /ut:TestUnittype/pr:TestProfile
- ⑩ cc "/ut:Test Unit Type/"
- ⑩ cc /gr:MyGroup

7.1.2 Generic.unit

Syntax: unit [<search-value>]

Comment: Switch context to a particular unit. If more than 1 unit is found, all the units are listed. If within a unit type context, all parameters with the D(isplayable) flag are shown.

Arguments:

- ⑩ **[<search-value>]:** Optional. Searches among all unit parameters values and unit-ids. Possible to use control characters to specify search more closely:
 - * (asterix) - 0 or more wildchars
 - _ (underscore) - 1 wildchar
 - ^ (circumflex) - start of string
 - \$ (dollar) - end of string
 - ! (exclamation) - negation of search

Examples:

- ⑩ unit
- ⑩ unit AABBC0123
- ⑩ unit "John Doe"

7.1.3 Generic.syslog

Syntax: syslog [<search-args>] [<list-args>]

Comment: Search syslog and list the results in various ways

Arguments:

- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[<search-args>]:** Optional. Default search is for everything the last 24h, but only return the 100 most recent entries. If search argument contains whitespace, enclose with double quotes. The search argument syntax is:
s(<option>)(,<option>)* Each option is described below:
 - ⑩ ts=<time> - The start time. There are two ways to specify the <time> option:
 - ⑩ \d+(m|h|d) - specify as a number of (m)in, (h)our or (d)ays ago.
 - ⑩ yyyyMMdd[-HH[mm]] - specify as a timestamp. Optional defaults to 0
 - ⑩ te=<time> - The end time. There are two ways to specify the <time> option:
 - ⑩ \d+(m|h|d) - specify as a number of (m)in, (h)our or (d)ays ago.
 - ⑩ yyyyMMdd[-HH[mm]] - specify as a timestamp. Optional defaults to 0
 - ⑩ If ts is not specified, it will be set to 1d before te.
 - ⑩ ev=\d+ - Event-id
 - ⑩ fn=.+ - Facility-name (ex: Device-Local<number>, Fusion TR069, Fusion Web, etc)
 - ⑩ fv=.+ - Facility-version (expr*)
 - ⑩ ip=.+ - An ip-address or part of an ip-address (expr*)

- ⑩ m=.+ - Part of the syslog message. May use % or _ as wildchar. (expr*)
- ⑩ s=(\d)+ - One or more severities. 7=DEBUG, 6=INFO, 5=NOTICE, 4=WARN, 3=ERROR
- ⑩ us=.+ - A user id (expr*)
- ⑩ r=\d+ - Max number of rows in the result. Default is 100
- ⑩ Some fields are marked with (expr*). These fields can be specified using the following syntax:
 - ⑩ word : matches messages which contain 'word'
 - ⑩ * : matches 0 or more characters
 - ⑩ _ : matches 1 character
 - ⑩ ^ : if used at beginning (allowed after !) of expression, matches expression only from start of message
 - ⑩ \$: if used at end of expression, matches expression only at end of message
 - ⑩ ! : if used at beginning of expression, will negate the matching
 - ⑩ | : split the search, so 'A|B' translates to 'A OR B', and '!A|B' translates to 'NOT A AND NOT B'
- ⑩ To search within a specific unittype, profile or unit, change to the appropriate context and search.
- ⑩ **[<list-args>]**: Optional. Default listing is timestamp, severity, facility, message, ipaddress, unitid, profile and unittype. The list argument syntax is: l(<column-type>)(,<column-type>)* Each column-type is described below:
 - t - Timestamp column
 - ev - Event id
 - fn - Facility name
 - fv - Facility version
 - ip - Ip address
 - m - Syslog message
 - s - Severity level
 - us - User name
 - un - Unit id
 - pr - Profile name
 - ut - Unit Type name

Examples:

- ⑩ syslog "sts=1d,m=Hello World,s=5,6,7,r=120"
- ⑩ syslog sts=20110515-12,ip=192,fv=4.2
- ⑩ syslog lun,t,fv,m

7.1.4 Generic.call

Syntax: call <scriptname>

Comment: A call to another script. Possible to pass context-information and variables.

Arguments:

- ⑩ **-u**: Specify context to run the script within. Default is to run in the same context as the callee.
 Syntax: -c/(<type>:<contextname>/)*
 Types are: 'ut' (Unit Type), 'pr' (Profile), 'un' (Unit), 'gr' (Group), 'jo' (Job) or 'up' (Unit Type Parameter). Use a single '/' to specify root. If context name contains whitespace, enclose the entire option with quotes: "-c/ut:Test UT/".
- ⑩ **-v**: Specify variables to pass to the script. The variables are passed using the variable name or a value

Syntax: -v(<var1>)(,<var2>)*

The variable names must be defined in the callee's context, if not the variable is passed as a value and is then referred to using `${_1}` for the first value, and `${_2}` for the second value etc..

Arguments:

- ⑩ **<scriptname>**: Can refer to scripts found in the filesystem or in Fusion. For referring to Fusion script files, the context cannot be root. In case a script has the same name in the filesystem and in Fusion, the Fusion script file is used. For filesystem scripts use / as directory path delimiter.

Examples:

- ⑩ call test.xss
- ⑩ call scripts/test.xss
- ⑩ call -c/ut:TestUnittype/pr:TestProfile/un:TestUnit/ test.xss
- ⑩ call -vFoo,Bar test.xss

7.1.5 Generic.setvar

Syntax: setvar <name> <expression>

Comment: Setting a variable which may be read in this session. The variables will not be passed to a new script when using call, then you need to use -v option in the call command. To reference a variable use `${name}`.

Arguments:

- ⑩ **<name>**: Any name, but cannot start with a number, or an underscore or contain whitespace.
- ⑩ **<expression>**: A Javascript expression. Use single quotes to specify a string value.

Examples:

- ⑩ setvar Foo 'Hello'
- ⑩ setvar Bar 'Hello World'
- ⑩ setvar '\${Bar}'+Test
- ⑩ setvar Sum 1 + 2

7.1.6 Generic.delvar

Syntax: delvar <name>

Comment: Delete a variable, the variable can no longer be accessed in this session.

Arguments:

- ⑩ **<name>**: The variable name

Examples:

- ⑩ delvar Foo

7.1.7 Generic.listvars

Syntax: listvars

Comment: List all variables defined in this session

7.1.8 Generic.cat

Syntax: cat <filename>

Comment: Prints the content of the filename to the screen.

Arguments:

- ⑩ **<filename>**: Can refer to files found in the filesystem or in Fusion. For referring to Fusion script files, the context cannot be root. In case a script has the same name in the filesystem and in Fusion, the Fusion script file is used. For filesystem files use / as directory path delimiter.

Examples:

- ⑩ cat test.xss
- ⑩ cat scrips/test.xss

7.1.9 Generic.delosfile

Syntax: delosfile <filename>

Comment: Deletes a file found in the os filesystem.

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<filename>**: Specify filename to delete. Use / as directory path delimiter

Examples:

- ⑩ delosfile test.log

7.1.10 Generic.unittypeimport

Syntax: unittypeimport <unittypename>|ALL

Comment: Import a unittype from a set of files previously exported using exportunittype command

Arguments:

- ⑩ **<unittypename>|ALL**: Specify the unittype name or ALL for all unittypes. In both cases the command expects a directory with this name to exist in working directory and it must contain the files exported from exportunittype command

Examples:

- ⑩ unittypeimport ALL
- ⑩ unittypeimport NPA201E

7.1.11 Generic.unittypeexport

Syntax: unittypeexport <unittypename>|ALL

Comment: Export a unittype to a set of files in a directory with the same name as the unittype

Arguments:

- ⑩ **<unittypename>|ALL**: Specify the unittype name or ALL for all unittypes. In both cases the command will create a directory with this name and store all the exported files there.

Examples:

- ⑩ unittypeexport ALL

⑩ unittypeexport NPA201E

Generic.unittypecompleteddelete

Syntax: unittypecompleteddelete <unittypename>|ALL

Comment: Delete a unittype or all unittypes.

Arguments:

⑩ <unittypename>|ALL: Specify the unittype name or ALL for all unittypes.

Examples:

- ⑩ unittypecompleteddelete ALL
- ⑩ unittypecompleteddelete NPA201E

7.1.12 Generic.logout

Syntax: logout

Comment: Logout of the shell, will prompt a new login

7.1.13 Generic.dbinfo

Syntax: dbinfo

Comment: Shows which db-user is used which database the shell is connected to

7.1.14 Generic.userinfo

Syntax: userinfo

Comment: Shows which fusion user is logged in

7.1.15 Generic.echo

Syntax: echo <text>

Comment: Echos the <text> argument to screen. Will evaluate any references to variables

Arguments:

⑩ <text>: Any text

Examples:

- ⑩ echo "Hello World"
- ⑩ echo \${VariableName}

7.1.16 Generic.exit

Syntax: exit

Comment: Exits Fusion Shell

7.1.17 Generic.help

Syntax: help [<command>|generic]

Comment: Helptext for the various commands.

Arguments:

- ⑩ [**<command>[generic]**]: Optional. If skipped help will show a list of commands available in this context. Use 'generic' to get list of generic commands. To get detailed help, specify a command name. There is no need to specify the whole command name, just specify enough to make the argument non-ambiguous.

Examples:

- ⑩ help
- ⑩ help generic
- ⑩ help cal
- ⑩ help <unitttype-name>

7.1.18 Generic.ls

Syntax: ls [<directory>]

Comment: Lists the content of the files in a directory

Arguments:

- ⑩ [**<directory>**]: Optional. Default is the working directory. Use / as directory path delimiter.

Examples:

- ⑩ ls
- ⑩ ls lib/.svn

7.1.19 Generic.sleep

Syntax: sleep [<seconds>]

Comment: Sleeps/halts the shell for a number of seconds.

Arguments:

- ⑩ [**<seconds>**]: Optional. Default is 60 seconds.

7.1.20 Generic.pausescript

Syntax: pausescript

Comment: Pause the script execution until user hits RETURN

7.1.21 Generic.error

Syntax: error <error-message>

Comment: Exit a script with an error message

7.2 *Root menu*

List of commands available:

- ⑩ listunittypes
- ⑩ listunits
- ⑩ setunittype
- ⑩ delunittype
- ⑩ listparams
- ⑩ listusers
- ⑩ setuser
- ⑩ deluser
- ⑩ listperms
- ⑩ setperm
- ⑩ delperm
- ⑩ listcertificates
- ⑩ setcertificate
- ⑩ delcertificate

7.2.1 **Root.listunittypes**

Syntax: listunittypes [<unittype-name-pattern>]

Comment: Lists the unittypes available in Fusion

Arguments:

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[<unittype-name-pattern>]:** Optional. Any string which will be used to match the list of unittype names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources.

Examples:

- ⑩ list
- ⑩ list NPA
- ⑩ list ^NPA
- ⑩ list NPA\d01

7.2.2 **Root.listunits**

Syntax: listunits [<search-value>]

Comment: List/search for units in Fusion

Arguments:

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[<search-value>]**: Optional. If this is the only argument, it will search among all unit parameters and unit-ids.

Examples:

- ⑩ listunits AABBC0123
- ⑩ listunits "John Doe"

7.2.3 Root.setunitttype

Syntax: setunitttype <unitttype-name> TR-069|N/A (<vendor> <description>)|<unitttype-file>

Comment: Add/change a unitttype. No parameters are added.

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<unitttype-name>**: Any string if adding. If changing it must be a valid unitttype name.
- ⑩ **TR-069|N/A**: Set to N/A for all other protocols (HTTP/TFTP) than TR-069.
- ⑩ **<vendor>**: Non-essential (to Fusion) information
- ⑩ **<description>**: Non-essential (to Fusion) information
- ⑩ **<unitttype-file>**: A file in the filesystem (cannot retrieve from Fusion filestore). Use / as directory path delimiter. The file must be a specially made xml-file with information on how to build a complete unitttype.

Examples:

- ⑩ setunitttype NPA201E TR-069 "Ping Communication" "Two port ATA"
- ⑩ setunitttype NPA201E TR-069 unittypedef/npa201e/npa201e-34707-master.xml

7.2.4 Root.delunitttype

Syntax: delunitttype <unitttype-name>

Comment: Delete a unitttype. The command will not succeed if profiles/groups/jobs/software/files exists in this unitttype. For a complete delete procedure, use the -delete option when executing xapsshell.jar

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<unittype-name>**: Must be a valid unittype name.

Examples:

- ⑩ delunittype NPA201E

7.2.5 Root.listparams

Syntax: listparams [staging]

Comment: Lists system parameters used in all unittypes.

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[staging]**: Optional. If set to 'staging' all system parameters regarding staging will be shown. Staging is a special concept, usually only for CPE vendors

Examples:

- ⑩ listparams
- ⑩ listparams staging

7.2.6 Root.listusers

Syntax: listusers [<username-pattern>]

Comment: List users in Fusion

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[<username-pattern>]**: Optional. Any string which will be used to match the list of users. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources.

Examples:

- ⑩ listusers
- ⑩ listusers admin
- ⑩ listusers comp.*

7.2.7 Root.setuser

Syntax: setuser <username> <fullname> <secret> <accesslist>|NULL <admin>

Comment: Add/change user.

Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<username>:** Any string if adding. If changing it must be a valid user name.
- ⑩ **<fullname>:** The full name of the user
- ⑩ **<secret>:** The password of the user. Will be stored in the database as a hash of the password. Thus if the password is forgotten, it cannot be retrieved. However, using this command it can be overwritten without any problem. The user/password information is needed to login to Fusion Web and Fusion Web Services.
- ⑩ **<accesslist>|NULL:** A comma-separated list of pages that the user can access in Fusion Web. The valid pages are
 - search - The search page
 - unit - The unit page
 - profile - The profile page
 - unittype - The unittype page
 - group - The group page
 - job - The job page
 - software - The software/firmware page
 - syslog - The syslog page
 - monitor - The monitor page - shows the status of Fusion
 - staging - A staging page, special case - only used for CPE vendors
 - report - The report pages - you need report certificate to view themSetting access to NULL will allow all pages!
- ⑩ **<admin>:** Can be set to "true" or "false". If set to "true" the user will have access to all unittypes and profiles regardless of any permissions added to this user later. If set to "false" the user will have no permissions unless explicitly given permissions (using setperm). The flag cannot be set to true unless the user running the command is itself an admin.

Examples:

- ⑩ setuser admin Administrator supersecret true
- ⑩ setuser support Support supersecret search,unit,profile false

7.2.8 Root.deluser

Syntax: deluser <username>

Comment: Delete a user, but requires that no permissions are attached to this user.

Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<username>:** Must be a valid username. If 'admin' user is deleted, a default 'admin' user will still exist in Fusion. Then it will have the password 'xaps'

Examples:

- ⑩ deluser support

7.2.9 Root.listperms

Syntax: listperms [<unittype-name-pattern>]

Comment: List permissions in Fusion. Each permission is attached to a user and controls which unittypes/profiles are allowed for that user. If a permission for a unittype is created with NULL as profile value, no other permissions for that unittype can be created.

Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

Arguments:

- ⑩ **[<unittype-name-pattern>]:** Optional. Any string which will be used to match the unittype of the permissions. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources.

Examples:

- ⑩ listperms
- ⑩ listperms NPA201E

7.2.10 Root.setperm

Syntax: setperm <username> <unittype-name> <profile-name>|NULL

Comment: Add a permission. Change not possible (have to delete first).

Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<username>:** A valid user name.
- ⑩ **<unittype-name>:** A valid unittype name
- ⑩ **<profile-name>|NULL:** If NULL, then all profiles within the unittype is permitted. Else only the specified profile is permitted.

Examples:

- ⑩ setperm support NPA201E NULL
- ⑩ setperm support NPA201E Default

7.2.11 Root.delperm

Syntax: delperm <username> <unittype-name> <profile-name>|NULL

Comment: Delete a permission.

Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<username>**: A permission user name.
- ⑩ **<unitttype-name>**: A valid permission unitttype name
- ⑩ **<profile-name>** | **NULL**: A valid permission profile name or NULL

Examples:

- ⑩ delpm support NPA201E NULL
- ⑩ delpm support NPA201E Default

7.2.12 Root.listcertificates

Syntax: listcertificates

Comment: List certificates in Fusion. Shows the complete certificate string in both encrypted version and clear-text

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

7.2.13 Root.setcertificate

Syntax: setcertificate <name> <certificate>

Comment: Add/change a certificate. Will not allow all kinds of changes (going from production to test certificate is not allowed)

Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

Arguments:

- ⑩ **<name>**: The name of the certificate is not important. Fusion will recognize the type no matter the name
- ⑩ **<certificate>**: A hexadecimal 256-char string which is the certificate. Certificates can only be retrieved from Ping Communication. Currently there are two certificates supported in Fusion: Report and Provisioning.

Examples:

- ```
10 setcertificate Test
E9D66AABBC012345E9D66AABBC012345E9D66AABBC012345E9D66AABBC01234
5E9D66AABBC012345E9D66AABBC012345E9D66AABBC012345E9D66AABBC0123
45E9D66AABBC012345E9D66AABBC012345E9D66AABBC012345E9D66AABBC012
345E9D66AABBC012345E9D66AABBC012345E9D66AABBC012345E9D66AABBC01
2345
```

### 7.2.14 Root.delcertificate

**Syntax:** delcertificate <name>

**Comment:** Delete a certificate. Make sure to have backed up the certificate string.

### Arguments:

⑩ **<name>**: A valid certificate name

**Examples:**

⑩ delcertificate Test

## 7.3 *Unittype menu*

List of commands available:

- ⑩ listparams
- ⑩ setparam
- ⑩ delparam
- ⑩ listunits
- ⑩ moveunit
- ⑩ systemparameterscleanup
- ⑩ listfiles
- ⑩ importfile
- ⑩ exportfile
- ⑩ delfile
- ⑩ listprofiles
- ⑩ setprofile
- ⑩ delprofile
- ⑩ listgroups
- ⑩ setgroup
- ⑩ delgroup
- ⑩ listjobs
- ⑩ setjob
- ⑩ deljob
- ⑩ listsyslogevents
- ⑩ setsyslogevent
- ⑩ delsyslogevent
- ⑩ enabletr069report
- ⑩ listexecutions
- ⑩ setexecution
- ⑩ generatetc
- ⑩ deltcuplicates
- ⑩ listtc
- ⑩ showtc
- ⑩ deltc
- ⑩ exporttcfile
- ⑩ exporttcdir
- ⑩ importtcfile
- ⑩ importtcdir
- ⑩ listtesthistory
- ⑩ deltesthistory

### 7.3.1 **Unittype.listparams**

**Syntax:** listparams [<unittype-parameter-name-pattern>]

**Comment:** List unittype parameters.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.

- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **[!]<unittype-parameter-name-pattern>**: Optional. Any string which will be used to match the list of unittype-parameters. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

#### Examples:

- ⑩ listparams
- ⑩ listparams DeviceInfo
- ⑩ listparams ^Device.Device.\*

### 7.3.2 Unittypesetparam

**Syntax:** setparam <unittype-parameter-name> [D][A][C][S][I][M](R|RW|X)

**Comment:** Add/change a unittype parameter.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<unittype-parameter-name>**: If adding: A name without spaces. It is a good idea to follow the TR-069 dotted pattern for parameter names. If the device is a TR-069 device, make sure that parameter names are equal to those registered in the device. Max length: 255 characters. Tip: Fusion TR-069 Server, discovery mode. If changing: A valid unittype parameter name.
- ⑩ **<flags>**: The flags D,A,C,S,I,M are optional, R, RW or X is mandatory. The order of the flags are not important.
  - [D]isplayable - The value will be shown as a column in the search page of Fusion Web. Cannot be combined with I and C.
  - [A]lwaysRead - The value will always be read from the device. Cannot be combined with I, RW or X.
  - [C]onfidential - The value will probably be sent from the device as "", usually used for passwords. This flag will suppress a warning that is logged when database and device differ while it was expected to be equal. The logs will obscure the value, to minimize exposure.
  - [S]earchable - Possible to search for this particular parameter in Fusion Web, using the Advanced Search.
  - [I]nspection - The value will only be read from the device in inspection mode. This mode can be activated in Fusion Web in the Unit Parameter page. After inspection mode is finished, the inspection parameters read will be deleted. Useful for short-lived parameters.
  - [M]onitor - The value will be sent to Fusion Syslog Server, and can be processed into a TR-report (given the appropriate parameters are set to M). The parameters are stored to database only if combined with A flag.
  - (R)eadOnly - The parameter can only be read from the device, never written to the device.
  - (R)ead(W)rite - The parameter can be written to the device. Only read from

device if combined with I.

(X) system - The parameter is only for internal use, never sent to the device. You can create as many X parameters as you like. Please adhere to standard name convention for such parameters:

System.X\_COMPAMY-COM.NameOfParameter<.SubName>

#### Examples:

- ⑩ setparam Device.DeviceInfo.PeriodicInformInterval RW
- ⑩ setparam System.X\_OWERA-COM.Comment SDX
- ⑩ setparam Device.DeviceInfo.SerialNumber AR

### 7.3.3 Unitttype.delparam

**Syntax:** delparam <unitttype-parameter-name>

**Comment:** Delete a unitttype parameter. Cannot be done unless the parameter is not used in any unit, profile, job, group.

#### Arguments:

- ⑩ **<unitttype-parameter-name>**: A valid unitttype parameter name
- ⑩ **<unitttype-parameter-name>**: A valid unitttype parameter name.
- ⑩ **<operator>**: Valid operators are EQ (equals), NE (not equals), LT (less), LE (less or equal), GE (greater or equal), GT (greater).
- ⑩ **<value>**: Any string. Max 255 characters. Long values will take more time to process in the event of a group search, so try to avoid if possible. The rules for matching are the same as for the 'listunits' commands in the unitttype/profile context. In other words, use '%' in the value string to match 0 or more arbitrary characters and use '\_' in the value string to match 1 arbitrary character. Avoid % and \_ at the start of the value. To search for NULL-values, use the string NULL.
- ⑩ **[<datatype>]**: Optional. Default datatype is VARCHAR (which is the same as 'text' or 'string'). You can also use SIGNED for an integer which might be negative, or UNSIGNED for integers which are always positive.

#### Examples:

- ⑩ delparam Device.DeviceInfo.PeriodicInformInterval

### 7.3.4 Unitttype.listunits

**Syntax:** listunits [(<search-value>) | (<unitttype-parameter-name> <operator> <value> [<datatype>])\*]

**Comment:** List units within this unitttype.

#### Arguments:

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabatically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:



- ⑩ [**<search-value>**]: Optional. It will search among all parameter values and unit-ids
- ⑩ **<operator>**: Valid operators are EQ (equals), NE (not equals), LT (less), LE (less or equal), GE (greater or equal), GT (greater).
- ⑩ **<value>**: Any string. Max 255 characters. Long values will take more time to process in the event of a group search, so try to avoid if possible. The rules for matching are the same as for the 'listunits' commands in the unittype/profile context. In other words, use '%' in the value string to match 0 or more arbitrary characters and use '\_' in the value string to match 1 arbitrary character. Avoid % and \_ at the start of the value. To search for NULL-values, use the string NULL.
- ⑩ [**<datatype>**]: Optional. Default datatype is VARCHAR (which is the same as 'text' or 'string'). You can also use SIGNED for an integer which might be negative, or UNSIGNED for integers which are always positive.

**Examples:**

- ⑩ listunits
- ⑩ listunits John
- ⑩ listunits Device.DeviceInfo.SerialNumber EQ AABBBCCDDEEFF00
- ⑩ listunits Device.DeviceInfo.SerialNumber NE %AABBCC%

### 7.3.5 Unittype.moveunit

**Syntax:** moveunit <unitid> <unittypename> <profilename>

**Comment:** Move a unit to another unittype. Requires that the target unittype has defined all unittype parameters found for this unit's parameters and that the profile exists.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<unitid>**: A valid unit-id in this unittype.
- ⑩ **<unittypename>**: Target unittype name, must exist.
- ⑩ **<profilename>**: Target profile name, must exist.

**Examples:**

- ⑩ moveunit 123456-ProductClass-123456789012 NPA201E-Test Default

### 7.3.6 Unittype.systemparameterscleanup

**Syntax:** systemparameterscleanup

**Comment:** Cleanup of old and unused system parameters. Will never delete a system parameter that is in use.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

### 7.3.7 Unittype.listfiles

**Syntax:** listfiles [<file-name-pattern>]

**Comment:** List files stored in Fusion.

**Arguments:**

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **[![<file-name-pattern>]**: Optional. Any string which will be used to match the list of file-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

#### Examples:

- ⑩ listfiles
- ⑩ listfiles 5.4.1

## 7.3.8 Unittype.importfile

**Syntax:** importfile <filename> <filetype> <version> <timestamp> <targetname> <description>

**Comment:** Add/import file to Fusion filestore.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<filename>**: Filename of the file (found in the filesystem). Use / as directory path delimiter.
- ⑩ **<filetype>**: SOFTWARE - CPE image files, used in software upgrade  
 TR069\_SCRIPT - A CPE script/config file, used in TR-069 Script Upgrade. A Target filename must be specified  
 SHELL\_SCRIPT - A Fusion script file, used in Fusion Shell, Shell Jobs, Triggers and Syslog Events  
 TELNET\_SCRIPT - A Telnet script file, used in Telnet Job  
 UNITS - A unit file, first column in file must be a valid unit-id, used in Trigger Script execution  
 MISC - A file, no particular use-case
- ⑩ **<version>**: Extremely important if importing a SOFTWARE or a TR069\_SCRIPT. It MUST be the version number reported from the software/script when installed in the device. Otherwise it will create a download loop! For other file types this is not as important.
- ⑩ **timestamp**: Specify a timestamp (format is : yyyyMMdd-HH:mm:ss) or set to NULL (which will translate into current timestamp)
- ⑩ **targetname**: Optional (set to NULL) for all file types except TR069\_SCRIPT, in which case the targetname is used to distinguish between multiple Script Upgrade.
- ⑩ **owner**: Owner of file. Can only be set if you're admin, if not it will automatically (and silently) be set to your username

- ⑩ **<description>**: Non-essential (to Fusion) description of the file.

**Examples:**

- ⑩ `importfile test.fss SHELL_SCRIPT 1 NULL NULL NULL "A Shell Script"`
- ⑩ `importfile firmware.bin SOFTWARE 5.0.3-alfa NULL NULL myuser "A firmware for the NPA201E"`
- ⑩ `importfile voip.zip TR069_SCRIPT 1 NULL /opt/voip.zip NULL "A Zip-file download to the CPE"`

### 7.3.9 Unitttype.exportfile

**Syntax:** `exportfile <filename>`

**Comment:** Export file from Fusion filestore.

**Arguments:**

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<filename>**: A filename from Fusion filestore. If the file already exists it will create a directory named after the unitttype, and place the file there. The import command will always search for such a location first when trying to import a file.

**Examples:**

- ⑩ `exportfile test.xss`

### 7.3.10 Unitttype.delfile

**Syntax:** `delfile <filename>`

**Comment:** Delete a file from Fusion filestore

**Arguments:**

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<filename>**: A valid filename in the Fusion filestore

**Examples:**

- ⑩ `delfile test.xss`

### 7.3.11 Unitttype.listprofiles

**Syntax:** `listprofiles [<profile-name-pattern>]`

**Comment:** List profiles within this unitttype

**Arguments:**

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabatically in an (a)scending order,

next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[!]<profile-name-pattern>]:** Optional. Any string which will be used to match the list of profile-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

**Examples:**

- ⑩ listprofiles
- ⑩ listprofiles Def

### 7.3.12 Unitttype.setprofile

**Syntax:** setprofile <profilename>

**Comment:** Add/change a profile. Does not add/change any profile parameters.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<profilename>:** Any string, enclose arguments with spaces using double quotes.

**Examples:**

- ⑩ setprofile Default

### 7.3.13 Unitttype.delprofile

**Syntax:** delprofile <profilename>

**Comment:** Delete a profile. Not allowed if there are units within this profile or if groups are connected to this profile.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<profilename>:** A valid profile within this unitttype.

**Examples:**

- ⑩ delprofile Default

### 7.3.14 Unitttype.listgroups

**Syntax:** listgroups [<group-name-pattern>] [PARENT-FIRST|PARENT-LAST]

**Comment:** List groups within this unitttype.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **[**!**<group-name-pattern>]:** Optional. Any string which will be used to match the list of group-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.
- ⑩ **[**PARENT-FIRST|PARENT-LAST**]:** Defines the ordering of the list. This is usually not interesting - mostly used for -export/import options of this shell

#### Examples:

- ⑩ listgroups
- ⑩ listgroups "All units"

### 7.3.15 Unittyp.setgroup

**Syntax:** setgroup <groupname> <parent-groupname>|NULL <description> <profile-name>|NULL <time-param-name>|NULL <time-format>|NULL <time-offset>|NULL

**Comment:** Add/change a group.

#### Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<groupname>:** Adding: Any string. Enclose with double quotes if it contains spaces. Changing: Valid group name.
- ⑩ **<parent-groupname>|NULL:** A group can be a child of another group. In that case it will inherit the group parameters from the parent group. The group parameters will then be ANDed together to form one or more criteria for the group search.
- ⑩ **<description>:** A non-essential (for Fusion) description of the group.
- ⑩ **<profile-name>|NULL:** Will restrict the search performed by this group till this profile only. If a parent group is specified, the profile setting of that group will override this.
- ⑩ **<time-param-name>|NULL:** If you like to create a time-rolling group (other terms might be 'time' and 'monitor' groups), you must specify a time parameter. The effect of this is that Fusion Core Server will change this group's time parameter criteria on a periodic interval, which in turn is extremely useful for a special Syslog Event copying group parameters to a unit. The whole idea is to be able to monitor a certain event as it varies over time. In that case, specify a valid unit type parameter. Please specify a System parameter name, run 'help setparam' and read about the X flag and the name convention. Otherwise set it to NULL
- ⑩ **<time-format>|NULL:** The time-format can be any string, but some characters (and group of characters) take on a special meaning. A format like 'yyyyMMddHH' will populate (through Fusion Core Server) the time parameter with something like

'2010111814'. The formatting rules are described in SimpleDateFormat (search on the internet). If time parameter is set and this format set to NULL, the default value will be 'yyyyMMdd'.

- ⑩ **<time-offset>|NULL**: When Fusion Core Server sets this value, it will subtract the offset (seconds) from current time. If timeparameter is set and this offset set to NULL, the default value will be 0.

#### Examples:

- ⑩ setgroup Test NULL Test NULL NULL NULL NULL
- ⑩ setgroup SubTest Test SubTest NULL System.X\_COMPANY-COM.SIPFailed  
yyyyMMdd 0

### 7.3.16 Unitttype.delgroup

**Syntax:** delgroup <groupname>

**Comment:** Delete a group

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<groupname>**: A valid groupname

#### Examples:

- ⑩ delgroup Test

### 7.3.17 Unitttype.listjobs

**Syntax:** listjobs [<job-name-pattern>] [DEP-FIRST|DEP-LAST]

**Comment:** List jobs within this unitttype.

#### Arguments:

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **[![<job-name-pattern>]**: Optional. Any string which will be used to match the list of job-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.
- ⑩ **[DEP-FIRST|DEP-LAST]**: Defines the ordering of the list. This is usually not interesting - mostly used for -export/import options of this shell

#### Examples:

- ⑩ listjobs
- ⑩ listjobs Upgrade

### 7.3.18 Unittyp.setjob

**Syntax:** setjob <jobname> <jobtype> REGULAR|DISRUPTIVE <groupname> <parent-jobname>|NULL <description> <file-version>|NULL <unconfirmed-timeout> <stop-rules>|NULL [<repeat-count> <repeat-interval-sec>]

**Comment:** Make a job. There is a variety of jobtypes to be made. Using job to perform provisioning is to leverage the full force of Fusion, instead of the simple Profile/Unit provisioning.

#### Arguments:

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<jobname>:** Adding: Any string. Enclose names with double quotes if it contain spaces
- ⑩ **<jobtype>:** CONFIG - Change parameters/configuration/settings on CPE  
SOFTWARE - Upgrade/downgrade software/firmware on CPE  
TR069\_SCRIPT - Upload a proprietary script/config-file/etc to the CPE - will be executed on CPE  
RESTART - Reboot the CPE  
RESET - Factory Reset on CPE  
SHELL - Execute a Fusion shell script upon provisioning of CPE  
KICK - Will issue a 'kick' to the CPE, which will trigger a regular provisioning cycle  
TELNET - Will execute a Telnet-script on the CPE (make sure Telnet-parameters are defined on unit/profile)
- ⑩ **REGULAR|DISRUPTIVE:** Type of service window to work within. If set to REGULAR the job will obey the regular service window (allow the execution of the job within this window). Otherwise set to DISRUPTIVE (usually a more restricted time window).
- ⑩ **<groupname>:** A valid group name, only units that matches this group parameters will be able to run the job
- ⑩ **<parent-jobname>|NULL:** If the job depends upon another job, set the parent job here. Otherwise set to NULL
- ⑩ **<description>:** A non-essential (for Fusion) description.
- ⑩ **<file-version>|NULL:** The file to be used in the job (if the job type require a file). The version specified will be used to retrieve the file of the correct type:  
Job Type SOFTWARE -> File Type SOFTWARE  
Job Type TR069\_SCRIPT -> File Type TR069\_SCRIPT  
Job Type SHELL -> File Type SHELL\_SCRIPT  
Job Type TELNET -> File Type TELNET\_SCRIPT  
For other job types no file can be specified
- ⑩ **<unconfirmed-timeout>:** If a device doesn't reconnect after a job within this specified number of seconds, the job for this unit is considered 'unconfirmed failed'
- ⑩ **<stop-rules>|NULL:** Stop rules tells a job to 'pause' or 'stop' when a certain condition occur. A stop rule can be specified on this (regexp) format (a|u|c\d+(\d+)?|n\d+. A more detailed explanation:  
a : any failure  
u : unconfirmed failure (the device does not respond)  
c : confirmed failure (the device responds with an error message)  
n : counter

Rule example 1: a10/1000 : Will stop the job if 10 out of the last 1000 unit jobs has any failure

Rule example 2: u5,n1000 : Will stop the job if 5 unconfirmed failure occurs or if 1000 unit jobs has been executed.

- ⑩ [**<repeat-count>**]: Optional. Set to a positive number to repeat a job. Normal behaviour is to never repeat a job. This is useful for monitoring jobs.
- ⑩ [**<repeat-interval-sec>**]: Optional. Set to a positive number to decide the interval of repeating jobs. This interval will shorten the periodic inform interval if necessary. Default is 86400 (24h).

#### Examples:

- ⑩ setjob Upgrade SOFTWARE DISRUPTIVE "All units" NULL "An upgrade job" 0.9-alfa 600 "u5,a100" NULL NULL
- ⑩ setjob Test CONFIG REGULAR "All units" NULL "A config job" 1.2.3-alfa 600 a100 NULL NULL

### 7.3.19 Unittypе.deljob

**Syntax:** deljob <jobname>

**Comment:** Deletes job.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<jobname>**: A valid jobname

#### Examples:

- ⑩ deljob Test

### 7.3.20 Unittypе.listsyslogevents

**Syntax:** listsyslogevents

**Comment:** List syslog events within this unittypе.

#### Arguments:

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

### 7.3.21 Unittypе.setsyslogevent

**Syntax:** setsyslogevent <id> <name> <group-name>|NULL <expression> <store-policy> <script-name>|NULL <delete-limit>|NULL <description>

**Comment:** Add/change a syslog event. A syslog event is an event which is trapped on the syslog server if a certain syslog message arrive.



### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

### Arguments:

- ⑩ **<id>**: Adding: The id must be a number larger or equal to 1000. Changing: A valid syslog event id.
- ⑩ **<name>**: A non-essential name (for Fusion) of the syslog event.
- ⑩ **<group-name> | NULL**: If specified, the syslog-event will only match the units within the group
- ⑩ **<expression>**: This is the expression with which syslog messages will be matched. Syntax:
  - word matches messages which contain 'word'
  - \* matches 0 or more characters
  - \_ matches 1 character
  - ^ if used at beginning (allowed after !) of expression, matches expression only from start of message
  - \$ if used at end of expression, matches expression only at end of message
  - ! if used at beginning of expression, will negate the matching
  - | split the search, so 'A|B' translates to 'A OR B', and '!A|B' translates to 'NOT A AND NOT B'The matching is partial - you don't need to match the entire syslog message with the expression. A message will perform/execute the first event it matches, so avoid setting up events that match on the same syslog message. Check out the matching using the command 'syslog sm=<expression>'
- ⑩ **<store-policy>**: Storepolicy defines what this syslog event will store the message. You may choose one out of these policies:
  - STORE - This is the default task for all messages that arrive to the syslog server
  - DISCARD - Discards the message, a useful event to throw away garbage messages.
  - DUPLICATE - Duplicate messages will only be printed once every 60 minutes (count will be updated)
- ⑩ **<script-name>**: If specified a script will be run upon the Syslog Event
- ⑩ **<delete-limit>**: If specified, deletion limit will override the default severity limit specified in the Fusion Core Server configuration. It may delay or hasten the deletion of specific type of syslog message. Default value is 0 (which is to say 'use the default severity limit')
- ⑩ **<description>**: A non-essential (to Fusion) description of the syslog event.

### Examples:

- ⑩ `setsyslogevent 1000 SipRegFailed-monitor NULL "reg failed" STORE NULL NULL "A monitor over sip registrations that failed"`
- ⑩ `setsyslogevent 2000 BlockedMsg-drop NULL "gw: Blocked message" DISCARD NULL 0 "Will drop unnecessary messages from the device"`
- ⑩ `setsyslogevent 3777 Script-runner MyGroup "DNS Error" DUPLICATE setdebugflagonunit.xss 15 "Will force a system-debug flag to 1 using a script"`

## 7.3.22 Unittypedelsyslogevent

**Syntax:** delsyslogevent <id>

**Comment:** Delete a syslog event

### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<id>**: A valid syslog event id

#### Examples:

- ⑩ `delsyslogevent 5000`

### 7.3.23 Unittype.listexecutions

**Syntax:** `listexecutions [<match-expression>]|NULL`

**Comment:** List all script executions run by Core Script Executor

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **match-expression**: A regular expression string which will be matched against script, args and request-Id

#### Examples:

- ⑩ `listexecutions`
- ⑩ `listexecutions Test`

### 7.3.24 Unittype.setexecution

**Syntax:** `setexecution <fusion-script-file> <args>|NULL <request-id>|NULL`

**Comment:** Add/Change a shell script execution in the Core server.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<script>**: A file of type SHELL\_SCRIPT and optionally arguments
- ⑩ **<request-id>**: A request id, to identify the source/requestor of the script execution

#### Examples:

- ⑩ `setexecution test.fss NULL`
- ⑩ `setexecution "test.fss -uut:NPA201E-Pingcom" MyRequest-1234`

### 7.3.25 Unittype.generatetc

**Syntax:** `generatetc`

**Comment:** Generate Test Cases for the TR-069 test based on the Unit type parameters available and TR-181 (v1.9) and TR-098

### 7.3.26 Unitttype.deltcduplicates

**Syntax:** deltcuplicates

**Comment:** Delete duplicate Test Cases

### 7.3.27 Unitttype.listtc

**Syntax:** listtc [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** List all or a subset of Test Cases

#### Arguments:

- ⑩ **<method>|NULL:** Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ **<parameter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)
- ⑩ **<paratagmeter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

#### Examples:

- ⑩ listtc NULL NULL NULL
- ⑩ listtc VALUE WANConnection "[READONLY][GENERATED]"

### 7.3.28 Unitttype.showtc

**Syntax:** showtc <id>

**Comment:** Show the Test Case for the given id, as it would be if exported to file

#### Arguments:

- ⑩ **<id>:** A valid Test Case id (get from listtc command)

#### Examples:

- ⑩ showtc 123

### 7.3.29 Unitttype.deltc

**Syntax:** deltc [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** Delete all or a subset of Test Cases

#### Arguments:

- ⑩ **<method>|NULL:** Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ **<parameter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)
- ⑩ **<paratagmeter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

#### Examples:

- ⑩ deltc NULL NULL NULL
- ⑩ deltc VALUE WANConnection "[READONLY][GENERATED]"

### 7.3.30 Unitttype.exporttcfile

**Syntax:** exporttcfile <directory> <id>

**Comment:** Export one single Test Cases

**Arguments:**

- ⑩ <directory>: A directory to where the Test Case file will be exported
- ⑩ <id>: A valid Test Case id (get from listtc command)

**Examples:**

- ⑩ exporttc tr069test 123

### 7.3.31 Unitttype.exporttcdir

**Syntax:** exporttcdir <directory> [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** Export all or a subset of Test Cases

**Arguments:**

- ⑩ <directory>: A directory to where the Test Case files will be exported
- ⑩ <method>|NULL: Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ <parameter-filter>|NULL: Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)
- ⑩ <paratagmeter-filter>|NULL: Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

**Examples:**

- ⑩ exporttc NULL NULL NULL
- ⑩ exporttc VALUE WANConnection "[READONLY][GENERATED]"

### 7.3.32 Unitttype.importtcfile

**Syntax:** importtcfile <filename>

**Comment:** Import a Test Case

**Arguments:**

- ⑩ <filename>: A Test Case file to be imported

**Examples:**

- ⑩ importtcfile tr069test/2930.tc

### 7.3.33 Unitttype.importtcdir

**Syntax:** importtcdir <directory>

**Comment:** Import all Test Cases in a directory

**Arguments:**

- ⑩ <directory>: A directory from where all Test Case files will be imported

**Examples:**

- ⑩ `importtcdir tr069test`

### 7.3.34 Unitttype.listtesthistory

**Syntax:** `listtesthistory [startTms|NULL [endTms|NULL]]`

**Comment:** List the test history of this unit.

**Arguments:**

- ⑩ **startTms:** Optional, can also be set to NULL (to delete from beginning). Otherwise specify date using syntax from `syslog-command`
- ⑩ **endTms:** Optional, can also be set to NULL (to delete until end). Otherwise specify date using syntax from `syslog-command`

**Examples:**

- ⑩ `listtesthistory`
- ⑩ `listtesthistory 3d 1h`
- ⑩ `listtesthistory 20120701 20120705-1300`

### 7.3.35 Unitttype.deltesthistory

**Syntax:** `deltesthistory [startTms|NULL [endTms|NULL]]`

**Comment:** Delete the test history of this unit.

**Arguments:**

- ⑩ **startTms:** Optional, can also be set to NULL (to delete from beginning). Otherwise specify date using syntax from `syslog-command`
- ⑩ **endTms:** Optional, can also be set to NULL (to delete until end). Otherwise specify date using syntax from `syslog-command`

**Examples:**

- ⑩ `deltesthistory`
- ⑩ `deltesthistory 3d 1h`
- ⑩ `deltesthistory 20120701 20120705-1300`

## 7.4 UnittypParameter menu

List of commands available:

- ⑩ listvalues
- ⑩ setvalues
- ⑩ delvalues
- ⑩ generateenum

### 7.4.1 UnittypParameter.listvalues

**Syntax:** listvalues

**Comment:** List all enumerated values for this unittyp parameter.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical lay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

### 7.4.2 UnittypParameter.setvalues

**Syntax:** setvalues enum|regexp <value>+

**Comment:** Add/change the set of accepted values for this unittyp parameter.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **enum|regexp:** The only supported 'mode' in Fusion Web is 'enum'. Later it might be possible to specify a regexp with which a parameter value must match.
- ⑩ **<value>+:** One or more values. If a value contains spaces enclose value with double quotes.

**Examples:**

- ⑩ setvalues enum 1 2 3
- ⑩ setvalues enum On Off

### 7.4.3 UnittypParameter.delvalues

**Syntax:** delvalues

**Comment:** Will delete all enumerations for this unittyp parameter

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### **7.4.4 UnittypeParameter.generateenum**

**Syntax:** generateenum

**Comment:** Generates enumeration input for the setvalue command, the enumerations are taken from the TR069 Datamodels (TR104, TR098, TR181)

## 7.5 Profile menu

List of commands available:

- ⑩ listparams
- ⑩ setparam
- ⑩ delparam
- ⑩ listunits
- ⑩ setunit
- ⑩ delunit
- ⑩ delallunits
- ⑩ moveunit

### 7.5.1 Profile.listparams

**Syntax:** listparams [<profile-parameter-name-pattern>]

**Comment:** List profile parameters.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[!]<profile-parameter-name-pattern>]:** Optional. Any string which will be used to match the list of profile-parameters. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

**Examples:**

- ⑩ listparams
- ⑩ listparams DeviceInfo

### 7.5.2 Profile.setparam

**Syntax:** setparam <parameter-name> <value>

**Comment:** Add/change a profile parameter.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<parameter-name>:** Adding: A valid unittype parameter name. Changing: A valid profile parameter name.



- ⑩ **<value>**: Any String, maximum 255 character.

**Examples:**

- ⑩ `setparam System.X_COMPANY-COM.Test "Hello world"`

### 7.5.3 Profile.delparam

**Syntax:** `delparam <profile-parameter-name>`

**Comment:** Delete a profile parameter.

**Arguments:**

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<profile-parameter-name>**: A valid profile parameter name.

**Examples:**

- ⑩ `delparam Device.DeviceInfo.PeriodicInformInterval`

### 7.5.4 Profile.listunits

**Syntax:** `listunits [( <search-value> ) | ( <unittype-parameter-name> <operator> <value> [ <datatype> ] ) *]`

**Comment:** List units within this unittype.

**Arguments:**

- ⑩ **-a**: Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o**: Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerical in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[<search-value>]**: Optional. It will search among all parameter values and unit-ids
- ⑩ **<operator>**: Valid operators are EQ (equals), NE (not equals), LT (less), LE (less or equal), GE (greater or equal), GT (greater).
- ⑩ **<value>**: Any string. Max 255 characters. Long values will take more time to process in the event of a group search, so try to avoid if possible. The rules for matching are the same as for the 'listunits' commands in the unittype/profile context. In other words, use '%' in the value string to match 0 or more arbitrary characters and use '\_' in the value string to match 1 arbitrary character. Avoid % and \_ at the start of the value. To search for NULL-values, use the string NULL.
- ⑩ **[<datatype>]**: Optional. Default datatype is VARCHAR (which is the same as 'text' or 'string'). You can also use SIGNED for an integer which might be negative, or UNSIGNED for integers which are always positive.

**Examples:**

- ⑩ listunits
- ⑩ listunits John
- ⑩ listunits Device.DeviceInfo.SerialNumber EQ AABBCCDDEEFF00
- ⑩ listunits Device.DeviceInfo.SerialNumber NE %AABBCC%

### 7.5.5 Profile.setunit

**Syntax:** setunit <unitid>

**Comment:** Add a unit to this profile

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<unitid>:** A TR-069 unit id must adhere to the standard which is <OUI>[-<ProductClass>]<SerialNumber>. The OUI is a 6-digit hexadecimal number representing the vendor. Usually this is the same as the 6 first digits of the MAC address of the unit. You can also find the appropriate OUI on the internet. ProductClass is optional, but is usually the name of the product. SerialNumber is in many cases the MAC address, but could be any string. For non TR-069 units you may set <unitid> to any string. The unittypes protocol decides which rule applies.

**Examples:**

- ⑩ setunit A
- ⑩ setunit 123456-VeryFastRouter-123456789AB

### 7.5.6 Profile.delunit

**Syntax:** delunit <unitid>

**Comment:** Delete a unit.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<unitid>:** A valid unit id.

**Examples:**

- ⑩ delunit 123456-VeryFastRouter-123456789AB

### 7.5.7 Profile.delallunits

**Syntax:** delallunits

**Comment:** Deletes all units within this profile. This command is a combined/wizard command in the sense that you could achieve the same as this command does by running 2 other commands (listunits > FILE, delunit < FILE).

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

## 7.5.8 Profile.moveunit

**Syntax:** moveunit <unitid> <profilename>

**Comment:** Move a unit from this profile to another profile within the same unittype. The unit will keep all unit parameters.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<unitid>:** A valid unit id.
- ⑩ **<profilename>:** A valid profile name within this unittype.

**Examples:**

- ⑩ moveunit 123456-VeryFastRouter-123456789AB new-improved-profile

## 7.6 Unit menu

List of commands available:

- ⑩ listallparams
- ⑩ listunitparams
- ⑩ setparam
- ⑩ delparam
- ⑩ kick
- ⑩ inspection
- ⑩ extraction
- ⑩ read
- ⑩ write
- ⑩ refresh
- ⑩ generatetc
- ⑩ deltcuplicates
- ⑩ listtc
- ⑩ showtc
- ⑩ deltc
- ⑩ exporttcfile
- ⑩ exporttcdir
- ⑩ importtcfile
- ⑩ importtcdir
- ⑩ testsetup
- ⑩ enabletest
- ⑩ disabletest
- ⑩ listtesthistory
- ⑩ deltesthistory

### 7.6.1 Unit.listallparams

**Syntax:** listallparams [<parameter-name-pattern>]

**Comment:** List all parameters, both profile and unit parameter. Last column shows which are unit parameters (U) and which are profile parameters (P).

#### Arguments:

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

#### Arguments:

- ⑩ **[!]<parameter-name-pattern>]:** Optional. Any string which will be used to match the list of parameter-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

**Examples:**

- ⑩ `listallparams`
- ⑩ `listallparams DeviceInfo`
- ⑩ `listallparams ^Device.DeviceInfo.*`

**7.6.2 Unit.listunitparams**

**Syntax:** `listunitparams [<parameter-name-pattern>]`

**Comment:** List unit parameters.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the `-u` option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is `-o(<columnnumber><a|n><a|d>)+`. An example might be `-o3aa4nd`, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[`![<parameter-name-pattern>`]:** Optional. Any string which will be used to match the list of parameter-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with `!` to negate the search.

**Examples:**

- ⑩ `listunitparams`
- ⑩ `listunitparams DeviceInfo`
- ⑩ `listunitparams ^Device.DeviceInfo.*`

**7.6.3 Unit.setparam**

**Syntax:** `setparam <parameter-name> <value>`

**Comment:** Add/change a unit parameter.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<parameter-name>:** Adding: A valid unittype parameter name. Changing: A valid unit parameter name.
- ⑩ **<value>:** Any String, maximum 512 character.

**Examples:**

- ⑩ `setparam Device.DeviceInfo.PeriodicInformInterval 100`
- ⑩ `setparam System.X_OWERA-COM.Comment "Service is unavailable"`

## 7.6.4 Unit.delparam

**Syntax:** delparam <parameter-name>

**Comment:** Delete a unit parameter

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<parameter-name>:** A valid unit parameter

**Examples:**

- ⑩ delparam Device.DeviceInfo.PeriodicInformInterval

## 7.6.5 Unit.kick

**Syntax:** kick [async]

**Comment:** Kick a device using ConnectionRequestURL or UDPConnectionRequestAddress. This feature is only applicable for TR-069 devices. The kick will initiate a TR-069 provisioning session

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **async:** If this argument is used, the kick will be initiated and then the command will return. If not used, the command will wait up til 30 sec to see if the kick is completed.

**Examples:**

- ⑩ kick
- ⑩ kick async

## 7.6.6 Unit.inspection

**Syntax:** inspection [async]

**Comment:** Same command as kick, but when device connect it goes into INSPECTION mode, were all parameters with the I-flag set will be read and listed with the 'listallparameters' command. To write changes to the device in INSPECTION mode, simply change a unit parameter and then run the 'write' command. To read again from the device in INSPECTION mode, simply run the 'read' command

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **async:** If this argument is used, the kick will be initiated and then the command will return. If not used, the command will wait up til 30 sec to see if the kick is completed.

**Examples:**

- ⑩ inspection

- ⑩ inspection async

### 7.6.7 Unit.extraction

**Syntax:** extraction [async]

**Comment:** Same command as kick, but when device connect it goes into EXTRACTION mode, were all parameters from the CPE/device will be read and listed with the 'listallparameters' command. To write changes to the device in EXTRACTION mode, simply change a unit parameter and then run the 'write' command. To read again from the device in EXTRACTION mode, simply run the 'read' command

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **async:** If this argument is used, the kick will be initiated and then the command will return. If not used, the command will wait up til 30 sec to see if the kick is completed.

**Examples:**

- ⑩ inspection
- ⑩ inspection async

### 7.6.8 Unit.read

**Syntax:** read

**Comment:** If in INSPECTION or EXTRACTION mode, you may read again from the device, to update the status of the device parameters

### 7.6.9 Unit.write

**Syntax:** write

**Comment:** If in INSPECTION or EXTRACTION mode, you may write changes to the device, by changing some unit parameters and then executing this command

### 7.6.10 Unit.refresh

**Syntax:** refresh

**Comment:** Refresh the unit parameter cache in shell

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

### 7.6.11 Unit.generatetc

**Syntax:** generatetc [test-case-id]

**Comment:** If the optional arugment is omitted, it will generate Test Cases for the TR-069 test based on the Unit type parameters available and TR-181 (v1.9), TR-104 (v1.1) and TR-098 (v1.4)

**Arguments:**

- ⑩ **test-case-id**: Specify a "master" Test Case with many N SET parameters. The generate command will produce one TC for each SET parameter, from 1-N params

**Examples:**

- ⑩ generatetc
- ⑩ generatetc 4049

### 7.6.12 Unit.deltcduplicates

**Syntax:** deltcdduplicates

**Comment:** Delete duplicate Test Cases

### 7.6.13 Unit.listtc

**Syntax:** listtc [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** List all or a subset of Test Cases

**Arguments:**

- ⑩ **<method>|NULL**: Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ **<parameter-filter>|NULL**: Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)
- ⑩ **<paratagmeter-filter>|NULL**: Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

**Examples:**

- ⑩ listtc NULL NULL NULL
- ⑩ listtc VALUE WANConnection "[READONLY][GENERATED]"

### 7.6.14 Unit.showtc

**Syntax:** showtc <id>

**Comment:** Show the Test Case for the given id, as it would be if exported to file

**Arguments:**

- ⑩ **<id>**: A valid Test Case id (get from listtc command)

**Examples:**

- ⑩ showtc 123

### 7.6.15 Unit.deltc

**Syntax:** deltc [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** Delete all or a subset of Test Cases

**Arguments:**

- ⑩ **<method>|NULL**: Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ **<parameter-filter>|NULL**: Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)



- ⑩ **<paratagmeter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

**Examples:**

- ⑩ deltc NULL NULL NULL
- ⑩ deltc VALUE WANConnection "[READONLY][GENERATED]"

### 7.6.16 Unit.exporttcfile

**Syntax:** exporttcfile <directory> <id>

**Comment:** Export one single Test Cases

**Arguments:**

- ⑩ **<directory>:** A directory to where the Test Case file will be exported
- ⑩ **<id>:** A valid Test Case id (get from listtc command)

**Examples:**

- ⑩ exporttc tr069test 123

### 7.6.17 Unit.exporttcdir

**Syntax:** exporttcdir <directory> [<method>|NULL [<parameter-filter>|NULL [<tag-filter>|NULL]]]

**Comment:** Export all or a subset of Test Cases

**Arguments:**

- ⑩ **<directory>:** A directory to where the Test Case files will be exported
- ⑩ **<method>|NULL:** Optional. Could be set to VALUE or ATTRIBUTE. Setting it to NULL will match all (methods)
- ⑩ **<parameter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a part of a parameter name. Setting it to NULL will match all (parameters)
- ⑩ **<paratagmeter-filter>|NULL:** Optional if the previous argument is specified. Could be set to a series of strings enclosed in square brackets [ ] to match tags. Setting it to NULL will match all (tags)

**Examples:**

- ⑩ exporttc NULL NULL NULL
- ⑩ exporttc VALUE WANConnection "[READONLY][GENERATED]"

### 7.6.18 Unit.importtcfile

**Syntax:** importtcfile <filename>

**Comment:** Import a Test Case

**Arguments:**

- ⑩ **<filename>:** A Test Case file to be imported

**Examples:**

- ⑩ importtcfile tr069test/2930.tc

### 7.6.19 Unit.importtcdir

**Syntax:** importtcdir <directory>

**Comment:** Import all Test Cases in a directory

**Arguments:**

- ⑩ **<directory>**: A directory from where all Test Case files will be imported

**Examples:**

- ⑩ `importtcdir tr069test`

## 7.6.20 Unit.testsetup

**Syntax:** testsetup <steps> <reset-on-startup> <method> <parameter-filter>|NULL <tag-filter>|NULL

**Comment:** Setup the TR-069 parameter test

**Arguments:**

- ⑩ **<steps>**: A comma-separated list of the following allowed string: GET, SET, RESET or REBOOT. This is the order in which the test will be executed. It is not allowed to repeat a step.
- ⑩ **<reset-on-startup>**: 0 or 1. Will initiate a Factory Reset before the test starts if set to 1. Default is 0.
- ⑩ **<method>**: Can be either VALUE or ATTRIBUTE. If set to VALUE, Get/SetParameterValues method will be run with the test cases, otherwise Get/SetParameterAttributes will be run. Default is VALUE
- ⑩ **<parameter-filter>|NULL**: A part of a parameter-name which will narrow the set of test-cases to be run in the test.
- ⑩ **<tag-filter>|NULL**: A set of tags, each enclosed in square brackets, which will narrow the set of test-cases to be run in the test.

**Examples:**

- ⑩ `testsetup SET,REBOOT,GET 0 VALUE WANConnection "[READWRITE] [GENERATED]"`
- ⑩ `testsetup GET 0 VALUE WANConnection "[READONLY] [GENERATED]"`

## 7.6.21 Unit.enabletest

**Syntax:** enabletest

**Comment:** Enable TR-069 parameter test mode, which performs a series of tests based on test-cases added to the Unit Type. As long as the device is in test-mode, it will not provision normally, so make sure you disable the test when you're satisfied. Check up on the test-case commands in the Unit Type menu

## 7.6.22 Unit.disabletest

**Syntax:** disabletest

**Comment:** Disable TR-069 parameter test mode.

## 7.6.23 Unit.listtesthistory

**Syntax:** listtesthistory [startTms|NULL [endTms|NULL]]

**Comment:** List the test history of this unit.

**Arguments:**

- ⑩ **startTms**: Optional, can also be set to NULL (to delete from beginning). Otherwise specify date using syntax from syslog-command

- ⑩ **endTms**: Optional, can also be set to NULL (to delete until end). Otherwise specify date using syntax from syslog-command

**Examples:**

- ⑩ listtesthistory
- ⑩ listtesthistory 3d 1h
- ⑩ listtesthistory 20120701 20120705-1300

## 7.6.24 Unit.deltesthistory

**Syntax:** deltesthistory [startTms|NULL [endTms|NULL]]

**Comment:** Delete the test history of this unit.

**Arguments:**

- ⑩ **startTms**: Optional, can also be set to NULL (to delete from beginning). Otherwise specify date using syntax from syslog-command
- ⑩ **endTms**: Optional, can also be set to NULL (to delete until end). Otherwise specify date using syntax from syslog-command

**Examples:**

- ⑩ deltesthistory
- ⑩ deltesthistory 3d 1h
- ⑩ deltesthistory 20120701 20120705-1300

## 7.7 Group menu

List of commands available:

- ⑩ listparams
- ⑩ setparam
- ⑩ delparam
- ⑩ delallparams
- ⑩ count
- ⑩ listunits
- ⑩ listdetails

### 7.7.1 Group.listparams

**Syntax:** listparams [<parameter-name-pattern>]

**Comment:** List group parameters

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[![<parameter-name-pattern>]:** Optional. Any string which will be used to match the list of parameter-names. The string will be interpreted as a regular expression which is a very powerful matching language. If you want to know how to take full advantage of regular expressions you need to consult internet resources. Special feature: Prefix with '!' to negate the search.

**Examples:**

- ⑩ listparams
- ⑩ listparams DeviceInfo
- ⑩ listparams ^Device.Device.\*

### 7.7.2 Group.setparam

**Syntax:** setparam <unitttype-parameter-name> <operator> <value>  
[<datatype>]

**Comment:** Add/change a group parameter. A group parameter is acutally a search criteria, hence the equal/negation character.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **<unittype-parameter-name>**: Adding: A valid unittype parameter name.  
Changing: A valid group parameter with the #<number> at the end.
- ⑩ **<operator>**: Valid operators are EQ (equals), NE (not equals), LT (less), LE (less or equal), GE (greater or equal), GT (greater).
- ⑩ **<value>**: Any string. Max 255 characters. Long values will take more time to process in the event of a group search, so try to avoid if possible. The rules for matching are the same as for the 'listunits' commands in the unittype/profile context. In other words, use '%' in the value string to match 0 or more arbitrary characters and use '\_' in the value string to match 1 arbitrary character. Avoid % and \_ at the start of the value. To search for NULL-values, use the string NULL.
- ⑩ **[<datatype>]**: Optional. Default datatype is VARCHAR (which is the same as 'text' or 'string'). You can also use SIGNED for an integer which might be negative, or UNSIGNED for integers which are always positive.

#### Examples:

- ⑩ setparam Device.DeviceInfo.SerialNumber EQ AABBBCCDDEEFF00
- ⑩ setparam Device.DeviceInfo.SerialNumber NE AAB%BCC\_
- ⑩ setparam Device.DeviceInfo.SerialNumber GE 123 UNSIGNED

### 7.7.3 Group.delparam

**Syntax:** delparam <group-parameter-name>

**Comment:** Delete a group parameter. This will change the search criteria for this group, and all child groups of this group.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

#### Arguments:

- ⑩ **<group-parameter-name>**: A valid group parameter

#### Examples:

- ⑩ delparam Device.DeviceInfo.SerialNumber

### 7.7.4 Group.delallparams

**Syntax:** delallparams

**Comment:** Delete all group parameters. This will change the search criteria for this group, and all child groups of this group.

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

### 7.7.5 Group.count

**Syntax:** count

**Comment:** Will count the number of units that matches this group's search criteria

#### Arguments:

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c**: Will list the current context to the output. This is useful when using this output as input to another command using the -u option.

### 7.7.6 Group.listunits

**Syntax:** listunits

**Comment:** Will list all units that matches this group's search criteria

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

### 7.7.7 Group.listdetails

**Syntax:** listdetails

**Comment:** Will list detailed information about this group.

## 7.8 Job menu

List of commands available:

- ⑩ listdetails
- ⑩ listparams
- ⑩ listfailedunits
- ⑩ delfailedunits
- ⑩ status
- ⑩ start
- ⑩ pause
- ⑩ finish
- ⑩ setparam
- ⑩ delparam
- ⑩ refresh

### 7.8.1 Job.listdetails

**Syntax:** listdetails

**Comment:** List details about this job

### 7.8.2 Job.listparams

**Syntax:** listparams [DEFAULT|<unitid>]

**Comment:** List job parameters for this job.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabetically in an (a)scending order, next sort on the (4)th column (n)umericallay in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

**Arguments:**

- ⑩ **[DEFAULT|<unitid>]:** If set to DEFAULT the command will list all job parameters that are common for all units run by this job. If set to a particular unit-id, the command will list only those job parameters set for a particular unit. The job parameters transferred to the unit are the set of DEFAULT job parameters + the unit specific job parameters. If argument is skipped, all job parameters will be listed.

**Examples:**

- ⑩ listparams
- ⑩ listparams DEFAULT
- ⑩ listparams 123456-Router-123456789AB

### 7.8.3 Job.listfailedunits

**Syntax:** listfailedunits

**Comment:** Will list all units that has had a failure of sorts when running the job. It might be that the unit in the end has managed to execute the job.

**Arguments:**

- ⑩ **-a:** Will list all available information about this object to the output. This is useful for export of data.
- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.
- ⑩ **-c:** Will list the current context to the output. This is useful when using this output as input to another command using the -u option.
- ⑩ **-o:** Will order the listing of the output. Syntax of the option argument is -o(<columnnumber><a|n><a|d>)+ An example might be -o3aa4nd, which is to be read like this: Sort the (3)rd column (a)lphabatically in an (a)scending order, next sort on the (4)th column (n)umerically in an (d)escending order. Parts of the argument may be ignored if incorrectly specified.

## 7.8.4 Job.delfailedunits

**Syntax:** delfailedunits

**Comment:** Will delete all records of units that failed when running the job. This command is mostly used for export/delete of the entire unittype, but it's also necessary to run this command if you try to delete a job

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

## 7.8.5 Job.status

**Syntax:** status

**Comment:** Will list status of the job

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

## 7.8.6 Job.start

**Syntax:** start

**Comment:** Starts the job.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

## 7.8.7 Job.pause

**Syntax:** pause

**Comment:** Pause the job. You may start the job again later.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.



### 7.8.8 Job.finish

**Syntax:** finish

**Comment:** Finishes the job, it cannot be started after this. You cannot finish a job before you have stopped it.

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

### 7.8.9 Job.setparam

**Syntax:** setparam DEFAULT|<unitid> <unittype-parameter-name> <value>

**Comment:** Add/change a job parameter

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **DEFAULT|<unitid>:** Set to DEFAULT to create a job parameter that will apply to all units executing this job. Set to <unitid> to create a job parameter specific to a unit.
- ⑩ **<unittype-parameter-name>:** Adding: A valid unittype parameter. Changing: A valid job parameter.
- ⑩ **<value>:** Any string. Max 255 characters. The value will be set as a unit parameter on the unit executing the job.

**Examples:**

- ⑩ setparam DEFAULT Device.ManagementServer.PeriodicInformInterval 1800

### 7.8.10 Job.delparam

**Syntax:** delparam DEFAULT|<unitid> <unittype-parameter-name>

**Comment:** Delete a job parameter

**Arguments:**

- ⑩ **-u:** Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.

**Arguments:**

- ⑩ **DEFAULT|<unitid>:** Set to DEFAULT to delete a job parameter that will apply to all units executing this job. Set to <unitid> to delete a job parameter specific to a unit.
- ⑩ **<unittype-parameter-name>:** A valid job parameter.

**Examples:**

- ⑩ delparam DEFAULT Device.ManagementServer.PeriodicInformInterval

### 7.8.11 Job.refresh

**Syntax:** refresh

**Comment:** Refresh status about a job. Otherwise counters seen in 'listjobdetails' are not updated in this session.

**Arguments:**

- ⑩ **-u**: Will use the context available from another command (piping) or input-file. The context will be changed before the command is run.