

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## MoShell 11.0e User Guide

This document presents an overview of the functionality included in **MoShell**, command line syntax, revision history and other important information.

It is important that all engineers working with **MoShell** read this document before using the tool as it contains important operational information.

## Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Contact	5
1.2	MO concept	6
1.3	O&M services and protocols	6
1.4	MO Tree and MO Naming Conventions	7
1.4.1	LDN - Local Distinguished Name	7
1.4.2	RDN - Relative Distinguished Name	7
1.4.3	FDN - Full Distinguished Name	8
1.5	MOM - Managed Object Model	8
1.6	Moshell Functionality	8
1.6.1	Alarm Service	8
1.6.2	OSE shell	8
1.6.3	Configuration Service	9
1.6.4	Performance Management Service	9
1.6.5	Log service	9
1.6.6	File transfer	9
<b>2</b>	<b>Installation and user settings</b>	<b>10</b>
2.1	Installation for Unix (Solaris/Linux)	10
2.2	Installation for Windows (using Cygwin)	10
2.3	Moshell directory structure	12
2.4	Starting an moshell session	12
2.4.1	Starting up Moshell	12
2.4.2	Loading the MO Tree	13
2.4.3	Performing Actions on Loaded MO Stubs	13
2.5	User-specific settings	14
2.5.1	File properties	14
2.6	Settings related to telnet/ftp/ssh/sftp communication	15
2.7	CORBA settings	15
2.8	Ports used by moshell	16
2.9	Running moshell across secure tunnels (RSG)	16
<b>3</b>	<b>Command syntax, including Regular Expressions</b>	<b>17</b>
3.1	How MOs are Identified	17
3.1.1	RDN - Relative Distinguished Name	17
3.1.2	LDN - Local Distinguished Name	17
3.1.3	FDN - Full Distinguished Name	17
3.2	How to address the MOs in MO-related commands	18
3.3	Regular Expressions	19
3.4	How to specify attribute values in set/cr/acc commands	21
3.5	Moshell command line	21
3.6	Piping	22
<b>4</b>	<b>Command descriptions</b>	<b>22</b>
4.1	Basic MO commands	22
4.1.1	mom[abcdfloptuxs] [<moclass/struct/enum>] [<attribute/action>] [<attr-type>] [<attr-flags>] [<description>]	22
4.1.2	lt/clt/ltc[1-9] <motype-filter> root all [<attribute==value> AND/OR <attribute==value>]	24
4.1.3	lc/lcc[1-9] <moGroup> <moFilter> <proxy(s)> all	24
4.1.4	lu/llu <moGroup> <moFilter> <proxy(s)>	25
4.1.5	pr[s][m]/lpr[s][m] [<moGroup> <moFilter> <proxy(s)>] [<mimName>]	25
4.1.6	ma/lma <moGroup> <moGroup> <moFilter> <proxy(s)> all [<attribute-filter>] [<value-filter>]	25
4.1.7	mr/lmr <moGroup> <moGroup> <moFilter> <proxy(s)> all [<attribute-filter>] [<value-filter>]	26
4.1.8	mp	26
4.1.9	get/lget [<moGroup> <moFilter> <proxy(s)> all] [<attribute-filter> all] [<value-filter>]	26
4.1.10	hget[c][m]/lhget[c][m] <moGroup> <moFilter> <proxy(s)> [<attribute-filter>] [<value1-filter>] [<value2-filter>] [<value3-filter>] etc...	28

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

4.1.11	kget/lkget [<moGroup> <moFilter> <proxy(s)>] [<attribute-filter>] [<attribute-type>] [<attribute-flag>] [<attribute-description>]	28
4.1.12	fro/lfro[m] <moGroup> <moFilter> <proxy(s)> all [<attribute-filter> all] [<value-filter>]	29
4.1.13	sql/select <command> [   <unix-cmds>]	29
4.1.14	st/lst <moGroup> <moFilter> <proxy(s)> all [<state-filter>]	29
4.1.15	prod <moGroup> <moFilter> [<productdata-filter>]	29
4.1.16	lk/lk <moGroup> <moFilter> <proxy(s)>	30
4.1.17	lko/lko <moGroup> <moFilter> <proxy(s)>	30
4.1.18	set[m][c][1]/lset[m][c][1] <moGroup> <moFilter> <proxy(s)> <attribute> [<value>]	30
4.1.19	eset[c][1]/leset[c][1] <moGroup> <moFilter> <proxy(s)> <attribute-filter> [<value>]	31
4.1.20	rset/lrset <moGroup> <moFilter> <proxy(s)> <attribute> [<value>]	31
4.1.21	bl[s]/lbl[s] <moGroup> <moFilter> <proxy(s)>	32
4.1.22	deb/ldeb <moGroup> <moFilter> <proxy(s)>	32
4.1.23	acl/lac <moGroup> <moFilter> <proxy(s)> all [<action-filter>]	32
4.1.24	acc[e]/lacc[e] <moGroup> <moFilter> <proxy(s)> all <action>	32
4.1.25	cr[e][n] <ldn>	33
4.1.26	del[b]/ldel[b] <moGroup> <moFilter> <proxy(s)>	34
4.1.27	rdel/lrdel <moGroup> <moFilter> <proxy(s)>	34
4.1.28	gm[c][d]/lgm[c][d] <moGroup> <moFilter> <proxy(s)>	36
4.1.29	safe+/safe-/safe?	36
4.1.30	s+/s++/s-/s?	36
4.1.31	u+[s]/u-/u? [<file>]	37
4.1.32	run[x][1] [-l <lineNr>] <command file> [<var1>] [<var2>] ...	39
4.1.33	trun[is1cr] <moScript> <http://ipaddress/script>	39
4.1.34	Ctrl-Z; touch /tmp/xxxx; fg (abort MO command)	40
4.1.35	pol[b][c][d][h][i][k][m][p][s][r][u][w][y] [-m <mo>] [<interval>] [<waitTime>] [<checkTime>]	40
4.1.36	re[i]	41
4.1.37	getmom [<momversion>]	42
4.1.38	parsemom [<momFile>]	42
4.1.39	flt/fltc <motype-filter>	42
4.1.40	ld <ldn>	42
4.1.41	fget/lfget <moGroup> <moFilter> <proxy(s)> all [<attribute>]	42
4.1.42	eget/leget <moGroup> <moFilter> <proxy(s)> all [<attribute>]	43
4.1.43	sget/lsgget/skget/lsgget/shget/lshget <moGroup> <moFilter> <proxy(s)> all	43
4.1.44	fset/lfset <moGroup> <moFilter> <proxy(s)> all <attribute> [<value>] [<attribute-type>]	43
4.1.45	facc/lfacc <moGroup> <moFilter> <proxy(s)> all <action> [<param1>] [<param2>]	43
4.1.46	fdel/lfdel <moGroup> <moFilter> <proxy(s)>	44
4.2	Other MO commands	44
4.2.1	cvls/cvmk/cvms/cvset/cvrm/cvrbm/cvcu/cvget[f]/cvput/cvls1	44
4.2.2	inv[h]xbpcr [<Filter>] [<stateFilter>]	45
4.2.3	cab[adefghlmrstxc] [   <unix-cmds> ]	47
4.2.4	stc[p][r] [<Filter>] [<stateFilter>]	49
4.2.5	std[ar] [<filter>]	50
4.2.6	stv[b][r] [<Filter>] [<stateFilter>]	52
4.2.7	stt[r] [<Filter>] [<stateFilter>]	53
4.2.8	ste[gr] [<Filter>] [<stateFilter>]	54
4.2.9	sti[bcopr] [<Filter>] [<stateFilter>]	57
4.2.10	sts	61
4.2.11	str	62
4.2.12	hc	63
4.2.13	dgc[meiasrfkx] [-m <rophours>] [-d <logdays>] [-b <boards boardgroup> all] [-k <nrdumps>] [-f <mofilter>] [<logdir>]	64
4.2.14	diff[a][d][m][o]/ldiff[a][d][m][o]	64
4.2.15	lkr[a]	66
4.2.16	resub <lubLink> [<VplTp> <Subrack>] [<VplTp>]	68
4.2.17	ir[cdps] [<lubLink>] [<CM>]	69
4.2.18	tg[r][c][d]	70
4.2.19	ueregprint/uer[d][t][i][s][p][v] [-m <mod> -i <imsi> -u <ueref> -n <maxUes> -c <utrancell> -r <iublink>] [<attribute-filter> =<value> all]	73
4.2.20	ced[h][p][s][g][r] [-m <module(s)> -c <utrancell> -r <iublink>] [   <unix-cmds>]	75
4.2.21	al[atkcg][u] [-al-u <alarm-id>] [   <unix-cmds>]	78
4.2.22	lg[abcdefghijklmnpqrstuvwxy12345] [-l <logdirectory logfile zipfile>] [-m <minustime>] [-p <plustime>] [-s <startdate>] [-e <enddate>] [-g <boardgroup>] [-n <nodefilter>] [-x <XBlog-filter ESlog-filter>] [-b <xb>] [-d <nrdumps>] [   <unix-cmds>]	79
4.3	Other commands	83
4.3.1	uv [<string> <var>=value]	83

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

4.3.2	pv [<string>]	83
4.3.3	!/l <unix-command>	83
4.3.4	l+[m][m][s][o]/l-/? [<logfile>]	83
4.3.5	dbc[s][a] [<cvname> <dbdat-file> <cvzip-file> <mobatch-folder>]	84
4.3.6	dbd [<cvname> <dbdat-file> <cvzip-file>] [<cvname> <dbdat-file> <cvzip-file>]	90
4.3.7	<ose/coli command> [ <unix-cmds>]	90
4.3.8	coli	92
4.3.9	comcli	92
4.3.10	c+/c1/c2/c-/c?	92
4.3.11	<linux/rcs-coli/comcli command> [ <unix-cmds>]	93
4.3.12	mcc/lmcc <moGroup> <moFilter> <proxy(s)> <comcli commands(s)> [ <unix-cmds>]	94
4.3.13	bo[ar]/ba[swdpmu]/br[wdm]/be[0-50]/bp	94
4.3.14	lh[z] <boardGroup> <moGroup> <OSE-command> run <commandfile> [   <unix-cmds>]	96
4.3.15	mon/monu/mond/monf/mon?/mon- [<board(s)> <boardGroup(s)>] [</path/to/logfile.pcap>]	97
4.3.16	sql+/sql-/sql? [<cheap>]	98
4.3.17	pgu[c][f][r] [-p <board1,board2,...>] /path/to/newLM [<cvcomment>]	98
4.3.18	proclod/proctemp [ <unix-cmds>]	99
4.3.19	proglst/progkill [-e] [<string>] [ <unix-cmds>]	99
4.3.20	fte <te-command> [<trace-groups> all] [<string>] [ <unix-cmds>]	100
4.3.21	goxb[ib] [-p <advpw>] <commands> [ <unix-cmds>]	100
4.3.22	ftree[f][d][1] [<lnh> <directory>] [ <unix-cmds>]	101
4.3.23	ftget[c]/ftput[c]/ftdel[a] [<options>] <source>[/] [<destination>]	101
4.3.24	htget <remoteFile> [<localfile/localldir>]	102
4.3.25	edit <remoteFile>	103
4.3.26	fclean[f][ff][a][d][e] [<lnh> <directory>] [-f <filename-filter>]	103
4.3.27	hi [<commandFilter>], !<commandNr>	104
4.3.28	time[t] <command> <logfile>	104
4.3.29	lmid[c][h]/upid[om] <pattern> refresh	104
4.3.30	p/w/pw/b	105
4.3.31	prox[+-]	105
4.3.32	col	105
4.3.33	ul	106
4.3.34	conf[bld][+]	106
4.3.35	gs[+]/gsg[+]	106
4.3.36	ip2d <ip-address>	106
4.3.37	d2ip/h2ip <number>	107
4.3.38	h2d/d2h <number>	107
4.3.39	h2b/b2h <number>	107
4.3.40	encpw <password>	107
4.3.41	mos2ro <moshell.zip>	108
4.3.42	wait <delay> <newtime>	108
4.3.43	return	108
4.3.44	print	109
4.3.45	alias/unalias <alias> <command>	109
4.3.46	lf[c] <file>	109
4.3.47	bg[g]/bgs/bgw [<commands> <id> all] [<maxtime>]	109
4.3.48	- smd[slcr] [-m <days>] [-s <size>] [-f <filter>] [-o a s n] [-u <user> all] [-d <directory>] [-n <max>]	110
4.3.49	q/by/exit/quit [<exitcode>]	111
4.4	PM commands	111
4.4.1	pmom[acdp]/lmom[c] [<moclass>] [<counter>] [<data-type>] [<flags>] [<description>]	111
4.4.2	kmom[d] [<area>] [<kpiname>] [<MOclass>] [<formula>] [<kpidescription>]	112
4.4.3	pget/lpget [<moGroup> <moFilter> <proxy(s)> all] [<attribute-filter> all] [<value-filter>]	112
4.4.4	spget/lspget [<moGroup> <moFilter> <proxy(s)> all] [<attribute-filter> all] [<value-filter>]	112
4.4.5	hpget[c][m]/lhpget[c][m] <moGroup> <moFilter> <proxy(s)> [<attribute-filter>] [<value1-filter>] [<value2-filter>] [<value3-filter>] etc...	112
4.4.6	pdiff/lpdiff [<moGroup> <moFilter> <proxy(s)> all] [<attribute-filter> all] [<value-filter>]	112
4.4.7	hpdiff[m]/lhpdiff[m] [<moGroup> <moFilter> <proxy(s)> all] [<attribute-filter> all] [<value1-filter>] [<value2-filter>] [<value3-filter>]	113
4.4.8	pmx[hfdnsckwlb3zei] [<moFilter> <mogroup>] [<counter-filter> <kpi(s)>] [-l <zipfile> <directory>] [-w <web-directory>] [-m <minushours>] [-p <plushours>] [-s <startdate>[.<starttime>]] [-e <enddate>[.<endtime>]] [-a -d -h] [-o <outputFormat>] [-tz <hrs>] [-f <formulafile>] [-j <precision>] [  <unix-cmds>]	113
4.4.9	pmr[agfkwop3z] [-g <moFilter> <mogroup>] [-z <mogroup>] [-r <report(s)>] [-l <zipfile> <directory>] [-w <web-directory>] [-i <iubCellModule-file>] [-f <formulafile>] [-c <configfile>] [-m <minushours>] [-p <plushours>] [-s <startdate>[.<starttime>]] [-e <enddate>[.<endtime>]] [-o <outputFormat>] [-t <thresholdfile>] [-tz <hrs>] [ <unix-cmds>]	115

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

4.4.10	pme[fd][cgur] [<pm_logdir>] [-b <boardgroup>] [-f ] [-m <minushours>] [-p <plushours>] [-s <start-date>[.<starttime>]] [-e <enddate>[.<endtime>]]	118
4.4.11	pst [<scan-filter> <scan-proxy>] [<scan-state>]	119
4.4.12	pgets[m][n][r] [<scan-filter> <scan-proxy>] [<contents-filter>]	119
4.4.13	pcr[pcfpda]/lpcr[pcfpda] <scannerName> <moclass-filter> <moinstance-filter> <mo-group> <counter-file> <counter-filter>] [<granularity>]	120
4.4.14	pbl <scan-filter> <scan-proxy>	121
4.4.15	pdeb <scan-filter> <scan-proxy>	121
4.4.16	pdel <scan-filter> <scan-proxy>	121
4.4.17	emom [uetr gpeh ctr all] [<event-filter>]	122
4.4.18	pset[d]	122
<b>5</b>	<b>Lazy</b>	<b>123</b>
5.1	Software Upgrade	123
5.2	RNC lub operations	123
5.3	Common RNC lub Integration Problems	124
5.4	Common RNC lu/lur Integration Problems	125
<b>6</b>	<b>Scripting</b>	<b>125</b>
6.1	Preset Variables	125
6.2	Variable assignment	127
6.3	Hashtables (arrays)	131
6.4	If/Else constructs	131
6.5	For constructs	133
6.6	User-defined functions	135
6.7	Nesting for and if statements	136
6.8	Example scripts	137
<b>7</b>	<b>Utilities</b>	<b>137</b>
7.1	Network Management Tools	137
7.2	Parameter Auditing Tools	137
7.3	PM Tools	138
7.4	Miscellaneous Tools	138
<b>8</b>	<b>Server Maintenance</b>	<b>138</b>
8.1	Hanging Processes	138
8.2	Disk full	139
8.3	Run out of memory	139
<b>9</b>	<b>Offline Mode and Multi Mode</b>	<b>139</b>
9.1	Offline Mode	139
9.2	SQL Mode	140
9.3	Multi Mode	140
<b>10</b>	<b>Revision History</b>	<b>141</b>

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 1 Introduction

### 1.1 Contact

For bug reports, installation issues, change requests etc. please use the forum at:

<http://utran01.au.ao.ericsson.se/moshell>

Alternatively contact:

- [Finn.Magnusson@ericsson.com](mailto:Finn.Magnusson@ericsson.com)
- [Joakim.xo.Ostlund@ericsson.com](mailto:Joakim.xo.Ostlund@ericsson.com)

#### **TR for MoShell**

Put bug reports on the web page (see Section 1.1) or write them in MHWEB:

To write an MoShell TR/CR in MHWEB:

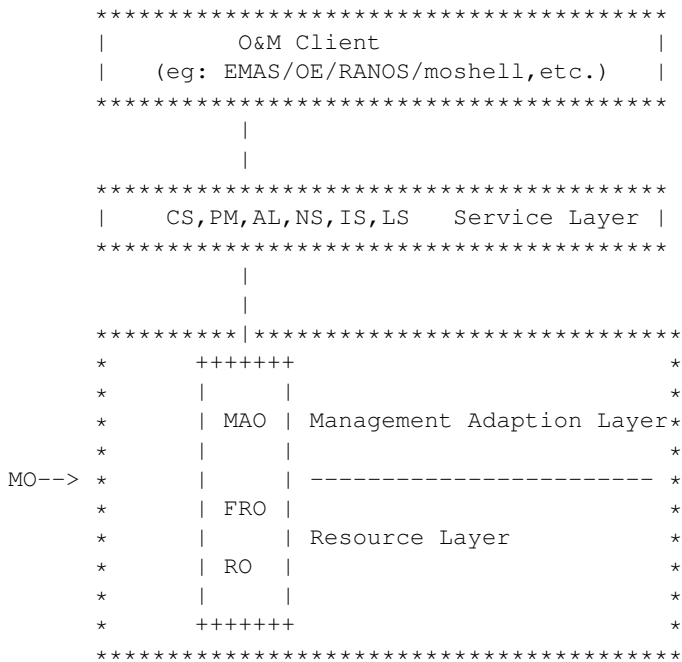
1. Product should be AMOS CXC 172 4313
2. MHO should be LMIR7-BASIC

In order to get the fastest resolution to your problem, please add the following information to your TR or bug report:

- `uv` and `pv` printout
- Any *complete* printout relevant to the fault
- How to recreate the fault (you can for instance include the `hi` printout showing all the commands that led to the fault)

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 1.2 MO concept



The O&M client can access the MOs through a number of services:

- *Configuration Service (CS)*: to read and change configuration data. Configuration data is stored in the MO attributes.
- *Performance Measurement (PM)*: to setup statistics scanners or event filters. The statistics counters are stored in MO pm-attributes and output to an XML file every 15 minutes. The events are output into binary files every 15 minutes.
- *Alarm Service (AS)*: to retrieve the list of alarms currently active on each MO.
- *Notification Service (NS)*: to subscribe and receive notifications from the node, informing about parameter/alarm changes in the MOs.
- *Inventory Service (IS)*: to get a list of all HW and SW defined in the node.
- *Log Service (LS)*: to save a log of certain events such as changes in the configuration, alarms raising and ceasing, node/board restarts, jvm events, O&M security events, etc

The MO is a way of modelling resources in a CPP node. It consists of:

1. A *Management Adaption Layer* which is implemented in java, in the MP running the jvm (the O&M MP).  
The purpose of the MAO (Management Adaptation Object) is to interface towards the various O&M services described above.
2. A *Resource Layer* consisting of Facade Resource Object (FRO) and a Resource Object (RO) which are implemented in C and run on the various boards. The RO is the actual resource modelised by the MO. The purpose of the FRO is to act as an interface between the MAO and the RO, by handling the configuration transactions and storing configuration data for the RO.

## 1.3 O&M services and protocols

The MO services described above (CS, AS, PM, IS, NS) are carried by the IIOP protocol (Internet Inter-ORB Protocol), also called CORBA (Common Object Request Broker Architecture).

At startup, the node generates its IOR (Interoperable Object Reference) and stores it in a **nameroot.ior** file which can be used by the O&M client to access the node. The IOR file contains the node's IP address and some specifications as to how the clients can send requests to the node.

The OSE-shell (also called COLI) can be accessed through telnet/ssh or locally, through the serial port (RS232). It gives direct access to the operating system, the file system, and parts of the FRO/RO layer.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

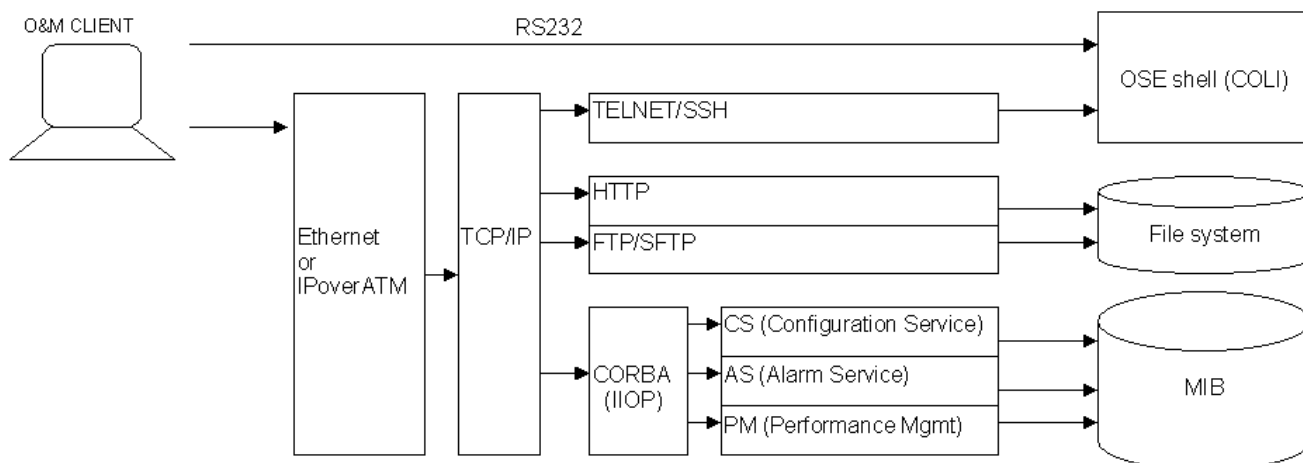


Figure 1: CPP nodes have various access methods for different services. For Managed Services like CS, CORBA is used. For the command shell, ssh or telnet is used. To collect PM XML ROP files FTP is used.

## 1.4 MO Tree and MO Naming Conventions

### 1.4.1 LDN - Local Distinguished Name

The MOs are organised in a hierarchical structure.

Each MO instance is uniquely identified in the node by its *Local Distinguished Name* (LDN).

The highest MO in a node, the so called *root MO* is the *ManagedElement*. This MO represents the whole node.

There is only one instance of the ManagedElement MO in the node and it is referenced by the LDN: **ManagedElement=1**

The string at the left of the equal sign is called the MO class (or MO type) and the string at the right of the equal sign is called the MO identity. In the case of the root MO, the *MO class* is **ManagedElement** and the *identity* is **1**.

If an MO is located further down in the MO tree, the LDN must contain the MO classes and identities of all the parents of that MO, in a sequence going from the root MO down to the MO in question. See example below:

```

ManagedElement=1
ManagedElement=1,Equipment=1
ManagedElement=1,Equipment=1,Subrack=MS
ManagedElement=1,Equipment=1,Subrack=MS,Slot=19
ManagedElement=1,Equipment=1,Subrack=MS,Slot=19,PlugInUnit=1
ManagedElement=1,Equipment=1,Subrack=MS,Slot=19,PlugInUnit=1,Program=DbmFpgaLoader
  
```

From this example, we can see that the ManagedElement has a child called **Equipment=1** which has a child called **Subrack=MS** (representing the main subrack of the node), which has a child called **Slot=19** (representing the slot in position 19), which has a child called **PlugInUnit=1** (representing the board located in that slot), which has a child called **Program=DbmFpgaLoader** (representing one of the programs loaded in that board).

The LDN of the lowest MO (the one called **Program=DbmFpgaLoader**) contains the address of all successive parents of that MO all the way up to the ManagedElement.

### 1.4.2 RDN - Relative Distinguished Name

The string located at the far right of an LDN, just after the last comma, is called a *Relative Distinguished Name* (RDN).

It is a unique way of addressing a MO instance in relation to its closest parent.

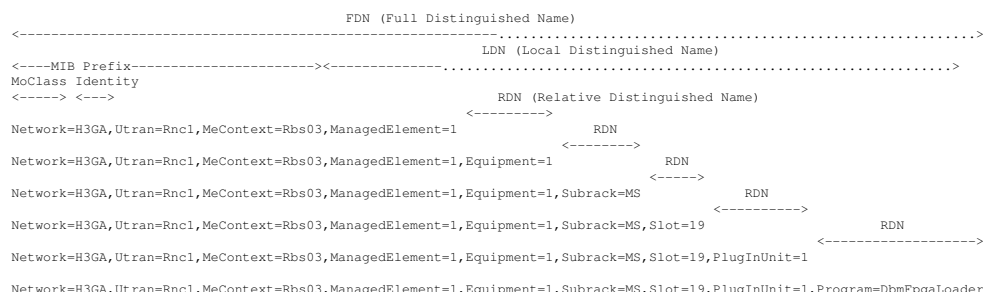
This means that there is only one MO instance with the RDN **Program=DbmFpgaLoader** under the parent MO **ManagedElement=1,Equipment=1,Subrack=MS,Slot=19,PlugInUnit=1**. However, there could be another MO instance with the same RDN under a different parent MO. For instance, there could be an MO instance with the RDN **Program=DbmFpgaLoader** under the parent MO **ManagedElement=1,Equipment=1,Subrack=MS,Slot=23,PlugInUnit=1**.

Therefore the RDN is a relative way of addressing an MO instance.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

### 1.4.3 FDN - Full Distinguished Name

When a node is connected to a Network Management System such as OSS-RC, there is a need to uniquely address each MO within the whole network. The *Full Distinguished Name* (FDN) adds a network element prefix (MIB prefix) in front of the LDN of each MO instance in order to specify which node this MO belongs to. See the figure below, summing up the FDN/LDN/RDN concept:



## 1.5 MOM - Managed Object Model

Each MO class contains a number of attributes which are used to store *configuration data* or *performance measurement data*.

Each MO class can also support a number of defined *actions*. These represent certain operations which can be performed by the MO. A typical example is the action **restart** which will cause the MO to restart the resource it is responsible for (e.g. a board, a program, etc.).

The *Managed Object Model* (MOM) is a reference document describing all the MO Classes that can exist in a node, together with their *attributes* and *actions*.

The format of the MOM can be UML, XML, HTML, or MS-Word.

The XML version of the MOM is usually stored on the web server of the node at the address:

`http://<ipaddress>/cello/oe/xml/<filename>.xml`

The MOMs for each SW release is also stored in HTML format on `http://cpistore.ericsson.se`

## 1.6 Moshell Functionality

MoShell is a text-based O&M client providing access to the following services:

- *Configuration service (CS)*
- *Alarm service (AS)*
- *Performance management service (PM)*
- *Log service (LS)*
- *OSE shell (COLI)*
- *File transfer (ftp/http)*

Access to all services is supported both in secure mode (secure Corba, ssh, sftp) and unsecure mode (unsecure corba, telnet, ftp).

### 1.6.1 Alarm Service

The list of active alarms can be retrieved with the commands `al` (to show an overview) or `ala` (the same as `al`, with more details).

### 1.6.2 OSE shell

Any OSE shell command can be typed at the moshell prompt and the output can be piped through external utilities (which exist in your workstate/server) if required.

Examples:



Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

```
te log read
te log read | grep ERROR
```

- Only the \$ prompt is supported. For instance, it is not possible to type `lhsh 000100` and expect a prompt to that board. The workaround is to type the command on the same line as the link handler shell, eg `lhsh 000100 te log read` or put a semicolon after the `lhsh xxxx`, eg `lhsh 001400 ; te log read ; vii ; llog`. Type `h ose` at the moshell prompt for more info.
- Other commands which require a shell such as `sqlc` have their own implementation. See Section [4.3.7](#).
- Any *Loco* commands should be written as `loco ts\nloco ...`. You can achieve this automatically using aliases, see Section [4.3.45](#).

### 1.6.3 Configuration Service

Moshell supports the following 6 operations from the configuration service:

1. **GetChildren** to load all or parts of the MO-tree
2. **GetAttribute** to read the attributes of an MO
3. **CallAction** to perform an action on an MO
4. **SetAttribute** to set (change) the value of an MO attribute
5. **CreateMO** to create a new MO in the node
6. **DeleteMO** to delete an MO from the node

### 1.6.4 Performance Management Service

Moshell supports the following operations from the performance management service:

- List Scanners and Event Filters
- Create Scanner
- Stop Scanner
- Resume Scanner
- Delete Scanner
- Set Event Filter

### 1.6.5 Log service

Moshell supports fetching and parsing of the following logs:

- availability log
- system log
- event log
- alarm log
- command log
- O&M security event log
- COLI log
- Hardware inventory log
- JVM events log (upgrade log)

### 1.6.6 File transfer

Moshell can download/upload files and directories to/from the node, using `http`, `ftp` or `sftp`.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 2 Installation and user settings

### 2.1 Installation for Unix (Solaris/Linux)

Download the file **moshellxxx.zip** from <http://utran01.epa.ericsson.se/moshell> to your home directory **/home/youruser** or to the temp directory **/tmp**. Very important: do not store the zip file inside the moshell folder otherwise the installation will be corrupted.

Then go to the folder containing the zipfile (`cd <folder>`) and run the following commands:

```
unzip -o moshellxxx.zip
bash moshell_install
```

When prompted to enter the directory where you want to install moshell, it is recommended to specify your HOME directory ( `~` ). If you have executed `moshell_install` from your home directory then you can press the enter key and the current directory is selected.

If a previous moshell installation already exists, it is recommended to install in the same directory as the old one. This way, all your custom files (jar/xml files, site files, etc.) get copied across to the new revision and the old revision gets moved to a different location so you can still access it if needed.

Note: In the case of AMOS installation use option `-a`, ie: `bash moshell_install -a` (must be run as root on OSS masterserver).

Note: for linux 64-bit, the 32-bit libc library is required, the package name is `libc6-i386` or `glibc.i686` or `ia32-libs`

Running moshell for the first time:

If you have set the PATH variable correctly in your `~/.bashrc` file, you should be able to run moshell from any directory. E.g:

```
moshell <ipaddress>
```

If this is the first time moshell is installed in this location, then it will download a number of jar files from the node. No progress indicator will be shown so just be patient as it will take a few minutes. Progress can be seen by doing `"ls -l"` in the `moshell/jarxml` directory. For more information about user settings etc, check the user guide.

If the Moshell execution fails on linux with the following error:

```
moshell/commonjars/lib/lin64/filefuncs.so: cannot restore segment prot after reloc: Permission denied
```

Then try to run the following commands, while logged in as root:

```
chcon -t texrel_shlib_t ./commonjars/lib/lin64/libz.so.1
chcon -t texrel_shlib_t ./commonjars/lib/lin64/filefuncs.so
```

If Moshell is unable to connect to the node on port 22, try executing the program `moshell/commonjars/ssh` manually. If it fails with the following error:

```
moshell/commonjars/ssh: error while loading shared libraries: cannot restore segment prot after reloc:
```

Then try running the following command, while logged in as root:

```
chcon -t texrel_shlib_t commonjars/ssh
```

### 2.2 Installation for Windows (using Cygwin)

When running on Windows, Moshell uses a *unix emulator* known as *Cygwin*.

Go directly to step 13, *MoShell Installation*, if you have already installed and configured Cygwin previously.

1. Go to the website <http://www.cygwin.com/>
2. Click on *Install Now*
3. Save the **setup-x86.exe** file, then execute it. Make sure to use *32-bit setup* (64-bit not supported by moshell).
4. Choose *Install from internet*, click *Next*
5. Root Directory **C:\cygwin** (It is not recommended to choose a different directory, especially if it contains spaces)  
If prompted for "Default Text File Type", choose UNIX (not DOS). Then click *Next*.
6. Select Internet Connection: put the proxy settings (generally use *IE5 settings* works with ESOE PCs). Click *Next*.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

7. Choose a Download Site.

8. Select Packages: Add the following packages:

- under *Archive* select *zip* and *unzip*
- under *Perl* select *perl* and *perl-XML-Simple*
- under *Net* select *inetutils*, *openssh*, and *openssl*
- under *Shells* select *rxvt VT102*
- under *Tcl* select *expect*
- optional: under *Editors* select *vim* (if you want to be able to edit files with vi)

9. Click *Next*, install will start. Wait for installation to complete.

10. Download the file [http://utran01.au.ao.ericsson.se/moshell/cygwin\\_install.txt](http://utran01.au.ao.ericsson.se/moshell/cygwin_install.txt) to C:/Cygwin .

11. Click on Start → Run.

In the "Run" window, type: `cmd` , then press <enter>.

A DOS window opens. At the DOS prompt, execute the following commands:

```
c:
cd cygwin
bin\perl.exe cygwin_install.txt
```

This will create the following files: `c:/cygwin/etc/profile`, `c:/cygwin/cygwin.bat`,  
`c:/cygwin/home/youruserid/.bashrc`, `c:/cygwin/home/youruserid/.Xdefaults`,  
`c:/cygwin/home/youruserid/.inputrc`.

If those files already exist, they are automatically moved to the folder `c:/cygwin/tmp/installbackup`.

12. Open a new cygwin terminal window. The window should be black with white text and the prompt should like this: `[~]$`

If not, then go through all the steps again and make sure you haven't missed out anything.

More info about Cygwin installation issues can be found at: <http://cygwin.com/faq/faq0.html>

Uninstall instructions for cygwin can be found at

[http://cygwin.com/faq/faq\\_setup.html#faq\\_setup\\_uninstall\\_all](http://cygwin.com/faq/faq_setup.html#faq_setup_uninstall_all)

13. Moshell installation. Follows these steps if you already have a working Cygwin environment.

- Download the file **moshellxxx.zip** from <http://utran01.au.ao.ericsson.se/moshell> to your home directory  
**c:/cygwin/home/youruserid**
- Open the cygwin shell and run:

```
unzip -o moshellxxx.zip
bash moshell_install
```

When prompted to enter the directory where you want to install moshell, it is recommended to specify your HOME directory (`~`).

If you have executed `moshell_install` from your home directory then you can press the enter key and the current directory is selected.

If a previous moshell installation already exists, it is recommended to install in the same directory as the old one. This way, all your custom files (jar/xml files, site files, etc.) get copied across to the new revision and the old revision gets moved to a different location so you can still access it if needed.

When prompted to enter the path to Java, choose either of:

- `/cygdrive/c/Windows/system32/java`
- `/cygdrive/c/Program~1/javasoft/jre/<version>/bin/java`

14. Running moshell for the first time

If you have set the PATH variable correctly in your `~/.bashrc` file, you should be able to run moshell from any directory. E.g:

```
moshell <ip-address>
```

If this is the first time moshell is installed on this PC, then it will download a number of jar files from the node. No progress indicator will be shown so just be patient as it will take a few minutes. Progress can be seen by doing `ls -l` in the **moshell/jarxml** directory. For more information about user settings etc, check the user guide.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

15. If the following error occurs when executing moshell on Vista:

```
fatal error - unable to remap C:\cygwin\home\xxx\moshell\commonjars\lib\file.dll to same address as parent
```

- download the files <http://utran01.epa.ericsson.se/moshell/fixvista.txt> and <http://utran01.epa.ericsson.se/moshell/fixvista.bat> to the desktop
- execute the fixvista.bat file

Known Limitation: **CTRL-C** may not work all the time. In this case, it is possible to do instead: **CTRL-Z**, then `kill %1`.

## 2.3 Moshell directory structure

The moshell directory contains a number of files and subdirectories:

- **logfiles/** To store the logfiles generated by the various utilites (**moshell**, **mobatch**, **swstat**, etc.)
- **examples/** Example of input files required by the utilities (command files, site files, etc.)
- **cmdfiles/** Place to store your command files (to be used by **mobatch**, **monode**, **telbatch**, **telnode**, etc.)
- **sitefiles/** Place to store your sitefiles (to be used by **mobatch**, **telbatch**, **swstat**, **swup**, etc.)
- **jarxml/** Place to store jar files (**oms.jar** and **vbjorb.jar**) and xml-MOM files (used by **Moshell**). Specific to certain nodes.
- **commonjars/** Place to store jar files common to all nodes
- **moshell** The file used to startup moshell. Contains some customizable variable settings.
- **mobatch** Run moshell commands on several nodes in parallel
- **gawk** Script interpreter used by the various utilities
- **prog.awk**, **funcs.awk** Main code for moshell, cannot be run on its own.
- **moshellUserGuide.pdf** Help file
- **rncaudit** Audit and consistency check of rnc data towards cell data CDR, utranrelations CDR, baseline, uerc, etc.
- **swstat** To view SW revisions, CV's and delete old upgrade packages
- **rbsaudit** Audit and consistency check of rbs data towards rbs data and baseline. Generation of mobatch corrective scripts.
- **swup** Network SW upgrades
- **cvms** create and set cv's on several nodes in parallel
- **momdoc** convert MOM from xml to html
- **mocmd** generate moshell command file from a baseline parameter file

## 2.4 Starting an moshell session

This section gives a brief overview of how to get started once you have installed moshell.

### 2.4.1 Starting up Moshell

A Moshell session is started from the Unix shell prompt using command: `moshell <node-name>|<node-address>`

If connecting with node name, an entry must exist in the ipdatabase file reference the node name against an ip or dns address.

Other ways of starting moshell are described by typing `moshell` on its own as well as in chapter Section 9 (Offline mode/multi mode Chapter) .

Upon startup, and running the command "It all", moshell will go through the following steps:

1. Download the node's IOR file and store it on the workstation. The node's IOR file is fetched from [http://nodeipaddress/cello/ior\\_files/nameroot.ior](http://nodeipaddress/cello/ior_files/nameroot.ior)
2. Check the node's MOM version The node's MOM is fetched from <http://nodeipaddress/cello/oe/xml/<filename>> where <filename> is one of the files listed in the user variable `xmlmomlist`. The MOM version is derived from the "mim" tag inside the MOM file, eg: `<mim name="RNC_NODE_MODEL_E" version="5" release="3">` becomes `RNC_NODE_MODEL_E_5_3`. If this MOM version does not exist on the workstation (under **moshell/jarxml** directory), then it is downloaded from the node and stored in that directory. If the MOM version could not be figured out (ie. moshell could not find any MOM on the node), the MOM specified in the moshell uservariable `default_mom` is used.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

3. Parse the MOM and generate an internal table specifying all MO classes, attributes, and actions supported by the node.
4. Initiate CORBA communication with the node by using the information contained in the IOR file.
5. Read the FDN of the Root MO
6. Ready to receive commands from the user

At this stage, it is possible to access the *Alarm Service* and *OSE shell* but the *Configuration Service* is limited since Moshell doesn't have any knowledge of what MO instances are contained in the node's MO tree (apart from the root MO).

The following commands are of use at this stage:

- `h` - to show the help and list of commands. Can be used with a command name after to show help about that command.

The menus are split into two (`m` and `n`) only for readability purposes.

## 2.4.2 Loading the MO Tree

Once Moshell first connects to the node it has no knowledge of the MO structure on the node (except for the *ManagedElement* MO class). In order to *get attributes* or *call actions* you first need to load the MO *stubs* onto your Moshell client.

The whole MO tree can be read with command `lt all`. The LDN of each MO of the MO tree is then allocated a "proxy" number and stored in an internal table in Moshell memory. The internal proxy table can be printed via the command `pr` which will show for each MO, the LDN and the proxy number.

When performing an operation on an MO (get, set, action, etc.), either the proxy number or the LDN can be given as argument.

In fact, by using a *Regular Expression* matching part of the LDN, an operation can be performed on several MOs at a time.

More information about this can be found in Section 3 or by typing `h syntax` at the prompt.

To save memory on the workstation, it is possible to load only parts of the MO tree instead of the whole MO tree.

For instance, by typing `lt pluginunit`, only the LDNs of MOs whose MO class is *PluginUnit* will be read.

Instead of typing the whole MO class, it is possible to type a regular expression that will match the MO class.

In this case, `lt plu` would be the same as `lt pluginunit`, since the string `plu` matches `pluginunit`

More information about this in Section 4 or by typing `h lt` and `h lc` at the prompt.

## 2.4.3 Performing Actions on Loaded MO Stubs

To perform operations on one or several MOs which you loaded in the previous section, follow the command syntax shown on the menu.

1. Example: To read the MO attributes of the MO with LDN **ManagedElement=1,Equipment=1,Subrack=MS,Slot=19,PluginUnit=1** you would type the following:

```
pr plu      #then lookup the proxy identity of that MO
get <proxy> #enter the MOs proxy identity as argument to the "get" command
```

OR

```
lget ms,slot=19,pluginunit=1$
```

2. Example: To read the MO attributes of all MOs whose MO class is *PluginUnit*

```
get plu #the get command will operate on all MOs whose RDN matches "plu"
```

More info about this in Section 3 or by typing `h syntax` at the Moshell prompt.

Help for each command can be found in Section 4 or by typing `h <command-name>` at the Moshell prompt.

**!!! Important note for CDMA nodes !!!** Most CDMA nodes do not keep a MOM on the node's harddisk.

To force moshell to use the correct MOM, here are some workarounds:

- store the correct MOM in your directory **moshell/jarxml**, then, after the moshell startup, use the command `parsemom <mom>` to parse the correct MOM, or specify the path to this MOM in your *default\_mom* user variable (see Section 2.5 for info on user variables).
- store the correct MOM on the node at **/c/public\_html/cello/oe/xml/CelloMOM.xml**
- check if the node has a MOM that is under a different file name than is specified in the moshell uservariable *xmlmomlist*. This can be done by doing `ls /c/loadmodules_norepl` or `ftree /c/loadmodules_norepl` and search for a MOM file (file extension **.xml**)

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 2.5 User-specific settings

There are a number of moshell configuration parameters (called *user variables*) which can be set either permanently or on a session basis. These settings have a default value which is defined in the file **moshell/moshell**. If one or more user variables need to be changed from the default value, it is recommended to store the new setting in one of the moshellrc files (**~/.moshellrc** or **moshell/jarxml/moshellrc** or **moshell/jarxml/moshellrc.\$USER**) instead of the moshell file. This way, the new setting will be kept even after an moshell upgrade.

It is also possible to define user variables on a session basis by:

1. using the command `uv [var=value]` from the moshell prompt (type `h uv` for more info)
2. or use the `-v` option from the command line when starting moshell (type `moshell` on its own for more info)

It is possible for many users to run moshell from a central location and have their logfiles, credential files, user variables and aliases stored in their own home directory. This is mainly intended for Solaris or Linux networks where many users will be able to run moshell from a common location without having to install it in their home directory. The common location can be the vobs or any user's home directory, for example the administrator's account or a common account.

The administrator can define a number of user variables and aliases and save them into the **moshell/jarxml/moshellrc** file. These user variables and aliases will apply to all users and will be kept after each moshell installation. It is also possible for the administrator to define individual moshellrc files which will be located as **moshell/jarxml/moshellrc.\$USER**.

Each user can also define their own settings and aliases and save them into the **~/.moshellrc** file in their home directory. If this file is not present, it will be created automatically and can be modified any time.

All user variables that are defined in the file **moshell/moshell** can be given a new value in the **moshell/jarxml/moshellrc** and/or the **~/.moshellrc** and/or the **moshell/jarxml/moshellrc.\$USER** file.

The user variables defined in **moshell/jarxml/moshellrc.\$USER** override those defined in **~/.moshellrc** which override those defined in **moshell/jarxml/moshellrc**, which in turn override those defined in the file **moshell/moshell**.

Here is a short list of user variables, look inside the **moshell** file for more info on each variable:

- *disk\_check* - check if enough free disk space on the workstation: 0=no check, 1=warning only, 2=exit if not enough space
- *disk\_limit* - the minimum free disk space required by the disk check, default 1G (1 Gigabyte).
- *java* - path to Java executable
- *ip\_database* - path to the IP database file (see example of this file in **moshell/examples/mobatch\_files**)
- *secure\_shell*, *secure\_ftp*, *username*, *ip\_connection\_timeout*, *ip\_inactivity\_timeout* - see Section 2.6 for info
- *corba\_class*, *security\_method*, *credential*, *sa\_credential*, *sa\_password*, *sls\_urls*, *sls\_username*, *sls\_password* - see Section 2.7 for information
- *default\_mom* - path to the default MOM, to use in case no MOM is found on the node
- *prompt\_highlight* - to enable or disable the bold font of the prompt
- *set\_window\_title* - to enable or disable the display of a window title
- *http\_port* - to specify the HTTP port value (e.g 80 for CPP, 8080 for simcello)
- *keepLmList* - files that shouldn't be deleted by the `fclean` command.
- *bldebsset\_confirmation*, *lt\_confirmation* - to specify if confirmation message shall be given in `lt/bl/deb/set` commands.
- *logdir* - path to the logfiles directory (this one can only be changed from moshellrc or .moshellrc, not uv or -v)
- a few more variables, see the **moshell** file for further details

Look in the **moshell** file for a more complete list and detailed explanation of each variable.

### 2.5.1 File properties

All files and subdirectories belonging to the moshell package are NON-writeable to outside users. Only the owner of the account where moshell is installed can make modifications to the installation. The only exception is the subdirectory **moshell/jarxml** which is writeable to everyone. The reason is because all users need to be able to store new xml MOM files in that directory (since the xml MOM gets frequently updated for every new node SW release).

Note: the same jar files can be used by all users, regardless of what CPP SW is running on the node. For instance, it's ok to use CPP 4 jar files towards a CPP 3 node. This also the case as CPP 5.1 moves towards JacORB.



Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 2.6 Settings related to telnet/ftp/ssh/sftp communication

The following user variables relate to telnet/ftp communication.

All settings can be either set in moshell file, in `~/.moshellrc` or via the `uv` command in the moshell session. See Section 2.5 and `h uv` for more info. Also check the **moshell** file for more info on each variable.

- *username* - which username to use when logging in to the node via telnet/ftp/ssh/sftp
- *secure\_shell* - whether to use telnet or ssh for access to OSE shell
- *secure\_ftp* - whether to use ftp or sftp for file transfers.
- *ip\_connection\_timeout* - timeout for the establishment of telnet/ftp/ssh/sftp connections
- *ip\_inactivity\_timeout* - inactivity timeout for telnet/ftp/ssh/sftp connections
- *telnet\_port/http\_port/ftp\_port/secure\_port* - to use a different port for telnet/http/ssh/sftp. Useful for connecting to SimCello/CPPemu or running moshell over the RSG using port forwardings.
- *node\_login* - whether to login or not (no login for SimCello or backup mode)

So, in order to enable ssh/sftp instead of telnet/ftp, do one of the following:

1. Set the variables *secure\_shell* and *secure\_ftp* in **moshell** file to the value 1 (not recommended since the value will get reset at the next moshell install/upgrade).
2. OR Add the following lines in the `~/.moshellrc` or the **moshell/jarxml/moshellrc** file

```
secure_shell=1
secure_ftp=1
```

3. OR Run the following commands from the Moshell prompt (the setting will only be valid for the current session):

```
uv secure_shell=1
uv secure_ftp=1
```

4. OR start moshell with the option `-v secure_shell=1,secure_ftp=1`

See Section 2.5 to find out more about setting user variables.

Note that for running ssh/sftp, an encryption key file must be located on the node. Two kinds of key can be used:

1. *RSA key* - the file `/c/java/host.p12` or `/c/configuration/ssh_host_rsa_key` is used
2. *DSA key* - the file `/c/configuration/ssh_host_dsa_key` is used

An example of each file can be found in **moshell/commonjars** directory.

Important Note regarding the *ip\_connection\_timeout*: By default, this timeout is set to 45 seconds in the **moshell** file. Certain OSE shell commands might not print any output for a longer period of time, eg: `format`, `rcp`, etc. To avoid having a connection timeout when running these commands, it is advised to extend the *ip\_connection\_timeout* prior to running the command, by doing `uv ip_connection_timeout=600` (for example).

## 2.7 CORBA settings

To specify which corba SW and corba security settings moshell should be using, use the following values of the user variable `corba_class`.

- `corba_class=1` : connect in secure mode with the visibroker (vbjorb) software
  - `security_method=1` (requires java  $\geq$  1.2.2) : use a host credential `host.p12`, only supported for nodes running CPP5.0 and below. The uservariable `credential` must be set accordingly, to point to the `host.p12` file. The `host.p12` file is downloaded from the PKS server or from the node.
  - `security_method=2` (requires java  $\geq$  1.3.1) : use a stand-alone credential `sam.pbe`. The `sam.pbe` file is downloaded from the SLS server, its path and password must be given in the uservariables `sa_credential` and `sa_password`.
- `corba_class=2` (requires java  $\geq$  1.3.1) : connect in unsecure mode with the prismtech (jacorb) SW.
- `corba_class=3` (requires java  $\geq$  1.4.2\_05) : connect in secure mode with the jacob SW, using a stand-alone credential `sam.pbe`. The `sam.pbe` file is downloaded from the SLS server, its path and password must be given in the uservariables `sa_credential` and `sa_password`.
- `corba_class=4` (requires java  $\geq$  1.4.2\_05) : connect in secure mode with the jacob SW, using a network-mode credential `ssucredentials.xml` which is automatically downloaded from SLS server. The username and password for

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

SLS login must be specified in the uservariables `sls_username` and `sls_password`. The address of the SLS is read from the node but can also be specified in the uservariable `sls_urls`.

- `corba_class=5` (requires java  $\geq$  1.4.2\_05) : connect in secure mode with the jacorb SW, using an already downloaded network-mode credential `ssucredentials.xml`. The path to the `ssucredentials.xml` file must be given in the uservariable `nm_credential`.

For more information about how to set the user variables, see Section 2.5 and `h uv`. Also check the **moshell** file for more info on each variable.

## 2.8 Ports used by moshell

If there is a firewall between MoShell and the CPP nodes, then the following TCP ports need to be open in the firewall:

### 1. Corba

#### a) For nodes running CPP5 and lower (vbjorb):

- Port 56834: for both secure and unsecure CORBA
- Port 56835: for both secure and unsecure CORBA
- Port 56836: not used

#### b) For nodes running CPP5.1 and above (jacorb):

- Port 56834: for unsecure CORBA
- Port 56835: not used
- port 56836: for secure CORBA

### 2. HTTP: 80

### 3. Telnet: 23 and/or SSH: 22

### 4. FTP: 21 and/or SFTP: 22. If FTP is used, an additional port range needs to be open for the data connection (21 is for control only). If using SFTP, then only port 22 is necessary (no extra ports for data).

### 5. Target monitor: ports 33077 to 33087 are needed in order to monitor the CPP trace and error log, using the monitor6054 utility.

### 6. Optional: for subscription to Corba CS/FM notifications, using the utility `runClient.sh` (-c/-a options): callback port range 53248-55295 from the node to the client. This is not needed for moshell, only for the utility `runClient.sh` -c/-a options.

## 2.9 Running moshell across secure tunnels (RSG)

**NOTE!!!:** Take good care when using this method as it allows to interact with live nodes from a customer network.

This method explains how to run moshell across the Remote Support Gateway secure tunnels.

This instruction is designed for a 3-step RSG, consisting of 3 successive servers:

- an Ericsson Gateway, we will call it *EGW*
- a Customer Support Gateway, we will call it *SSG*
- an O&M server (e.g. Solaris or Linux workstation such as OSS-RC, OMINF, etc), we will call it *WS*. This machine should have IP connectivity to the CPP nodes.

In the instruction, we will use the following abbreviations:

- **<EGWipaddress>** **<EGWuser>** **<EGWpw>** - ipaddress/dnsname, userid and password of the Ericsson Gateway
- **<SSGipaddress>** **<SSGuser>** **<SSGpw>** - ipaddress/dnsname, userid and password of the Customer Support Gateway
- **<WSipaddress>** **<WSuser>** **<WSpw>** - ipaddress/dnsname, userid and password of the O&M workstation
- **<nodeIP>** - ipaddress/dnsname of the CPP node

### 1. Prerequisite if running this method in Windows (only needs to be done the first time):

- In StartMenu -> Settings -> Control panel -> System -> Device manager -> Network Adapters :
- Select Menu Action -> Add legacy hardware
- Then follow instruction to add Microsoft loopback adapter



Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- In StartMenu -> Settings -> Network Connections, look for the loopback adapter (should be called something like *Local Area Connection <number>*, Rename it to *loopback*

2. Set up the IP address on loopback interface:

On Windows:

```
netsh interface ip set address name="loopback" static <nodeIP> 255.255.255.0
```

On Unix/Linux (need root pw):

```
su - ; ifconfig eth0:1 <nodeIP> netmask 255.255.255.0 ; exit
```

3. Run the following ssh session and exit it:

```
ssh -l <EGWuser> <EGWip>
<EGWpw>
show userports
show ports
exit
```

From the show userports printout, you can see the applicable port range, from the show ports, you see which ones are already in use.

Choose an appropriate port, we will call it <EGWport>

4. Run the following ssh session and leave the window open:

```
ssh -l <EGWuser> -L 10022:127.0.0.1:<EGWport> <EGWip>
<EGWpw>
add <EGWport> <WSip> 22
ssh <SSGip> <SSGuser>
<SSGpw>
```

5. Get a new terminal window on the PC, run the following commands and leave the window open:

```
ssh -g -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null \
-p 10022 -l <WSuser> -L 20022:<nodeIP>:22 -L 20080:<nodeIP>:80 \
-L 20023:<nodeIP>:23 -L 56834:<nodeIP>:56834 -L 56835:<nodeIP>:56835 127.0.0.1
```

6. Get a new terminal window on the PC and run the following line (split for readability)

```
moshell -v portbase=20000 <nodeIP>
```

## 3 Command syntax, including Regular Expressions

### 3.1 How MOs are Identified

MOs can be identified using the RDN, LDN or FDN.

#### 3.1.1 RDN - Relative Distinguished Name

This is used to identify an MO in relation to its nearest parent in the MO tree.

The RDN contains MO Class (also called MO Type), the equal sign, and MO identity. Example:

```
AtmPort=MS-24-1
```

**AtmPort** is the *MO Class*, **MS-24-1** is the *identity*.

#### 3.1.2 LDN - Local Distinguished Name

This is used to uniquely identify an MO within a node.

The LDN shows the hierarchy above the MO, within the Managed Element's MO tree. Example:

```
ManagedElement=1,TransportNetwork=1,AtmPort=MS-24-1
```

#### 3.1.3 FDN - Full Distinguished Name

This is used to uniquely identify an MO within a network (used by RANOS/CNOSS/OSS-RC). Example:

```
SubNetwork=AUS,SubNetwork=H2RG_0201,MeContext=St_Leonards_Station_2065010,ManagedElement=1,TransportNetwork=1,AtmPort=MS-24-1
```

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

## 3.2 How to address the MOs in MO-related commands

The first argument in the MO-related commands is usually used to specify the MOs that should be used by the command.

There are currently six different ways to specify the MO(s):

### 1. all

All loaded MOs will be affected. Example:

- a) `get all userlabel` to get attribute *UserLabel* on all MOs

Note: instead of `all`, it is also possible a regex wildcard such as `.` or `*`. This has the same effect.

### 2. Proxy ID(s)

All MO(s) with the given proxy id(s) will be operated upon.

To specify several MO proxies, there are two ways:

- Specify each Proxy ID with a space in between. Example:
  - a) `pr 0 2 5` to print the MO proxies 0, 2 and 5.
- Give a range of Proxy IDs. Examples:
  - a) `pr 4-10` prints MO proxies from 4 to 10.
  - b) `pr 10-4` prints all MO proxies from 10 down to 4 (reverse order, useful for deleting MOs).
  - c) `acc 10-20 restart` calls the action **restart** on MOs with proxy 10 up to 20.

Note: proxy ranges and individual proxy ids can be mixed on the same line.

Example: `pr 0 2 3-5 8 10-12`

### 3. Link handler (for PluginUnit and Spm MOs only!). Examples:

- a) `acc 001400 restart` - to restart the MO **Subrack=MS,Slot=14,PluginUnit=1**.
- b) `bl 001900/sp0.lnh` - to lock the first SPM on the SPB in slot 19 with LDN: **Subrack=MS,Slot=19,PluginUnit=1,Spu=1,Spm=1**. Note that MOs start counting from 1 and the link handlers start from 0!

### 4. MO Group

MO Groups are user defined groups of MOs. All MO(s) belonging to the given MO group will be operated upon.

To create a MO group, see command description for `ma/lma` in Section 4.1.6. MO groups can also be created with the commands `hget/lhget`, `lk/llk`, `st/lst`, `pdiff/lpdiff`.

Note: In RNC, running the `bo` command will automatically create a number of MO groups containing the `cc/dc/pdr` device MOs for each module.

### 5. Board Group

MOs (**PluginUnit** or **Spm**) mapped onto the boards belonging to the given board group will be operated upon.

Example 1:

- `baw sccp sccp` All boards with the swallocation matching "sccp" will go into the board group "sccp"
- `bl sccp` All **PluginUnit** or **Spm** MOs connected to boards of this board group will be locked

Example 2: in RNC, using the default board groups created after running the `bo` command:

- `acc mod10 restart`
- `pr dc10`
- `acc dc10 restart` the board group `dc10` is mapped onto the **Spm** MOs
- `bl dc10dev` in this case we are using the MO group containing the **Device** MOs, see above

### 6. MO-Filter (regular expression)

MO(s) whose LDN/RDN match the *regular expression pattern* will be affected.

If the command starts with `/` then the pattern will match against the LDN.

If the command doesn't start with `/`, then the pattern will match against the RDN.

If the command doesn't start with `/`, and the filter contains no commas, then the pattern will match against the RDN.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

If the command doesn't start with */*, and the filter contains commas, then the pattern will match against the LDN but will not include the children.

Examples:

a) `pr ms-24-1`

```
TransportNetwork=1,AtmPort=MS-24-1
```

b) `lpr ms-24-1`

```
TransportNetwork=1,AtmPort=MS-24-1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc32
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc33
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc337
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc332
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc34
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc35
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc40
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc64
```

c) `pr ms, slot=5, plug`

```
Equipment=1,Subrack=MS,PlugInUnit=1
```

d) `lpr ms, slot=5, plug`

```
Equipment=1,Subrack=MS,PlugInUnit=1
Equipment=1,Subrack=MS,PlugInUnit=1,Program=basic
Equipment=1,Subrack=MS,PlugInUnit=1,Program=nss
Equipment=1,Subrack=MS,PlugInUnit=1,Programs=spas
....
```

When using the *MO-Filter*, it is a good idea to test the pattern with `pr/lpr` command before issuing a `get/set/acc/cr/del` command, in order to see which MOs will be matched by the pattern.

Sometimes, a second or third argument can be given, which is usually a string matching the *attribute* or *attribute value* that you want to display.

### 3.3 Regular Expressions

**Note: MOSHELL pattern matching is NOT case sensitive**

The search string that is used in the filters is a Unix Regular Expression (like the patterns used in the `grep -E` command). Therefore, special meta-characters such as `.` `*` `[` `]` `^` `$` can be used.

Short description of some meta-characters:

- `"."` - any single character
- `"*"` - 0 or more occurrences of the previous character
- `"[ ]"` - matches a character or range of characters inside the brackets
- `"[^ ]"` - NOT matching a character or range of characters inside the brackets
- `"|"` - OR
- `"^"` - beginning of string
- `"$"` - end of string
- `"!"` - negation
- `"%"` - reverse order

Examples of using meta-characters:

- `a*` means a or aa or aaa, etc.
- `.*` is like a wildcard as it matches 0 or more occurrences of any character
- `[a-z]` matches all letters from a to z
- `[abe]` matches letters a,b, and e

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- `[^3]` matches any character but not 3
- `3|5|6` matches 3 or 5 or 6
- `^a.*4$` matches a string beginning with a and finishing with 4, with any character in the middle

Regular expressions can also be grouped together using brackets, e.g:

- `cell(11|23|45)` matches cell11 or cell23 or cell45

Examples of using regular expressions in the filters:

1. `lpr ms-24-1.*vp2`

```
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc34
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc35
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc40
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc64
```

2. `lpr %ms-24-1.*vp2`

```
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc64
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc40
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc35
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc34
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2
```

3. `lpr !loadmodule|program`

- All MOs except those matching **loadmodule** or **program** will be printed

4. `lpr 20.*os`

```
Equipment=1,Subrack=1,Slot=20,PlugInUnit=1,Etm4=1,Os155PhysPathTerm=1
Equipment=1,Subrack=1,Slot=20,PlugInUnit=1,Etm4=1,Os155PhysPathTerm=2
```

5. `pr cc[1-4]`

```
TransportNetwork=1,AtmCrossConnection=AtmCC1
TransportNetwork=1,AtmCrossConnection=AtmCC2
TransportNetwork=1,AtmCrossConnection=AtmCC3
TransportNetwork=1,AtmCrossConnection=AtmCC4
```

6. `pr cc[135]`

```
TransportNetwork=1,AtmCrossConnection=AtmCC1
TransportNetwork=1,AtmCrossConnection=AtmCC3
TransportNetwork=1,AtmCrossConnection=AtmCC5
```

7. `lpr =6.*prog.*=1`

```
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=15
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=1
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=14
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=13
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=12
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=11
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=10
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=19
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=18
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=17
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=16
```

8. `lpr =6.*prog.*=1$`

```
Equipment=1,Subrack=1,Slot=6,PlugInUnit=1,Program=1
```

9. `lpr ms-24-1`

```
TransportNetwork=1,AtmPort=MS-24-1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc32
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc33
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc337
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp1,VpcTp=1,VclTp=vc332
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc34
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc35
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc40
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc64
```

10. `lpr ms-24-1.*=vc[^3]`

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc40  
TransportNetwork=1,AtmPort=MS-24-1,VplTp=vp2,VpcTp=1,VclTp=vc64

### 3.4 How to specify attribute values in set/cr/acc commands

1. For attributes of type **Struct**, use the following syntax: attr1=val1[,attr2=val2[,attr3=val3]]...

Example:

```
set sid sib11 sib11repperiod=128
set mtp3bspitu sppriority prioslt=2
set mtp3bspitu sppriority prioslt=2,prioco=2
```

2. For attributes of type MoRef, just type the MO LDN (without **ManagedElement=1**). E.g.:

```
lset AtmPort=1221,VplTp=vpl atmTrafficDescriptor
transportnetwork=1,atmtrafficedescriptor=C1P4500
```

It is also possible to skip the first parent (eg **TransportNetwork**, **SwManagement**, etc). E.g.:

```
cr rncfunction=1,utranCell=30451,utranrelation=30451to305212
Attribute 1 of 1, utranCellref (moRef:UtranCell): utranCell=30521
```

3. For attributes of type **array of MoRefs**, separate each element of the array with spaces. Eg:

```
set jvm admclasspath loadmodule=oms loadmodule=vbjorb ...

acc aal2pathdistributionunit=1 addPath
Parameter 1 of 1, aal2PathVccTpId (sequence-moRef-Aal2PathVccTp):
aal2pathvcctp=csa aal2pathvcctp=csb
```

4. For attributes of type **array of Struct**, separate each element of the array with semicolons. Eg:

```
set rncfunction aliasPlmnIdentities
mcc=300,mnc=23,mnclength=2;mcc=345,mnc=32,mnclength=2;mcc=208,mnc=123,mnclength=3
```

5. For attributes of type array of integer/long/float/string/boolean, separate each element of the array with commas or spaces. Eg:

```
set antfeederCable=6 ulattenuation 10,10,10,10,10,10,10,10,10,10

set antfeederCable=6 ulattenuation 10 10 10 10 10 10 10 10 10 10

set jvm options -Xms65000k,-Dhttp.root=/c/public_html,
-Dse.ericsson.security.PropertyFileLocation=/c/java/SecurityManagement.prp

set cell=3041 ActiveOverlaidCDMAChannelList true,true,false,true

set cell=3041 ActiveOverlaidCDMAChannelList true true false true

acc managedelementdata addDhcpServerAddress
Parameter 1 of 1, ipAddressArray (sequence-string): 10.1.1.3,10.1.1.4
```

6. For attributes of type array of integer, it is also possible to specify ranges of values.

Eg, in the command below, the attribute will be set to 1,2,3,4,5,23,24,25,26

```
set IpInterface=1,DscpGroup=1 dscpValues 1-5,23-26
```

7. To input an empty value:

- in **set** command, just leave the value field blank. Eg:

```
set 0 userlabel
set reliableprogramunit admpassiveslot
```

- in **cr** command, type **null** or **d**. This is only supported for non-mandatory (restricted) attributes, because mandatory attributes must be given a value.
- in **acc** command, type **null**. This is only supported for parameters of type MoRef or String.

### 3.5 Moshell command line

The command line uses the Readline library from bash. Here are some of the supported function keys:

- **right arrow** or **Ctrl-f** - move forward one character
- **left arrow** or **Ctrl-b** - move backward one character
- **up arrow** - previous command in history buffer

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- **down arrow** - next command in history buffer
- **backspace** - delete one character backward
- **Ctrl-d or <del>** - delete one character forward
- **Ctrl-a or <home>** - go to beginning of line
- **Ctrl-e or <end>** - go to end of line
- **Ctrl-u** - erase all characters backward
- **Ctrl-k** - erase all characters forward
- **Alt-f** - move forward one word
- **Alt-b** - move backward one word
- **select or select + ctrl-<insert>** - copy to clipboard
- **<insert> or shift-<insert>** - paste from clipboard

Note about command history: if you type the beginning of a command and then use the up/down arrow key, you will see all previous commands starting with this string

### 3.6 Piping

Some commands support piping, e.g. All OSE shell commands, lh, tg, str, etc.

This is usually indicated in the menu and the help for that command. Some examples are:

```
te log read | grep ERROR
lh mp te log read | grep ERROR
str | grep cell=30456
```

For other commands that don't support piping (like MO commands), the workaround is to save the output to a logfile then run the unix command on that logfile by using the `l` or `!` command. Example:

```
l+                #open the logfile, an arbitrary name will be given
prod loadmodule   #run the command
l-                #close the logfile
l sort $logfile    #run unix command sort on the logfile.
l grep -i basic $logfile
```

Note: `$logfile` is automatically set by MoShell to contain the name of the latest log file created.

## 4 Command descriptions

Here all the commands and their syntax which are possible using Moshell are supported. Each of the OSE shell commands are not mentioned specifically, but it is possible to run all of them through Moshell.

### 4.1 Basic MO commands

#### 4.1.1 mom[abcdfloprtuxs] [<moclass/struct/enum>] [<attribute/action>] [<attr-type>] [<attr-flags>] [<description>]

Print description of *MO Classes, CM/FM Attributes, Actions, Enumerations and Structures*.

Regular expressions can be used in the various filters. There are five levels of filtering, as shown in the command syntax.

Options:

- **a** : show only the definitions relating to application MOs
- **b** : shows the default attribute values.
- **c** : show all the MO classes specified in the filter as well as their children/grandchildren/etc classes.
- **d** : gives a shorter printout, without the description part.
- **f** : shows the attribute flags (only applies when combined with options "b", "r", "l")
- **l** : shows the attribute value lengths.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- **p** : show only the definitions relating to platform MOs (CPP)
- **r** : shows the valid attribute value range.
- **x** : show unidirectional and bidirectional MO relationships
- **t** : show MO containment relationships and cardinality. The command momt shows three tables: first table with all valid LDNs, second table with parent MO(s), and third with children MO(s). The first argument can be used to show only MO classes matching the filter. The options o, u, l (momto/momtu/momtl) can be used to show only certain specific tables: momtl for the LDN table, momto for the parent table (MOs over), and momtu for the children table (MOs under). The option s is used to show the systemCreated flag in the LDN table and parent table (eg: momts / momtls).

Some of the options can be combined, see examples below.

After execution of the MOM command, two scripting variables are automatically created:

- `$moclass_filer` : contains a regular expression matching all MO classes that were printed by the command
- `$attrib_filter` : contains a regular expression matching all attributes that were printed by the command

These variables can then be used in the "get" command to read attributes matching certain MOM conditions based on the attribute name, data type, flag, or description text. In kget command, it is possible to specify the MOM conditions directly from the kget command arguments.

Examples:

1. momt - View the whole MO tree
2. momt atmp - View all possible parents and children of the **AtmPort** MO
3. mom atmp - View a description of the **AtmPort** MO
4. momcd atmp - List all MO classes under the **AtmPort** MO
5. momc atmp - View a description of all MO classes under the **AtmPort** MO
6. mom vcl - View a description of the MO class **VclTp**
7. mom vcl . - View a description of all attributes of MO class **VclTp**
8. momd . restart - List all attributes and actions matching the word *restart*
9. momd . . struct - List all attributes of type struct and/or all actions containing struct parameters
10. momd . . . restricted - List all attributes that have the restricted flag
11. momd utrancell . . !restricted|readonly - List all utrancell attributes that do not have the restricted or readonly flag
12. momd . . . . license - List all MOs, attributes and actions whose description contains the word *license*
13. mom . . . . license - View the description of all MOs, attributes and actions whose description contains the word *license*
14. momd restart . - List all struct or enumerates matching the word *restart*
15. mom restart . - View the description of all struct or enumerates matching the word *restart*
16. momd . . enumref:admst - List all attributes of type *enumRef:AdmState*
17. mom adminproductda . - View a description of all struct members contained in struct *AdminProductData*
18. momd . . sequence:moref restricted - List all attributes of type *sequence:moRef* who have a flag *restricted*
19. momb utrancell - List the default values for all attributes in the MO class *UtranCell*
20. mombf utrancell . . !restricted - List the default values for all UtranCell attributes that do not have the flag *restricted*
21. mombr . power|pwr - List the default values and valid ranges for all attributes that match the word *power* or *pwr*
22. momx - Show the relationships between MO classes
23. momx iublink - Show the relationships to and from *IubLink*
24. momx iublink.\*utrancell|utrancell.\*iublink - Show the relationships between *IubLink* and *UtranCell*
25. momx reservedby - Show the relationships connected via *reservedBy* attribute
26. mom . . ^moref, then get . \$attrib\_filter - Print attribute values for all attributes of data type *moRef*



Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

#### 4.1.2 lt/clt/ltc[1-9] <motype-filter>|root|all [<attribute==value> AND/OR <attribute==value>]

Load MO tree (full or partial) and build proxy table.

lt stands for *Load MO Types*, clt stands for *Conditional Load mo Types*, ltc stands for *Load MO Types and their Children*. The numeric option in ltc is for specifying the number of levels of children to load. Without the option, all levels of children are loaded.

This command queries the node to find out which MOs it contains and creates a table with the MO LDNs and a proxy number.

The first argument of the lt/clt/ltc command can be:

- *root* clear the proxy table and allocate a proxy for the root MO (ManagedElement)
- *all* build a proxy table with all MOs contained in the node.
- *all!<motype-filter>* build a proxy table with all MO's except some MO classes.
- *<motype-filter>* get a proxy for all MO types matching the specified pattern.

Examples are as follows:

1. `lt atmpor` - load all MOs of type matching the string "atmpor", this will usually be the **AtmPort** MOs
2. `clt atmpor` - conditionally load all MOs of type matching the string "atmpor". Loading only performed if no MOs of this type are already loaded.
3. `ltc equipm` - load the Equipment MO and all its children (all the way down)
4. `ltc1 equipm` - load the Equipment MO and only one level of children
5. `ltc2 transp` - load the Transport MO and two level of children
6. `lt ^utranCell|fach|rach|pch` - load all utranCells, fach, rach, pch MOs
7. `lt iub` - load all iublins
8. `lt all!relation` - load all MOs except the utranrelation/gsmrelation MOs.

The pattern in *motype-filter* is a regular expression, more information can be found with command `h syntax` and `h pr`

The argument *root/all* clears the proxy table, whereas `lt <motype-filter>` doesn't, so the MO LDNs get appended to the existing table.

If the same MO type is loaded several times, only the latest fetched instance is kept. Previously fetched instances of that MO type are deleted from the internal table.

The second argument (optional) is a filter constraint for the attribute value. Example:

1. `lt utranCell operationalState==0` - load proxys for all disabled cells
2. `lt utranCell primaryCpichPower==270` - load proxys for all cells that have pichpower=270
3. `lt all operationalState==0 OR administrativeState==0` - load proxys for all MOs in the node that have opstate 0 or admstate 0.
4. `ltc rncFunction operationalState==0` - load proxys for all MOs under RncFunction that have opstate 0.

Note: This type of search is very hard for the node if it has to search through a large number of MOs (ie several thousand).

For more information about Filter constraint, refer to Reference [?].

#### 4.1.3 lc/lcc[1-9] <moGroup>|<moFilter>|<proxy(s)>|all

Load MO tree (full or partial) and build proxy table.

The `lc` command is for loading the LDNs of the children MOs lying under an MO or group of MOs. `lc` on its own or combined with the numeric option "1" will only load the direct children. With the numeric options 2 to 9 it is possible to specify the number of levels of children to be loaded. The "c" option (`lcc` command) is for loading all possible levels of children under the MO.

The parameters of the command are:

- *all* - build a proxy table with all MOs contained in the node.
- *<proxy(s)>* - load children of one or several MO's given their proxy id. For example:
  - `lc 0` load the direct children of the ManagedElement MO (only one level of children)
  - `lc2 0` load two levels of children under ManagedElement MO
  - `lcc 0` load all levels of children of the ManagedElement MO (same as `lc all` or `lt all`).



Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- `lc 5 6 7` load children of proxys 5, 6, and 7.
- **<moFilter>** - loads children of all MOs whose RDN match the pattern. For example:
  - `lc3 transportnetwork=1` load three levels of children under TransportNetwork MO
  - `lc cell=3002` loads direct children for MOs whose RDN match **cell=3002**
  - `lcc ms-24-1` loads children of all MOs whose RDN match **ms-24-1**

The pattern in *mo-filter* is a regular expression, more information can be found with command `h syntax` and `h pr`.

#### 4.1.4 lu/llu <moGroup>|<moFilter>|<proxy(s)>

Unload MOs from MO tree.

The purpose is to reduce the size of the proxy table by unloading unwanted MOs. This is useful on large nodes with > 50,000 MOs. Memory usage on the workstation will be reduced and MO commands will be faster. The typical case is to unload all relation MOs in the RNC (UtranRelation and GsmRelation) which are very numerous but not used in most commands.

Example:

- `lt all`
- `lu relation`

#### 4.1.5 pr[s][m]/lpr[s][m] [<moGroup>|<moFilter>|<proxy(s)>] [<mimName>]

Print MO LDNs and proxy ids for all or part of the MO tree currently loaded in moshell.

Options:

- `s` (silent) : to print only the total number of MOs matched in each MO class.
- `m` (mim) : to print the mimName of each MO instance. Only applicable for COM/ECIM nodes.

Examples:

- `pr` - print all MOs
- `pr 0-1000` - prints the MOs with proxy id 0 to 1000
- `lpr subrack=ms` - print all MOs whose LDN match subrack=ms. This will print the MO Subrack=MS as well as all its children.
- `pr !utranrel` - print all MOs except those with an RDN matching *utranrel*

For further information see Section 3 or `h syntax`.

Note: The `pr` command is useful to test patterns used in mo-filters. For instance, some patterns will match more MOs than you expected, which will result in executing a command on some unwanted MO's. Therefore, it is good to first try your pattern on the `pr/lpr` command, then do it "for real" on a command that actually communicates with the node. The `pr/lpr` command also shows the total number of MOs matching the pattern.

#### 4.1.6 ma/lma <moGroup> <moGroup>|<moFilter>|<proxy(s)>|all [<attribute-filter>] [<value-filter>]

Add MO(s) to an MO group.

The first argument (mandatory) indicates the name of the MO-Group.

The second argument (mandatory) indicates the Proxy Id's or MO-Filter of the MOs to match.

If no further arguments are given then the MOs whose RDN/LDN match the MO-filter (or who have the corresponding proxy) will be put in the MO-Group.

If further arguments are given then a get or pget command is performed using the second/third/(fourth) argument of the `ma/lma` command.

The third argument will be a string to match the attribute and the fourth (if it's used) will match the value.

If the attribute is of any other type than MoRef, then the MO(s) whose attribute match the fourth argument will be put in the group.

If the attribute is of type MoRef, then the MO(s) contained in the attribute is put in the group (except if there is a fourth argument).

Refer to the following examples:

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

- `ma test atmpport.*24-1` - all MOs whose RDN match **atmpport.\*24-1** are put in the group *test*
- `lma test atmpport.*24-1` - all MOs whose LDN match **atmpport.\*24-1** are put in the group *test*
- `ma test atmpport operationalst 0` - all MOs whose RDN match **atmpport** and who have *operationalState* matching "0" will be put in the group *test*
- `ma test atmpport physpathterm` - all MOs who are referenced through the attribute **physpathtermld** of the MOs matching **atmpport** will be put in the group *test* (since **physpathtermld** is an attribute of type MoRef)
- `lma test subrack=ms,slot=10,program loadmodule` - all loadmodule MOs connected to program MOs running on slot 10 in main subrack will be put in the group *test* (since **loadmodule** is an attribute of type MoRef)
- `ma test atmpport physpathterm slot=23` - all atmpports whose physpathterm reference matches **slot=23** will be put in the group *test*
- `ma test reliableprogram admactiveslot slot=10` - all reliableprograms that are on slot 10 will be put in the group *test*
- `ma test vcltp pmrec 0` - all vcltps with 0 pmreceivedcells are put in the group

Two-step example: To put all unlocked-disabled MOs in a group in order to lock them:

```
ma test all operational 0
ma test1 test administ 1
bl test1
```

To put all cells belonging to module 3 in a group in order to lock them

```
ma iubmod3 iublink module =3$
ma cellmod3 iubmod3 reservedby
bl cellmod3
```

To put all vcltps that have 0 receivedcells and more than 0 transmitted cells in a group in order to find out which upper layers are affected

```
ma faultyvcltp vcltp pmrec ^0
mr faultyvcltp vcltp pmtrans ^[^0]
lk faultyvcltp
```

Note: it is also possible to create MO groups with the commands `hget/lhget`, `st/lst`, `lk/lk`, and `pdiff/lpdiff`.

#### 4.1.7 `mr/lmr <moGroup> <moGroup>|<moFilter>|<proxy(s)>|all [<attribute-filter>] [<value-filter>]`

Remove an MO group or remove MOs from an MO group (MOs will NOT be deleted, only the group).

Exactly the same syntax as `ma` described in Section 4.1.6 except that the MOs matching will be removed from the MO-Group instead of added.

#### 4.1.8 `mp`

Print all defined MO groups. See `ma` command in Section 4.1.6 for more info about MO groups.

Note: To print the contents of a group, use the `pr <mo-group>` command.

#### 4.1.9 `get/lget [<moGroup>|<moFilter>|<proxy(s)>|all] [<attribute-filter>|all] [<value-filter>]`

Read CM/FM attribute(s) from MO(s).

Note: to read PM attributes, use `pget/lpget` (see Section 4.4.3).

Examples:

1. Get all attributes from all MOs except those whose RDN matches *utranrel* or *iub*
  - `get !(utranrel|iub)`
2. Get all attributes from MOs whose proxies range from 10 to 30
  - `get 10-30`

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

3. From all MOs, get productdata containing the string 0843 (can be useful to find out which MO's are related to a certain loadmodule)
  - `get all product 0843`
4. From all Utrancells, get sintersearch attribute different from 10
  - `get utrancel sinter !10`
5. Get traffic descriptors for all VclTp(s)
  - `get vc trafficdes`
6. Get VcpTp(s) used by all cross-connections
  - `get cross vc`
7. Get piutype for all MO's under "Equipment"
  - `lget equip piutype`
8. get attenuation values for all cables (in RBS)
  - `get cable atten`
9. get all attributes in nodesynch MO, belonging to site 1164 (in RNC)
  - `lget 1164,nodesy`
10. get all attributes in MO's under "RadioNetwork" matching pwr or power or sir
  - `lget radion p.*w.*r|sir`
11. view which cells are connected to which iub's (in RNC)
  - `get cell iub`
12. get all attributes from nodebfunction MO, except those matching "overload"
  - `get nodebfunc !overload`

#### Important information:

1. when doing a `get <mo(s)>` or `get <mo(s)> all`, one CORBA request is sent for each MO, asking for all attributes of that MO.
2. when doing a `get <mo(s)> <attribute(s)>`, a CORBA request is sent for up to 100 MOs at a time, asking for the specified attributes of those MOs.

The implications of this is that it is faster but:

1. if one or more attributes cannot be read due to some exception (eg: fRO not accessible), then all attributes will return the same exception, even if they can be read.
2. if one or more MOs contain one or more attributes that cannot be read, then all MOs within that CORBA request will return the same exception even if they can be read.

The workaround for the first problem is to find out which attribute is causing the problem. The command `sget/lsgget` can be used for this. The `sget` command reads each attribute one by one. The attribute(s) that is/are causing the exception(s) will then be easy to spot.

It is then possible to use the standard "get" command with with the negative filter (!) to exclude the "faulty" attribute.

- `get nodebfunction !overload` - all attributes of the nodebfunction MO except those matching "overload" will be read.
- `pget utrancell !pmnoofrrc` - all pm attributes of the utrancell MO except those matching "pmnoofrrc" will be read.

The workaround for the second problem is to lower the speed of reading so that only one MO instead of 100 is read per CORBA request.

This is done using the `speed` command.

Example: `st all` - one MO is returning an exception which means that up to 100 MOs cannot be read. Instead do:

```
speed 1
st all
speed 100
```

The command will be slower but the exception will only affect the MO(s) that have it and not the "healthy" ones.

The speed command affects `get`, `pget`, `kget`, `prod`, and `st` commands.

Prepared (also subject responsible, if other) EAB Finn Magnusson		No. 1553-CXC1328930		
Document responsible/Approved EAB Finn Magnusson	Checked	Date 2015-04-19	Rev. R	File moshellUserGuide.tex

By default, `speed` is set to 100, which means that up to 100 MOs share the same CORBA request.

By running the command `speed 1`, the exception will not affect the other MOs. However the speed will be slower. It is possible to use a value from 1 to 200 to define the speed. It is recommended to not use a speed higher than 100 since this takes more memory from the node.

Type `speed` on its own to see the current speed.

**Scripting and variable assignment with `get`** It is possible to store the output into a variable

Example:

1. Store one value into a variable
  - `get utrancell pich > $pich`
2. Store many values into an array
  - ```
for $mo in utrancellgroup
    $mordn = rdn($mo)
    get $mo pich > $pichTable[$mordn]
done
```

Refer to the Section 6 (Scripting Chapter) for more information.

#### 4.1.10 `hget[c][m]/lhget[c][m] <moGroup>|<moFilter>|<proxy(s)> [<attribute-filter>] [<value1-filter>] [<value2-filter>] [<value3-filter>] etc...`

Read CM/FM attribute(s) from MO(s), print horizontally one line per MO (instead of one line per attribute).

Options:

- `c`: display the output in CSV format for easier export to excel (for instance).
- `m`: print all MOs in a single table instead of separate tables per MO class

Example:

- `hget reliableprogramunit slot|operational` print the RPU attributes `admActiveSlot`, `admPassiveSlot` and `operationalMode`
- `hget reliableprogramunit slot|operational slot=10 . ^2` print all RPUs that are defined on slot10 (active), any slot for passive, and 2 for the operationalmode
- `hget reliableprogramunit slot|operational !slot=10 . !^1` print all RPUs that are not defined on slot10 (active), any slot for passive, and operationalmode is not equal to 1
- `hget loadmodule type|productdata` print the attributes `loaderType` and `productData` on all LoadModule MOs. Note that `productData` is a struct containing 5 members so the `productData` attribute will take up 5 columns
- `hget loadmodule type|productdata@name` only print the attribute `loaderType` and the structmember `productData:productName` passive, and `operationalmode` is not equal to 1
- `hgetm port state` print all attributes matching the word "state" on all MOs matching the word "port" and display all lines in one single table instead of a separate table per MO class.

For "slow" `hget`, use "shget/lshget": reads only one attribute at a time.

#### 4.1.11 `kget/lkget [<moGroup>|<moFilter>|<proxy(s)>] [<attribute-filter>] [<attribute-type>] [<attribute-flag>] [<attribute-description>]`

Display CM/FM attributes in exportable printout format.

Same as `get/lget`, but with a different output format to allow import of the dump into external tools like MCOM, CCT, ETRAN.

For "slow" `kget`, use "skget/lskget": reads one attribute at a time.

The 2nd to 5th arguments have the same meaning as the arguments used in the "mom" command.

Examples:

- `kget` : print all MO attributes
- `kget !relation=` : print all attributes except from MOs with RDN matching "relation="
- `kget . . moref` : print all attributes of data type matching "moref"

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `kget . . . ericsson` : print all attributes with flag matching "ericsson"
- `kget . . . . dbm` : print all attributes with description matching "dbm"

#### 4.1.12 `fro/lfro[m] <moGroup>|<moFilter>|<proxy(s)>|all [<attribute-filter>|all] [<value-filter>]`

Read MO persistent data from node database via SQL.

The `fro/lfro` command reads only the frold connecting the MAO and FRO parts of the MO.

The `from/lfrom` command reads all MAO/FRO data of the MO.

The command syntax and printout format is identical to that of the `get/lget` command.

Examples:

- `from upgradepackage=cxp9012014_r5a - print MAO/FRO persistent data for MO UpgradePackage=CXP9012014_R5A`
- `from 0 - print MAO/FRO persistent data for the ManagedElement MO`
- `fro plugin - print frold for all pluginunit MOs`
- `fro plugin . 5 - print the PlugInUnit MOs who have a froid matching the value 5`
- `fro plugin . ^5$ - print the PlugInUnit MOs who have a froid equal to 5`
- `lfro ms,slot=10,plugin sairesource > $sai - save the sairesourceid of a particular pluginunit into a variable $sai`

Note: If the SQL client LM (CXC1325608) is not started, the `fro[m]` command starts it automatically using the "sql+" command. After the session, the SQL client should be turned off using the "sql-" command. Type "h sql+" for more info.

#### 4.1.13 `sql/select <command> [ | <unix-cmds>]`

To run a SQL command while in db.dat mode. The db.dat or cv.zip is loaded with moshell option "-d".

Examples:

- `select * from tables | grep pgm`
- `select * from tables where name like '%iur%'`
- `select * from cspgmresource_01 where pno='CXC 132 0784'`

#### 4.1.14 `st/lst <moGroup>|<moFilter>|<proxy(s)>|all [<state-filter>]`

Print state of MOs (operationalState and administrativeState when applicable).

It is similar to writing `get/lget <mo> state`, the only difference is that it presents the two states side-by-side in a more visible way.

The state filter matches towards both the Operational state and the Administrative state.

Examples:

- `st - view state of all MOs`
- `st all dis - view all disabled MOs`
- `lst equip dis - view all disabled MOs under "Equipment"`
- `st all 1.*0 - view all MOs which are unlocked and disabled`
- `st all ^0 - view all MOs which are locked:`
- `lst sector - view state of all MOs under "Sector" (in RBS)`
- `lst cell - view state of all channels in all 3 cells in the RBS`
- `lst cell=120 - view state of all channels in cells starting with 120 (in RNC)`

#### 4.1.15 `prod <moGroup>|<moFilter> [<productdata-filter>]`

Print the attribute productData on applicable MO(s).

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

It is similar to typing "hget <mo> productdata". This command prints *product data* of all MO(s). It is similar to typing `get all productdata`, except that the *productData* appears in one row. It is possible to filter only MOs matching a certain product identity. Examples:

- `prod loadmodule cxc1320784` - print all MOs matching "loadmodule" and where the value of productData attribute matches "cxc1320784"

MO classes that have a productdata attribute can be found via `mom` command:

```
mom all all struct:.*productdata
```

Typically, this includes *Slot*, *Subrack*, *PiuType* and *LoadModule*.

#### 4.1.16 lk/lk <moGroup>|<moFilter>|<proxy(s)>

View all MO(s) linked to an MO, and their states (*administrativeState* and *operationalState*).

Examples:

- `lk mtp3bsrs=` - View all core network interface stacks
- `lk ranap=cs` - View all MO's linked to Ranap=cs
- `lk iublink=iub-12` - View all MO's linked to iub 12
- `lk atmport=ms-24-1` - View all MO's linked to atmport MS-24-1 (and its VclTp's)

#### 4.1.17 lko/lko <moGroup>|<moFilter>|<proxy(s)>

The old `lk`. Obsolete command, use `lk/lk` instead.

Output format of e.g. `lko ranap=cs`:

```
=====
MOs linked to 1316 RncFunction=1,Ranap=cs (-,E)
=====
localSccpApRef      Ranap=cs          --> (-,E) SccpSp=1,SccpSrc=1,SccpAp=1
sccpGlobalTitleRef  Ranap=cs          --> (-,-) SccpSp=1,SccpSrc=1,SccpGlobalTitle=1
remoteSccpApRef     Ranap=cs          --> (-,E) SccpSp=1,SccpSrc=1,SccpAp=2
reservedBy          SccpSp=1,SccpSrc=1,SccpAp=1 --> (-,-) SccpSp=1,SccpSrc=1,SccpEntitySet=1
mtp3bApId           SccpSp=1,SccpSrc=1,SccpAp=2 --> (-,E) Mtp3bSp=1,Mtp3bAp=1
reservedBy          SccpSp=1,SccpSrc=1,SccpAp=2 --> (-,-) SccpSp=1,SccpSrc=1,SccpEntitySet=2
routeSetId          Mtp3bSp=1,Mtp3bAp=1 --> (-,E) Mtp3bSp=1,Mtp3bSrs=1
reservedBy          SccpSp=1,SccpSrc=1,SccpEntitySet=2 --> (-,-) SccpSp=1,SccpSrc=1,SccpGlobalTitle=2
slsReservedBy       Mtp3bSp=1,Mtp3bSrs=1 --> (-,E) Mtp3bSp=1,Mtp3bSls=1
reservedBy          Mtp3bSp=1,Mtp3bSls=1 --> (-,E) Mtp3bSp=1,Mtp3bSrs=1,Mtp3bSr=1
nniSaalTpId         Mtp3bSp=1,Mtp3bSls=1,Mtp3bS1=2 --> (-,E) NniSaalTp=csb
nniSaalProfileId    NniSaalTp=csb --> (-,-) NniSaalProfile=1
aal5TpVccTpId       NniSaalTp=csb --> (-,E) Aal5TpVccTp=csb
processorId         Aal5TpVccTp=csb --> (U,E) Subrack=MS,Slot=9,PlugInUnit=1
vclTpId            Aal5TpVccTp=csb --> (-,E) AtmPort=MS-7-1,VplTp=vp12,VpcTp=1,VclTp=vc34
atmTrafficDescriptorId AtmPort=MS-7-1,VplTp=vp12,VpcTp=1,VclTp=vc34 --> (-,-) AtmTrafficDescriptor=U3P4500M3000
nniSaalTpId         Mtp3bSp=1,Mtp3bSls=1,Mtp3bS1=1 --> (-,E) NniSaalTp=csa
nniSaalProfileId    NniSaalTp=csa --> (-,-) NniSaalProfile=1
aal5TpVccTpId       NniSaalTp=csa --> (-,E) Aal5TpVccTp=csa
processorId         Aal5TpVccTp=csa --> (U,E) Subrack=MS,Slot=8,PlugInUnit=1
vclTpId            Aal5TpVccTp=csa --> (-,E) AtmPort=MS-6-1,VplTp=vp11,VpcTp=1,VclTp=vc34
atmTrafficDescriptorId AtmPort=MS-6-1,VplTp=vp11,VpcTp=1,VclTp=vc34 --> (-,-) AtmTrafficDescriptor=U3P4500M3000
=====
```

In the middle column is the originating MO. In the far right column is the referenced MO. In the left column is the attribute containing the referenced MO. The letters in brackets show the *administrativeState* and *operationalState* of the referenced MO:

- **U** = Unlocked
- **L** = Locked
- **E** = Enabled
- **D** = Disabled
- **-** = Not Applicable

#### 4.1.18 set[m][c][1]/lset[m][c][1] <moGroup>|<moFilter>|<proxy(s)> <attribute> [<value>]

Set an attribute value on one or several MO's.

Only attributes that do not have the flag `readOnly` or `restricted` can be set. Use the `mom` command to check the flags of an attribute. For `restricted` attributes, it is possible to use the `rset` command.

Options:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **m** : for setting several attributes simultaneously in the same MO, using a single transaction for all attributes in each MO. Needed for changing certain attributes in the RNC UtranCell (locationAreaRef, uarfcnDI, uarfcnUI).
- **1** : for setting one attribute on many MOs simultaneously, using a single transaction for all the MOs. Needed for changing certain attributes in the eNB EUTRANCell (tac, frameStartOffset, subframeAssignment, specialSubframePattern).
- **c** : for setting an attribute to its current value. When using this option, the attribute value shall not be specified since the existing attribute value is used.

Note: These options cannot be combined.

Examples:

- `set cell primarycpichpower 250` - set primarycpichpower to 250 on all cells (in RNC)
- `lset uerc= sirmin 60` - set sirmin to 60 on all MO's under uerc (in RNC)
- `lset ms,slot=1,pluginunit=1$ userlabel` - set an empty value for the userLabel of this PluginUnit
- `set mtp3bspitu sppriority prioslt=2` - set an attribute of type struct
- `setc iublink= preferredsubrackref` - set the preferredsubrackref attribute to its current value on all Iublinks, in order to evenly spread the sites across all modules.
- - setting three attributes simultaneously on a MO (split on several lines for readability)  

```
setm utrancell=3012 locationarearef locationarea=9
    servicearearef locationarea=9,servicearea=1
    routingarearef locationarea=9,routingarea=1
```
- `set1 ^eutrancellfdd= tac 1280` - set the tac attribute simultaneously on all EUTRANCellFDD in the eNB.

Please see Section 3 and specifically Section 3.4 for more information on how to set values for each specific attribute type (e.g. **Struct**, **array of MoRef**, **array of Struct**, etc).

#### 4.1.19 `eset[c][1]/leset[c][1] <moGroup>|<moFilter>|<proxy(s)> <attribute-filter> [<value>]`

Set one or several attributes on one or several MO's, using regexp matching on the attribute name.

Same as `set` command except that the second argument uses regular expression matching on the attribute name so all attributes whose name match the filter will be affected by the operation. Refer to the help of the `set` command for more information about syntax and command options `c` and `1`.

Examples:

- Activate all features in the RBS/ERBS. The attribute is called `featureStatexxx`, where `xxx` is the name of the feature. All MOs that have an attribute matching the word "featurestate" will have that attribute set to 1  

```
>> eset . featurestate 1
```
- Change the `ENodeBFunction::eNodeBPlmnId` on ERBS. The `EUTRANCellFDD::bPlmnList` must be changed at the same time in one transaction.  

```
>> eset1 ^enodebf|^eutrancellfdd plmn mcc=240,mnc=99,mnclength=2
```

#### 4.1.20 `rset/lrset <moGroup>|<moFilter>|<proxy(s)> <attribute> [<value>]`

Set attribute value on a restricted attribute or change the MOid of an MO.

A *restricted attribute* is an attribute that can only be set when the MO is created.

The `rset` command works by doing `rdel/lrdel` on the MO and recreating all previously deleted MOs using the new attribute value.

To change the MOid of an MO, the attribute name should be made up of the MOclass followed by "id", eg: `atmportid`, `pluginunitid`, etc.

Example 1, change a restricted attribute:

```
rset unisaaltp=.*1004 unisaalprofileid unisaalprofile=win30a
```

Example 2, change the MOid:

```
rset utrancell=cell123 utrancellid cell1456
```



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.1.21 bl[s]/lbl[s] <moGroup>|<moFilter>|<proxy(s)>

Lock or soft-lock MO(s).

Works by setting the `administrativestate` to 0 (hard-lock) or 2 (soft-lock).

The `s` option is for *soft-lock*. The `administrativestate` is set to 2 ("shutting down") which means that the resource will have a grace period to handover traffic to other resources, before it gets locked. The `administrativestate` will automatically go over to 0 after the grace period, which can be between a few seconds to a couple of minutes, depending on the MO type.

The `administrativeState` will automatically go over to 0 after the grace period of around 30 seconds.

Without the `s` option, the resource is locked immediately.

To unlock an MO, use the command `deb/ldeb`.

Examples:

1. `bl aal2.*ca[246] - block aal2paths ca2, ca4, ca6`
2. `lbl subrack=ms,slot=19 - block all MO(s) under subrack=ms,slot=19`
3. `bl 234 256 248 - block proxys 234, 256, and 248`
4. `bl 001500 - block a board. Same as lbl subrack=ms,slot=15,pluginunit=1$.`
5. `bls 001500 - soft-block a board.`

Note: there is also an OSE command called `bl`. If you need to run the OSE command instead of the moshell command, just type a `"\"` (backslash) in front. E.g.: `\bl`

#### 4.1.22 deb/ldeb <moGroup>|<moFilter>|<proxy(s)>

Unlock MO(s).

Works by setting the `administrativestate` to 1.

To lock an MO, use the command `bl/lbl`.

1. `ldeb subrack=ms,slot=19 - deblock all MO(s) under subrack=ms,slot=19`
2. `deb 001900 - deblock a board. Same as ldeb subrack=ms,slot=19,pluginunit=1$)`

#### 4.1.23 acl/lac <moGroup>|<moFilter>|<proxy(s)>|all [<action-filter>]

Lists available MO actions.

Examples:

1. `acl all restart - View all restart actions`
2. `acl all [^(restart)(eteloopback)] - View all actions except restart and eteloopback`
3. `acl sync - View all actions related to the synchronisation MO. Output:`

| Proxy | MO                | Action                | Nr of Params |
|-------|-------------------|-----------------------|--------------|
| 396   | Synchronization=1 | changeSyncRefPriority | 2            |
| 396   | Synchronization=1 | removeSyncRefResource | 1            |
| 396   | Synchronization=1 | resetLossOfTracking   | 1            |
| 396   | Synchronization=1 | addSyncRefResource    | 2            |

4. `acl all listrou - Find the MO with action matching regular expression listrou:`

| Proxy | MO                    | Action     | Nr of Params |
|-------|-----------------------|------------|--------------|
| 471   | Ip=1,IpRoutingTable=1 | listRoutes | 0            |

#### 4.1.24 acc[e]/lacc[e] <moGroup>|<moFilter>|<proxy(s)>|all <action>

Execute an MO action.

If the action requires parameters, these will be prompted for. If no value is entered at a prompt, the action is aborted. In order to be avoid being prompted for the parameters, use the action `facc/lfacc`, then the parameters can be given on the same line as the command.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Examples:

1. Restart a board:

- `acc 001400 restart - Same as:`  
`lacc subrack=ms,slot=14,pluginunit=1$ restart`
- `acc 001900/sp0.lnh restart - Same as:`  
`lacc subrack=ms,slot=19,.*,spm=1$ restart`

2. Restart the node:

- `acc 0 manualrestart`

3. List the IP routing table:

- `acc ip listroutes`

4. Perform End-To-End loopback on VclTp MOs:

- `lacc atmport=ms-24-1 eteloopback`

5. Perform PhaseMeasurement on NodeSynchTp MOs:

- `acc nodes performPhaseMeasurement`

6. Add a new synchronization reference:

- `acc sync addSyncRefResource`

Note 1: To specify an attribute of type *Struct*, use the following syntax:

- `attr1=val1,attr2=val2,attr3=val3...`

This is the same syntax as used in `set` and `cr` commands, and is not case sensitive. Example (in the case of `routingTableEntry` in action **deleteStaticRoute**) (note: line split for readability):

```
destinationIpAddress=10.1.10.0,destinationNetworkMask=255.255.255.0,
nextHopIpAddress=10.128.15.1,routeMetric=3
```

Alternatively, the option `e` can be used, in which case each struct member is prompted on a separate line (command: `acce/lacce`).

Note2: Action `manualRestart` on `ManagedElement` MO

- This action can be restricted with the uservariable `restart_confirmation`. See description in the file `moshell/moshell`.
- Node restarts usually result in decrementing of the attribute `ConfigurationVersion::rollbackCounter` which leads to node rollback upon reaching zero. However the action `manualRestart` on `ManagedElement` is a special case which does not lead to decrement of the `rollbackCounter`, when executed from `moshell/AMOS`.

#### 4.1.25 `cr[e][n] <ldn>`

Create an MO.

When run without option, the command will prompt the user to enter the values for all mandatory and restricted attributes. The order in which the attributes are prompted is the same order in which they are listed inside the XML MOM file. Restricted attributes are optional, type `d` to select default value. To abort, type `<enter>` at the prompt.

Options:

- `e` : to receive a separate prompt for each individual struct member, useful when inputting attribute values of type struct
- `n` : allow the user to enter any attribute name and value. Each attribute name and value shall be entered on a separate line with space between the attribute name and value. Any attribute can be specified, does not have to be a mandatory or restricted attribute. Type "end" once all attributes have been entered.

Examples:

```
cr swmanagement=1,upgradepackage=FAB102572%2_R14D.xml
cr rncfunction=1,iublink=2456
cre swmanagement=1,loadmodule=CXC123456_R9A
crn rncfunction=1,iublink=90
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

**Notes:**

To specify an empty attribute value, type `null` or `d`.

To specify an attribute of type *Struct*, use the following syntax:

```
attr1=val1,attr2=val2,attr3=val3...
```

This is the same syntax as used in `set` command, and is not case sensitive. Example, in the case of `productdata` in `loadmodule` (note: line split for readability!):

```
productnumber=CXC1322155/2,productrevision=R3C08,productname=test,
productinfo=test,productiondate=20010229
```

Alternatively, the option `e` can be used, in that case, each struct member is prompted on a separate line (command: `cre`).

An moshell script for MO creation can be automatically generated by using the simulated undo mode: `u+s, del <mo>, u-`, then edit the `undocommandfile` with the required values. By default the undo script will contain `crn` commands. To use `cr` commands instead of `crn`, do `uv use_crn=0`.

By default, the mandatory/restricted attributes that are marked as `deprecated` are not prompted by the `cr` command. This behaviour can be changed by setting the uservariable `exclude_deprecated` to 0.

As in the `ld` command, the LDN doesn't need to contain `ManagedElement=1`.

The mo-types are not case sensitive but the mo-id is.

**4.1.26 del[b]/ldel[b] <moGroup>|<moFilter>|<proxy(s)>**

Delete MO(s).

An MO can only be deleted when its `reservedBy` list is empty and when it does not have any children. If the MO does have children and/or a non-empty `reservedBy` attribute, it is possible to use the `rdel/lrdel` command instead.

The command first prints the MO(s) to be deleted, then asks for confirmation. Once the MO(s) are deleted, they are also removed from the proxy list.

To delete all MOs that have just been created by a script, just check the range of proxies that were created and delete them in the reverse order. This can be done easily by using the reverse proxy order.

Example: proxies 22 to 46 were created by a *CREATE* script (in `run` or `trun`). To undo it:

```
del 46-22
```

To delete an MO and all its children, just use the `%` sign in front of the `ldn/rdn` filter. Example:

```
ldel %ms,slot=20,plug will delete the programs first then the piu
```

To delete an MO group in the reverse order of the proxies, put the `%` sign in front of the MO group name. Example:

```
del %mymogroup - will delete all MOs of the MO group "mymogroup" in the reverse order of their proxies
```

**Note:**

The following MOs can only be deleted while in state `LOCKED`: `PlugInUnit`, `EUtranCell`, `TermPointToENB`, `GpsReceiver`, `IpAccessHostEt`, `IpAccessHostGpb`, `IpAccessHostSpb`, `IpSyncRef`.

By default, the "del" command will automatically perform the locking prior to deleting, unless the "b" option has been used (`delb`).

**4.1.27 rdel/lrdel <moGroup>|<moFilter>|<proxy(s)>**

Delete MO(s) together with children and reserving MOs.

For MO classes shown in the list below, the command finds out all the related MOs, then presents the list of MOs to be deleted and asks the user for confirmation. For all other MO classes, the command acts in the same way as a regular "del" operation, ie, just tries to delete the MO itself.

List of MO classes for which `rdel` acts recursively:

- `Aal0TpVcctp`
- `Aal1TpVcctp`
- `Aal2Ap`
- `Aal2PathDistributionUnit`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- Aal2PathVcctp
- Aal2RoutingCase
- Aal5TpVccTp
- AntennaNearUnit
- AtmCrossConnection
- AtmPort
- Cdma2000Cell
- Ds0Bundle
- EUtranCellFDD
- EUtranCellTDD
- EutranFrequency
- ExternalCdma2000Cell
- ExternalCdma20001xRttCell
- ExternalENodeBFunction
- ExternalEUtranCellFDD
- ExternalEUtranCellTDD
- ExternalGeranCell
- ExternalGsmCell
- ExternalUtranCell
- ExternalUtranCellFDD
- ExternalUtranCellTDD
- ImaGroup
- IpEthPacketDataRouter
- IuLink
- Iub
- IubLink
- IurLink
- M3uAssociation
- Mtp2Tptlu/Ansi/Ttc/China
- Mtp3bSIltu/Ansi/Ttc/China
- Mtp3bSIs
- Mtp3bSrs
- NbapCommon
- NbapDedicated
- NniSaalTp
- NodeSynchTp
- PacketDataRouter
- Ranap
- Rnsap
- SectorEquipmentFunction
- UniSaalTp
- UtranCell
- VclTp
- Vmgw

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- VpcTp
- VplTp

#### 4.1.28 gm[c][d]/lgm[c][d] <moGroup>|<moFilter>|<proxy(s)>

Generate MO Creation/Deletion script.

Options:

- c : Generate MO Creation script.
- d : Generate MO Deletion script.

The output of the command is a command file containing the creation/deletion commands.

By default the create statements will use the `crn` command but it is possible to use the `cr` command by setting `uv use_crn=0`

It is possible to convert the command file to `trun` format with the `u!` command.

Examples:

- `lgmc equipment=1` : Generate a MO script to create all the MOs under Equipment=1 . Parent and referenced MOs will be created before children and reserving MOs.
- `gmd subrack=1,slot=2,pluginunit=1` : Generate a MO script to delete the MO Subrack=1,Slot=2,PlugInUnit=1 . Children and reserving MOs will be deleted before parent and reserved MOs.

#### 4.1.29 safe+/safe-/safe?

Purpose: Apply strict MO matching rules on MO WRITE commands

- `safe+` : apply safe MO syntax, no proxy id or no regular expression matching for MO WRITE commands, exact MO name(s) must be given (case insensitive is allowed)
- `safe-` : allow proxy id and regular expression matching (this is the default setting)
- `safe?` : check the current setting

List of MO WRITE commands: `bl`, `bls`, `deb`, `del`, `rdel`, `fdel`, `set`, `eset`, `rset`, `fset`, `acc`, `facc`

Examples:

- `bl utrancell=11` - will lock cell=11 only (cell=111, or cell=11a , etc. would not be affected)
- `bl utrancell=11|utrancell=12|utrancell=13` - will lock cell=11, cell=12, cell=13
- `bl utrancell=11|12|13` - same as above
- `bl utrancell=11|12|13|iublink=1` - will lock cell=11/12/13 and iublink=1
- `bl pluginunit=1` - nothing will happen as there are several MO instances with this name
- `bl subrack=ms,slot=28,pluginunit=1` - will lock this MO as the name given is unique

The setting is off by default but can be saved to "on" by adding the line `safe_syntax=1` in the file `moshell/jarxml/moshellrc` or the file `./moshellrc`

#### 4.1.30 s+/s++/s-/s?

Purpose: Sort MO list in alphabetical order instead of proxy order.

- `s+` : activate alphabetical sorting of the MO list by order of LDN (`sort_proxy=1`)
- `s++` : activate alphabetical sorting of the MO list by order of MO class (`sort_proxy=2`)
- `s-` : go back to default behaviour where the MO list is sorted by proxy number (`sort_proxy=0`)
- `s?` : check if alphabetical sorting is active or not

Once the `s+/s++` command has been entered, the alphabetical sorting takes effect on all subsequent MO commands such as, `pr`, `get`, `set`, `st`, etc.

Type `s-` to revert to the default behaviour of sorting by proxy number.

To change the default behaviour, it is possible to use the uservariable `sort_proxy`, eg, adding the line `sort_proxy=1` in the file `~/moshellrc`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Example:

Default behaviour: MOs are displayed in the order of the proxy numbering. The order of the proxy numbering depends on the order with which the MOs were created on the node.

```
RNC32> pr uerc=
=====
Proxy  MO
=====
2587  RncFunction=1,UeRc=0
2591  RncFunction=1,UeRc=16
2604  RncFunction=1,UeRc=91
2620  RncFunction=1,UeRc=1
2628  RncFunction=1,UeRc=17
2643  RncFunction=1,UeRc=70
2644  RncFunction=1,UeRc=59
2670  RncFunction=1,UeRc=2
```

Activate MO list sorting and run some MO commands. MOs now appear in alphabetical order.

```
RNC32> s+
Proxy sorting: activated.
```

```
RNC32> pr uerc=
=====
Proxy  MO
=====
2587  RncFunction=1,UeRc=0
2620  RncFunction=1,UeRc=1
2670  RncFunction=1,UeRc=2
2754  RncFunction=1,UeRc=3
4071  RncFunction=1,UeRc=4
....<cut>....
```

```
RNC32> hget uerc= userlabel
```

```
=====
MO      userLabel
=====
UeRc=0  Idle
UeRc=1  SRB (13.6/13.6)
UeRc=2  Conv. CS speech 12.2
UeRc=3  Conv. CS unkn (64/64)
UeRc=4  Interact. PS (RACH/FACH)
UeRc=5  Interact. PS (64/64)
UeRc=6  Interact. PS (64/128)
```

#### 4.1.31 u+[s]/u-/u? [<file>]

Handling of undo mode (for cr/del/rdel/set/bl/deb/acc commands). Can be used for generation of MO scripts as well.

- u+ To start the *undo mode*
- u+s To start the *simulated undo mode*
- u- To stop the *undo mode*
- u? To check if *undo mode* is active or not
- u! To convert moshell command files to trun/emas format (and vice-versa) or undo logfiles to command files, see description below for more info.

While running in "undo mode", the MO data is saved in a special logfile for all MOs on which the following commands are run:

- cr
- del/lldel/rdel/lrdel
- bl/lbl/deb/ldeb
- set/lset

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `acc/lacc`

Upon stopping of undo mode, an undo file is generated to revert the MO configuration changes. It can also be used for deleting and recreating MOs when one needs to change a restricted attribute.

The undo file will contain the following commands:

- `del` commands to remove created MOs.
- `cr` commands to put back deleted MOs.
- `bl/deb` commands to change MOs back to their original administrative state.
- `set` commands to change MOs back to their original attribute values.
- `acc` commands to revert certain MO actions. This only works on actions that have an opposite, see note below.

When running the simulated undo mode (`u+s`), all MO operations (`cr/del/rdel/bl/deb/set`) are simulated. Two command files are generated, one for deletions and one for creation.

The files generated by undo mode and simulated undo mode are stored in the following variables:

- `$undologfile`
- `$undodelcommandfile` (simulated undo mode only)
- `$undocommandfile`

Conversion functionality (**u!**): The **u!** command takes as input one of the files generated by the undo mode. Different output will be generated depending on the input file.

- if the input file is an undologfile, the output will be an undodelcommandfile and an undocommandfile
- if the input file is an undodelcommandfile or an undocommandfile, the output will be an undotrunkfile (`$undotrunkfile`) which is a command file in trun/emas format.
- if the input file is an undotrunkfile (trun/emas format), the output will be an undocommandfile (run format)

Note: to undo **create** commands run from a "trun" script, just run a delete on the proxy range in reverse order. See Section 4.1.26 or `h del` for more info.

Note: the undo mode currently cannot reverse a set command done on an attribute of type sequence of struct.

Note: the following actions are currently supported in the undo mode:

- `addPath/removePath`
- `addDhcpServerAddress/removeDhcpServerAddress`
- `setAutoActivate/setAutoDown`
- `activateRemoteSp/inactivateRemoteSp`
- `addRemoteSp/removeRemoteSp`
- `activate/deactivate`
- `localInhibit/localUninhibit`
- `activateLinkSet/deactivateLinkSet`
- `blockSignalingRoute/deBlockSignalingRoute`
- `addRepertoire/deleteRepertoire`
- `addSlot/deleteSlot`
- `addCicRange/removeCicRange`
- `addNri/removeNri`
- `addTdmTermGrpMos/removeTdmTermGrpMos`
- `addSyncrefResource/deleteSyncrefResource`
- `addAal2ApToRc/removeAal2ApFromRc`
- `writeSystConst/resetSystConst, deleteConst/writeConst`
- `changeFrequency, pnpChangeFrequency, setFrequencyBand`
- `removeIpAccessHostMos/addIpAccessHostMos`
- `manualMspSwitch, manualSwitch, switch`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.1.32 run[x][1] [-l <lineNr>] <command file> [<var1>] [<var2>] ...

Run a command file in moshell format.

The command file layout is the same as for *monode* and *mobatch*. See examples in **moshell/examples/mobatch\_files**.

It shall contain all lines to be sent to the moshell prompt, including password (for ose commands), but NOT confirmations (**y**). This applies to commands such as *lt/ltc*, *lc/lcc*, *del/lde1*, *bl/lbl*, *set/lset* where confirmation is automatically entered when running a command file.

Comments can be put in the command file using the # sign. By typing **<TAB>**, the unix file system is displayed, making it easier to find the location of the command file.

If some arguments are given after the command file, the scripting variables \$1, \$2, \$3, etc. will be set to the corresponding values. The variable \$0 will be set to the whole line contents. Type "h scripting" for info.

Options:

- **1**: to execute the file in one transaction. Only applicable for COM nodes. To run a MO script in one transaction on CPP nodes, use *trun1*.
- **x**: to stop execution of the file upon failure of a MO WRITE command (create, delete, set, action). The following scripting variables are set automatically when the script stops:
  - *\$errorline* points to the line number where the script stopped
  - *\$errorcmdline* points to the line number of the last command that failed
  - *\$nextcmdline* points to the line number of the next command to execute

Switches:

- **l <lineNumber>**: to start file execution at a specific line number.

#### 4.1.33 trun[is1cr] <moScript>|<http://ipaddress/script>

Run a command file in EMAS/MoTester format.

Execute a command file in EMAS/MoTester format.

By typing **<TAB>**, the unix file system is displayed, making it easier to find the location of the command file.

It is also possible to specify a file located on a web server (eg. when the script is located on the CPP node).

The following commands are supported: ECHO, CREATE, SET, DELETE, ACTION, CHECK, CALL, WAIT.

Lines can be commented out by adding // at the beginning of each line.

See examples below. For more info, refer to MoTester documentation on <http://utran01.au.ao.ericsson.se/moshell/training/references/cpp/runMoTester.html>.

By default, the command file halts when a command fails.

Options:

- **i**: ignore exceptions, the execution does not halt when a command fails.
- **s**: simulated run, the command file execution is simulated, no commands are actually executed on the node. Can be used to verify the syntax of a script prior to running it for real. The simulated mode is always used in "offline mode" or "simulated undo mode", regardless of the "trun" options.
- **1**: executes the whole script in one transaction, then prompts for confirm or rollback. This option should be used with great care and only when absolutely necessary (e.g. when changing IP address of the node, see example script in *moshell/examples/misc/ip\_change.mo*). In regular usage, it is recommended to not use this option as it has been observed to cause database corruptions in certain cases, for instance when creating/deleting certain types of MOs within the same transaction. Database inconsistencies can be checked with the command *dbc*.
- **c**: used in combination with option **1** above, will avoid the prompt by automatically confirming the transaction
- **r**: used in combination with option **1** above, will avoid the prompt by automatically rolling back the transaction

Script example:

```
CREATE
(
  parent "ManagedElement=1,SwManagement=1"
  identity "ROJ1192104_3_R4"
  moType PiuType
  exception none
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
nrOfAttributes 3
  productData Struct
    nrOfElements 5
    productNumber String "ROJ1192104/3"
    productRevision String "R4"
    productName String "TUB"
    productInfo String "TU"
    productionDate String "20030116"
  boardWidth Integer 3
  role Integer 2
)

DELETE
(
  mo "ManagedElement=1,SwManagement=1,PiuType=ROJ1192104_3_R4"
  exception none
)

SET
(
  mo "ManagedElement=1,Equipment=1,Jvm=1"
  exception none
  admClasspath Array Reference 4
    "ManagedElement=1,SwManagement=1,LoadModule=Oms"
    "ManagedElement=1,SwManagement=1,LoadModule=Asms"
    "ManagedElement=1,SwManagement=1,LoadModule=VbjOrb"
    "ManagedElement=1,SwManagement=1,LoadModule=Cma"
)

ACTION
(
  actionName addRepertoire
  mo "ManagedElement=1,SwManagement=1,SwAllocation=TB_LLP"
  exception none
  nrOfParameters 1
    Ref "ManagedElement=1,SwManagement=1,Repertoire=Cello_Common_MP"
  returnValue ignore
)
//wait 2 seconds (time given in milliseconds)
WAIT 2000
//run another MO script
CALL("/home/eric/scripts/newfile.mo")
```

#### 4.1.34 Ctrl-Z; touch /tmp/xxxx; fg (abort MO command)

Abort an MO command or a "for" loop.

To abort an MO command (like `get/st/set/acc...`) or a `for` loop, you need to do two steps:

First type **Ctrl-Z**, to suspend moshell. Then, at the unix prompt, create an empty file `/tmp/xxxx` (where **xxxx** is the *process number* indicated in moshell menu and in the window's title bar) and resume moshell. This is done with the following command:

```
touch /tmp/xxxx ; fg
```

If the moshell prompt doesn't come back even after typing <enter> a number of times, try again suspending (**ctrl-z**) and resuming (`fg`).

#### 4.1.35 pol[b][c][d][h][i][k][m][p][s][r][u][w][y] [-m <mo>] [<interval>] [<waitTime>] [<checkTime>]

Poll the node until the MO service is up or until an operation has completed.

Options applicable for CPP nodes:

- **s**: poll the node until telnet/ssh server is up.
- **h**: poll the node until http server is up.
- **r**: poll the node until http server is down. Eg to find out when the node restart has begun.
- **d**: poll the node until the action `startHealthCheck` is completed.
- **m**: poll the node until /c disk mirroring is completed.
- **k**: poll the node until the system clock is in locked mode.
- **c**: poll the node until the `ConfigurationVersion` has completed its ongoing operations, by monitoring the attribute



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

`currentMainActivity`. Useful to use after action restore/forcedRestore.

- `u`: poll the node until the UpgradePackage has completed its ongoing operations, by monitoring the attribute `progressHeader`. If `progressHeader` is `awaiting_confirm`, the `confirmUpgrade` action will be sent automatically to the node.

Options applicable for COM nodes:

- `h`: poll the node until COM port is up.
- `r`: poll the node until COM port is down. Eg to find out when the node restart has begun.
- `u`: poll the node until the UpgradePackage MO has completed its ongoing operations, by monitoring the attribute `reportProgress`. If the UpgradePackageState is `waiting_for_commit`, the `confirm` action will be sent automatically to the node.
- `b`: poll the node until the Brm\* MO (BrmBackup,BrmBackupManager,BrmBackupScheduler,BrmFailsafeBackup) has completed its ongoing operations.
- `i`: poll the node until the KeyFileManagement has completed its ongoing operations.
- `w`: poll the node until the SwM MO has completed its ongoing operations.
- `y`: poll the node until the CertM/NodeCredential MO has completed its ongoing operations.
- `p`: poll any MO that has a progress attribute, using option `-m <mo>`.

Only one option can be given at a time, ie, it is not supported to combine several options. If no options are given, then it will poll the node until the MO service is up. Note that this polling is done automatically before each MO command. If the loaded CV has changed during the polling then moshell will automatically reload the MOM and MIB (`getmom,parsemom,lt all`).

Arguments (optional):

- `<interval>`: to specify the time in seconds between each polling. Default value is 10 seconds.
- `<waitTime>`: to specify the time in seconds to wait before starting to poll. Default is 20 seconds (60 seconds for `polu`). The reason for this waiting time is because it can take some time before the node starts to execute an operation.
- `<checkTime>`: to specify the time in seconds to wait before checking the result of an action, when using options '`c`' or '`u`'. Default value is 60 seconds.

To abort the polling, do `ctrl-z`, then `touch <stopfile>` (the path to stopfile is printed in the window title), then `fg`. See `h ctrl-z` for more info.

Examples: Performing various operations on a UpgradePackage and polling the node in between each, to find out when it's possible to carry on.

- `acc upgradepackage=CXP9012014_R10CD nonblockinginstall`
- `polu`
- `acc upgradepackage=CXP9012014_R10CD verifyupgrade`
- `polu`
- `acc upgradepackage=CXP9012014_R10CD rebootnodeupgrade`

Note: if `polu` is executed on a node which is being upgraded from the OSSRC SMO application, then the uservariable `polu_confirmupgrade` should be set to 0 to prevent `polu` from confirming the upgrade. Otherwise this would confuse SMO. Refer to the description of this uservariable inside the file `moshell/moshell`.

#### 4.1.36 re[i]

Disconnect and reconnect to the CM service (mobrowser) and/or the PM service (pmtester).

This is useful if the security settings have changed on the node during the moshell session. The "`i`" option is to refetch the iorfile which is necessary if the IORfile has changed on the node (this happens for instance when going from vbjorb to JacORB or changing to corba security). Note that when moshell first starts up, it is neither connected to CM nor PM.

To connect to CM service, just use the `re` command, the `lt` command or any other MO commands (eg `pr`, `get`, etc). As soon as moshell has connected to the CM service the userlabel/site attribute of **ManagedElement** will be read and prompt will be set accordingly.

To connect to PM service, just type the `pst` command which will list all scanners defined on the node.

If there is a node restart with change of CORBA supplier, ie going from Vbjorb (cpp3/4/5) to JacORB (cpp5.1 and above) or vice-versa, then it is necessary to issue the `rei` command which will also refetch the IOR file.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

It is NOT necessary to type `re` after a node restart/upgrade or jvm restart, as long as the corba definitions have stayed the same (corba supplier and corba security setting). Moshell stays connected all the time, though it may not be possible to perform operations while the restart is happening.

#### 4.1.37 getmom [<momversion>]

Check the MOM version currently stored on the node or download a MOM from utran01 server.

When the command is run on its own, without argument, a check will be done to find the MOM version of the node. For CPP nodes, the check is done by reading the header of the MOM file stored under `http://<NodeIpAddress>/cello/oe/xml`. For COM nodes, the check is done by reading the identifier and version attributes in the Schema MOs. Usually this check is done automatically when moshell connects to the MO service.

When the command is run with an argument, then it will try to fetch the corresponding MOM file from the utran01 server and store it in the jarxml folder.

Example:

```
>> getmom RNC_NODE_MODEL_K_9_115_COMPLETE
```

#### 4.1.38 parsemom [<momFile>]

Parse an xml MOM file

Without argument, the `parsemom` command just reparses the current MOM version. Can be used in conjunction with the `getmom` command to check and parse the current MOM.

With argument, the `parsemom` command will parse a different MOM to the one currently loaded in moshell. Can be useful if an incorrect MOM is stored on the node or if just wanting to browse a MOM offline. Example:

```
parsemom moshell/jarxml/RNC_NODE_MODEL_D_3.xml
```

#### 4.1.39 flt/fltc <motype-filter>

Load proxys for an MO type that is not defined in the MOM. ("Force" lt/ltc).

Can be useful in case the xml MOM isn't up to date with the node SW, or in case there is no xml MOM.

Example: `flt rncsystemparameters`

#### 4.1.40 ld <ldn>

Load one MO from the tree and add to the proxy table.

`ld` stands for *Load LDN*. This command loads a proxy for an MO, given its LDN. The LDN doesn't need to contain *ManagedElement=1*, this is assumed.

The MO types are not case sensitive but the MO-ID is! Examples:

```
ld transportnetwork=1,atmport=MS-6-1,vpltp=vpl,vpctp=1,vcltp=36
ld rncfunction=1,iublink=22024
```

The command can be quite slow if it has to search through a large number of MO instances (for instance: `ld rncfunction=1,utrancell=3245`, if there are very many utrancells, will take a while).

#### 4.1.41 fget/lfget <moGroup>|<moFilter>|<proxy(s)>|all [<attribute>]

Read attributes that are not listed in the MOM (f="Force").

With `fget/lfget`, the exact attribute name must be specified in the command.

Any attribute can be displayed as long as it is supported by the node SW.

Example: `fget ^pluginunit= resourceid`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.1.42 eget/leget <moGroup>|<moFilter>|<proxy(s)>|all [<attribute>]

Read attributes that are not listed in the MOM (e="Extended").

With eget/leget, the attribute name is optional, or an attribute filter can be used.

Only attributes listed in the file moshell/commonjars/extendedMOM.txt can be shown. The attributes of this file can also be shown in the commands get/kget/sget if the uservariable `use_extended_mom` is set to 1 (default: 0).

Example: `eget plug res`

#### 4.1.43 sget/lsgget/skget/lsgget/shget/lshget <moGroup>|<moFilter>|<proxy(s)>|all

Read CM/FM attributes from MO(s), one by one ("Slow" get).

Slow but useful in case the standard "get" command is not working due to some attribute returning an exception.

#### 4.1.44 fset/lfset <moGroup>|<moFilter>|<proxy(s)>|all <attribute> [<value>] [<attribute-type>]

Set an attribute that is not described in the MOM ("Force" set).

Can be useful in case the xml MOM isn't up to date with the node SW, or in case there is no xml MOM (e.g. MGW application part).

The syntax is similar to the "set" command except that the attribute type has to explicitly specified using the reference list below.

Example:

```
lfset subrack=ms,slot=20,pluginunit=1$ administrativestate 0 i
```

Following attribute types are supported:

- `i` integer/long/enum
- `l` longlong
- `s` string
- `b` boolean
- `r` moref
- `f` float
- `t` struct
- `ai` array of integer/long/enum
- `al` array of longlong
- `as` array of string
- `ab` array of boolean
- `ar` array of moref
- `af` array of float
- `at` array of structref

#### 4.1.45 facc/lfacc <moGroup>|<moFilter>|<proxy(s)>|all <action> [<param1>] [<param2>]

Perform actions that are not defined in the MOM ("force" action).

Can be useful in case the xml MOM isn't up to date with the node SW, or in case there is no xml MOM.

If the parameter is a an *integer* or a *string*, the parameter type does not need to be specified as in the example below.

Example:

- `lfacc Sector=1,Carrier=1,HsDschResources=1 startRDBTCellHidden 16`
- `facc CommContexts=1 readHsMusOnCCHidden 0`

Otherwise it should be explicitly specified, using the reference list below. Examples:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- lface Equipment=1,Subrack=1,Slot=4,PlugInUnit=1,RaxDeviceGroup=1,UbchDeviceSet  
defineCqiPatternHidden 0 5 15,16,17,18,19:ai

Following parameter types are supported:

- i integer/long/enum
- l longlong
- s string
- b boolean
- r moref
- f float
- t struct
- ai array of integer/long/enum
- al array of longlong
- as array of string
- ab array of boolean
- ar array of moref
- af array of float
- at array of structref

#### 4.1.46 fdel/lfdel <moGroup>|<moFilter>|<proxy(s)>

Delete MO(s), including systemCreated MOs.

Works in the same way as the regular del/lfdel command except that it also tries deleting the systemCreated MOs, whereas the del command skips them.

Though the systemCreated MOs cannot be deleted, this command can be useful when generating set commands in "simulated undo mode" for those systemCreated MOs.

## 4.2 Other MO commands

### 4.2.1 cvls/cvmk/cvms/cvset/cvrm/cvrbrm/cvcu/cvget[f]/cvput/cvls1

CV backup handling: list, make local, make remote, remove, setstartable.

A set of commands similar to the "cv" commands in OSE but operate through MO interface instead of telnet/ssh.

Command syntax and description (CPP):

- cvcu : display the current cv information only (equivalent of "cv cu").
- cvls [<cv-filter>] : display both the current cv information (equivalent of cv cu) and cv list (equivalent of cv ls). It is possible to filter the output of cvls to only show CVs where the CV name or CV attributes match a certain string. The proxy Id of the CVs can be used in the commands cvrm and cvget. The display of CV proxy Id can be disabled with command "safe+".
- cvls1 : similar to the cvls command except that it executes via the OSE shell instead of the MO service.
- cvmk <cvname> [<operator>] [<comment>] : create a local cv backup. Operator name and comments (not longer than 40 characters) can be given as argument.
- cvset <cvname>|<cv-Id> : set a cv as startable.
- cvms <cvname> [<operator>] [<comment>] : create a cv and make it startable (combination of cvmk and cvset)
- cvget[f] <cvname>|<cv-filter>|<cv-Id> [<destdir>] : make a remote backup of a cv to the workstation. The operation is done with the MO action putToFtpServer unless option "f" has been specified, in which case the transfer will be done by FTP/SFTP. The second argument is optional. If not given, a default folder is chosen for the backup  
~/moshell\_logfiles/logs\_moshell/cv/<node>/<date>\_<time>/
- cvput <zipped-cvfile> : transfer a remote CV backup (zip file) from the workstation to



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

the node. The operation is done with the MO action getFromFtpServer.

- `cvmr <cvname>|<cv-filter>|<cv-Id>` : remove one or more cv's. If the argument does not match an existing CV then all CVs matching that string will be removed. A confirmation message is printed before removal. The CV(s) will automatically be removed from rollback list when necessary.
- `cvrbrm` : remove one or more cv's from the rollback list. If the argument does not match an existing CV then all CVs matching that string will be removed. A confirmation message is printed before removal.

Command syntax and description (COM):

- `cvcu` : display the current backup information only.
- `cvls` : same as above plus the list of SwVersions, UpgradePackages and BrmBackups. The Id field of the BrmBackup can be used in the commands cvrm and cvget.
- `cvmk <cvname>` : create a local backup.
- `cvmr <cvname>|<cv-filter>|<cv-proxyId>` : remove one or more backups from the node. If the argument does not match an existing backup then all backups matching that string will be removed. A confirmation message is printed before removal.
- `cvget <cvname>|<cv-filter>|<cv-Id> [<destdir>]` : export a backup to the workstation. The second argument is optional. If not given, a default folder is chosen for the backup  
~/moshell\_logfiles/logs\_moshell/cv/<node>/<date>\_<time>/
- `cvput <zipped-cvfile>` : transfer a backup (zip file) from the workstation to the node.

Examples:

- `cvls`: List all CVs
- `cvls CXP9011274_R9A`: List all CVs using Upgradepackage CXP9011274\_R9A
- `cvms RNC11_Final`: Create a cv and make it startable (no userid or comments given)
- `cvms RNC11_Final eanzmagn cell power increased to 33dBm`: Create a cv and make it startable (userid and comments given)
- `cvmr Temp`: Remove all cv's whose name match the string "Temp"
- `cvmr !Final`: Remove all cv's whose name don't match the string "Final"
- `cvmr !Final|RNC`: Remove all cv's whose name don't match the string "final" or the string "RNC"
- `cvget RNC11_Final`: Make a remote backup of a CV to the workstation where moshell is running
- `cvget RNC11.*Fi`: Make a remote backup of all CVs whose name matches RNC11.\*Fi
- `cvput /home/eric/RNC11_Final.zip`: Transfer a remote cv backup from the workstation to the node
- `cvmr 1-15`: Remove the oldest 15 CVs.
- `cvget 3,5,8`: Fetch CV number 3, 5, and 8 from the cv list.
- `cvset 23` : Set CV number 23 as startable

#### 4.2.2 inv[hlxbpcr] [<Filter>] [<stateFilter>]

Complete HW/SW inventory. Includes information about RPUs, licensing, JVM, devices, XPs, ISL, etc.

This command performs a complete HW/SW inventory via the MO and COLI interface. All SW including JVM, RPU, and Device SW (spm,dsp,fpga) is shown. Licensing (features and capacity) as well as overview of the ISL links is also shown.

Options:

- `h`: fetch HW information only, only via MO commands.
- `x`: fetch HW information only, using MO commands and COLI commands. The `x` option is similar to the `h` option but shows the XP link handler addresses associated to each AuxPlugInUnit MO. Applicable for RBS and ENB only. This information is also shown when no options are specified.
- `l`: show licensing information only (feature and capacity licenses)
- `p`: show CPU load of the PlugInUnits
- `r`: re-read the inventory data from the node.
- `c`: print the HW and CPRI link tables in CSV format

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **b**: print the Received BER value in the RBS SFPs

The first time the command is run, it takes a bit longer because the data has to be fetched from the node before parsing. The following times the command is run, the existing data is parsed again, unless the **r** switch is used ("refresh"), in which case, the data is fetched again and parsed. When no options are specified, all the information will be displayed except the CPU load.

Arguments:

- The first argument (general filter) allows to only show the lines matching the filter string. Negative filter is supported by putting a exclamation mark in front of the filter. See examples further down.
- The second argument (state filter) allows to only show the lines where the MO status matches the state filter.

Printout description:

- the first table shows the MP/BP hardware info, position, and status. The CPU column shows the CPU load (read from pget plug load), the GE column (when subrack contains CMXB) shows the connection status to the 10G IP backplane (attribute EthernetSwitchModule::backPlanePortState), the CE column (when subrack contains SCXB) shows the connection status to the 1G IP backplane (attribute ControlSwitch::backPlanePortState). The column **c/p** shows the disk usage on /c or /p (depending if it is a central MP or regular MP), and the column **/d** shows the disk usage of the /d volume.
- the second table shows the XP/EP hardware/software info and status
- the third table shows the subracks and backplanes HW info. The switchState field consists of three digits: the state of the switch, the state of plane A, and the state of plane B.
- the next table(s) show either ISL links or CPRI links and/or RDS/DOT inventory.
  - the ISL links are only applicable in nodes containing several subracks, eg RNC/MGW. All ISL links connecting the main subrack to each extension subrack are shown. The status of each link is show: 1st digit corresponds to "operationalStateSync" and the 2nd digit corresponds "operationalStateTraffic". The ports connected to each ISL are shown, including the type of board (SCB/SXB) and the port state (active/passive).
  - the RDS/DOT inventory table is only applicable for RBS and ENB. The description of the fields can be found further down in this help.
  - the CPRI links are only applicable for RBS or ENB, these are the digital links between DU-XMU-RU/RRU, can be either electrical (T=E) or optical (T=0). If the node contains optical CPRI links, there will be two additional tables showing the product information and diagnostics for the SFP's located at the extremities of each CPRI link (SFP= Small Form-factor Pluggable transceiver). See description in note further down.
- the next two tables shows the list of features and capacity and their licensing status  
The FAJ numbers are read primarily from the License.xml file on the node and if not found there, they are read from the excel sheets on [https://ericoll2.internal.ericsson.com/sites/SW\\_License\\_Handling\\_Community/SWLicenseHandling/ELIS/Wiki/License](https://ericoll2.internal.ericsson.com/sites/SW_License_Handling_Community/SWLicenseHandling/ELIS/Wiki/License)  
The information about restricted features is read from the document 2/22104-FGB101135.
- the next table shows the java loadmodules that are running in the JVM.
- the next table shows the configuration and status of the ReliableProgramUniters (RPU).
- the last table shows the Programs running in each processor including devices.

Description of the "state" column:

State information is always abbreviated to one digit, in the same way as for other printouts such as stt, stv, str, etc.

- 1 = unlocked enabled
- 0 = unlocked disabled
- L = locked (opstate could be enabled or disabled)

Description of the LED columns:

- FAULT: RED
- OPER: GREEN
- MAINT: YELLOW or BLUE (old boards use YELLOW, new boards use BLUE)
- STAT: YELLOW (only applicable for EvoC)

Description of the fields in the RDS/DOT table (only applicable for RBS/ENB containing IRUs with DOTs):

- RDS: the product name of the DOT unit
- LNH/ID: the linkhandler address and port number (1 to 8) of the IRU to which the DOT is connected
- P: the PowerOverEthernet status. Read from the command `eqpm rdPoe status all` (1=detected:true,

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

0=detected:false)

- C: the Connection status. Read from the command `rdc dump all` (1=RD\_OPERATIONAL, 0=RD\_FAILED/NOT\_PRESENT)
- T: the LTU status. Read from the command `rdsh all hal ltu lock` (1=LOCKED, 0=NOT\_LOCKED)
- TX/RX: the status of the DOT's TX/RX branch A and B respectively. Read from the command `rdc dump all` (1=ENABLE, 0=DISABLE)
- TEMP: the temperature of the DOT, in Celsius. Read from the command `rdsh all temp`.
- VII: the status of the visual indicator of the DOT. Read from the command `rdsh all vind`.
- FAULT: indicates the presence of alarms on the DOT. Read from the command `eqpm rdFault dump`

Description of the fields in the SFP diagnostics table (only applicable for RBS/ENB containing optical CPRI links):

- ID: the unique id of the CPRI link which can be used to match against the connection and product information in the above two tables of the printout.
- LINK: indicates if the CPRI link is Up or Down according to the information from the COLI command "ricr".
- WL: the wavelength of the laser, in nm.
- TEMP: temperature of the SFP in Celsius
- TXbs: the TX bias level in percent of the low/high warning thresholds. Calculated with the formula  $100 * (\text{Bias} - \text{LowLimit}) / (\text{HighLimit} - \text{LowLimit})$ . Should not be negative or higher than 100%. Acceptable values are typically in the range of 1% to 50%, depending on SFP vendor.
- TXdBm: the transmitted power in dBm
- RXdBm: the received power in dBm
- BER: the counter value of the received Bit Error Rate
- DILoss/UILoss: the difference in dB between the TX power of the sending port and the RX power of the remote port. Acceptable values should be between -1 dB and +3dB. If the CPRI link is down or the SFP voltage is not between 2 and 4V, then some of the diagnostics values will show "NA" ("Not Applicable"). Diagnostics values for Non-Ericsson SFP's will show "NE" ("Non Ericsson")

Examples:

- `inv CXC132055` → only rows matching CXC132055 will appear. This is convenient to lookup the name of an LM and in see which boards it is running.
- `inv :` → only RPU information will be printed.
- `inv nss` → to see which loadmodules contain the string "nss" and in which boards they are running.
- `inv . L|0` → only rows where the state of the MO is locked or disabled will appear.
- `inv roj L|0` → only rows matching "ROJ" and where the state is locked or disabled will appear.
- `inv !program` → only rows NOT matching the word "program" are displayed.

#### 4.2.3 cab[adefghlmrstxc] [ | <unix-cmds> ]

Display of miscellaneous COLI printouts relating to hw, sw, restarts, leds, cpu load, errors, disk/ram usage

The `cab` command offers a number of options, it is possible to combine several options, eg: `cabslxrdg`, `cablx`, `cabxs`, etc.

The command `cabslxrdgm` will give the maximum amount of information.

Options:

- `h` : prints MP/BP HW info and led status, MP temperature, and coreMgr status. If no options are given then this is the default option.
- `t` : same as "h" but without the temperature, nor the TX/VSWR values (in RBS/ENB)
- `x` : same as "h" plus led and hw info for the XP boards (eg: TMA, MCPA, Fans, RU, RRU, etc.). With option "c" ("cabxc") the output is in CSV format.
- `s` : same as "h" plus list of programs running in all MP/BP
- `r` : prints all MP/BP restarts grouped by board. To see this info in chronological order, use the command "lgg". Abnormal restarts are highlighted in red.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **a** : prints only abnormal MP/BP restarts.
- **d** : print disk usage. Disks that are getting over a certain limit will appear in color. The limit can be defined in cabview file.
- **f** : print disk and flash usage.
- **g** : print MP/BP HW errors (e.g. faulty disk, faulty RAM, etc).
- **m** : print MP/BP RAM memory usage.
- **e** : print MP/BP added T&E trace conditions. Superceded by `fte s` command.
- **l** : MP/BP/SP processor load. Superceded by `proclload` command.

The following OSE commands are run and parsed by the various "cab" functions:

- **h** : `pboot sh par, vii, mirror s, ppctemp, boardtemp`
- **t** : `pboot sh par, vii, mirror s`
- **x** : `pboot sh par, vii, mirror s, ps port*, par get SYS_HW*, listObj subrack, getAttrObj subrack, warpA/warpB read, warp3 txpwr, fui get temp, fui get vswr`
- **s** : `pboot sh par, vii, mirror s, listloaded`
- **l** : `pboot sh par, capi prio, capi core all`
- **r** : `llog -l`
- **a** : `llog -l` Only restarts with error code not matching 0xB0AD or 1010[9-F] or containing a PMD are printed
- **d/f** : `vols, ls /d/loadmodules`
- **g** : `pboot sh par, dumpelg`
- **m** : `pboot sh par, mmu, mm -p`
- **e** : `pboot sh par, te s, te s -restart`

Note 1: Regarding the CoreManager status: If a board has got a CoreManager status, it means that the board is running the Core Manager programs (EqmMgr, Database, LoaderServer). If the node is configured with Fault Tolerant Core, there are two boards running the Core Manager functionality. One board is *Active* and the other one is *Standby*.

When the node is configured with Fault Tolerant Core, the **/c** drive is mirrored between the two Core Manager boards.

If the status of the Standby board is *StandbyReady*, then it means that the **/c** drive is correctly mirrored and the standby board can take over the active role at any time, in case the active board fails or restarts.

If the status of the Standby board is *StandbyWriting*, then it means that the **/c** drive is performing a small update and the standby board can take over in a short while, as soon as the disks are updated.

If the status of the Standby board is *StandbySync*, then it means that the **/c** drive is performing a complete update and the standby board will not be able to take over until this is completed. The progress is shown as a percentage value (eg: **StandbySync-56%**).

Note 2: When many commands are to be sent, the `cab` function will put them into a command file, transfer that file (via (s)ftp) to the node and run that file from within the node, using the `shell -f` command.

This will save time instead of having to send each command one by one to the node.

There is a user variable called `fast_cab_threshold` which determines the number of boards in the node above which a command file will be transferred to the node.

See Section 2.5 and the **moshell** file for more info about user variables.

Note 3: Regarding PMD Ids appearing in `cabr/caba`:

A Post-Mortem Dump (PMD) may be associated with an abnormal board restart. In this case, the PMD Id is shown in `cabr/caba` commands. It is possible to show and collect the PMD files with the command `lgp`. Alternatively the commands `dump list -a, ftreef /c/pmd`, or `lg1` can also be used to show the PMD files.

Note4: Regarding TX power calculation in "cabx":

For RU PL4:

```
TXPwrA=(DL_PM_PA0_C0+DL_PM_PA0_C1+DL_PM_PA0_C2+DL_PM_PA0_C3)*16384*powerClassA/(1228800*8491396)
TXPwrB=(DL_PM_PA0_C0+DL_PM_PA0_C1+DL_PM_PA0_C2+DL_PM_PA0_C3)*16384*powerClassB/(1228800*8491396)
```

Where:

- **DL\_PM\_PA0\_Cx** are read from RU COLI commands "warpA read" for TXA och "warpB read" for TXB
- **powerClass** is read from RU COLI command "db list \*currentPowerClass", or "txm rh all wrk dump"

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

For RU PL5:

- $TXPwrA = pwrClassA * 268435456 / 8491396 * (10^{(B0/10)} + 10^{(B1/10)} + \dots + 10^{(B7/10)})$
- $TXPwrB = pwrClassB * 268435456 / 8491396 * (10^{(A0/10)} + 10^{(A1/10)} + \dots + 10^{(A7/10)})$

Where:

- A0 to A7 and B0 to B7 are read from RU COLI command "warp3 txpwr" or "warp:0/warp:1 txpwr"
- powerClass is read from RU COLI command "db list \*currentPowerClass", or "txm rh all wrk dump"

Note5: Regarding RSSI measurement in "cabx" for WRBS:

The RSSI is value is read from the MP trace `bus_receive` on `CDCI_TR`. A asterisk in front of a TrDevice in the printout indicates that this TrDevice has reported a measurement. TrDevices without asterisk could be due to that this is a TX or that the cell is disabled.

Note6: Refreshing of the cache.

The cab command reads most of its data from the node each time it is executed. However there is some static data such as board list and cell list which is read only once and then kept in a cache. If this information has changed during the session, it is possible to refresh the cache by running the "bor" command.

#### 4.2.4 stc[p][r] [<Filter>] [<stateFilter>]

Display state and configuration of Atm/Tdm CrossConnections.

The filter allows to only show the rows matching the filter string. The stateFilter allows to only shows the MOs matching the state filter.

The first time the command is run, it takes a bit longer because the data has to be fetched from the node before parsing. The following times the command is run, the existing data is parsed again, unless the `r` switch is used (*refresh*), in which case, the data is fetched again and parsed.

The first field is the MO id of the Atm/Tdm CrossConnection.

The second field is the state information, consists of five digits:

- 1st digit: *operational/State* of the Atm/Tdm CrossConnection MO (0=disabled, 1=enabled)
- 2nd and 3rd digits: *operational/State* of the VclTp MOs (A and B side).
- 4th and 5th digits: only applicable if "p" option was used. Shows the status of the **pget** on VclTp MOs (A and B side). Set to 0 if the transmittedCells counter has incremented but the receivedCells counter has not incremented within a given time period (by default 25 seconds, can be changed in the uservariable `pm_wait`): this indicates that there is no response from the remote end.

The third and fourth field are the Mold for VclTpA and VclTpB, abbreviated in the following way: AtmPortId/VpiTpId/VclTpId

The fifth and sixth fields are the actual Vpi/Vci value for VclTpA and VclTpB. Can be useful in case the Mold of the VclTp does not match the Vpi/Vci value.

The seventh and eighth fields are the Mold for the traffic descriptor of VclTpA and VclTpB.

The last field is the *userLabel* of the AtmCrossConnection MO.

Note: in the case of TdmCrossConnections, the third and fourth fields show the Timeslot in the Ds0Bundles A and B. The fifth and sixth fields show the Mold of the Ds0Bundle A and B.

Examples:

1. `stc 2051` - show all crossconnections where the information matches "2051" (in the case below, where the atmport is 2051)
2. `stc .* 0` - show all crossconnections that are not working properly.

Printout format, AtmCrossConnection:

```
=====
CCId          CSSPP VclTpA    VclTpB          Vp/VcA  Vp/VcB  TD-A          TD-B          UserLabel
=====
MGTS44500_MSC6_AAL2a  11110 2041.*1.*501 2071.*1.*501  1/501   1/501  C2P12000      C2P12000      MGTS44500_MSC6_AAL2a
MGTS44500_MSC6_AAL2b  11111 2041.*1.*502 2071.*1.*502  1/502   1/502  C2P12000      C2P12000      MGTS44500_MSC6_AAL2b
SOLVER44800_MSC6_AAL2a 10101 2051.*1.*136 2071.*3.*512  1/136   3/512  C2P12000      C2P12000      SOLVER44800_MSC6_AAL2a
MGTS45600_MSC9_AAL2a  11100 2052.*1.*300 2041.*1.*300  1/300   1/300  U2P3520M3520 U2P3520M3520 MGTS45600_MSC9_AAL2a
=====
```

Printout format, TdmCrossConnection:

```
=====
CCId          CSS TSA TSB Ds0A          Ds0B          UserLabel
=====
1192_1191_ts16 101 1 1 1,Slot=27.*E1.*=1277,Ds0.*=127702 1,Slot=27.*E1.*=1276,Ds0.*=127602 TS 16 127702_port1191
=====
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.2.5 std[ar] [<filter>]

Display state and configuration of devices (RNC and MGW only).

Argument (optional):

Only lines matching the filter will be displayed. If no argument, all lines are displayed.

Example:

- `std fax` - show fax devices (mgw)
- `std pdr` - show pdr devices (rnc)
- `std 0020` - show devices on board 0020

Options:

- `r`: to refresh the printout. In MGW the device data is locally cached and updated when the "r" option is given. This is the same behaviour as in commands such as `bo`, `stc`, `stt`, `stv`, `inv`, etc. In RNC, the "r" option forces moshell to re-check the device to module relationship. All other data is refreshed each time, even without the "r" option.
- `a`: to fetch some additional device usage information.

#### std on MGW

The first table (only printed with option `a`) shows the DSP SW and usage for each MSB board. The `GMD` field indicates the status of the GRA-GPB ("G"), the MSB ("M") and DSPs ("D"). The GRA-GPB and MSB status correspond to the state of the corresponding `PlugInUnit`: L=locked, 1=enabled, 0=disabled. The DSP status is found from the command `pingdsp` on MSB3 and `mmpp pingdp` on MSB4. The DSP SW is found from the command `rev` on MSB3 and `mmpp dspc devt` on MSB4. The `DeviceType` and all remaining fields are found from the command `gradsl` on GRA-GPB:

- `Set = DevSetNr` : device set id
- `ResId = resourceId`
- `nRes = nrOfResources` : total nr of allocated resources for this device set
- `nIdle = nrOfIdle` : nr resources not in use
- `graCap = graRdCapacity` : available capacity expressed in PUs available for normal calls
- `dspCap = dspRdCapacity` : rdScaledCapacity (reported by DSP and used by MFD only) is available devices expressed as remaining PUs
- `totCap = rdCapacityTot` : total capacity expressed in PUs (reported by RD in `attachCfm`)
- `rej = nrTimeoutRej` : number of rejected requests because of 30+30ms + 1 sec DSP supervision timer timeout. At this point DSP is marked as failed
- `nRest = nrGraOrderedDspRestarts` : number of GRA ordered DSP restarts because of 30+30ms + 1+10 sec DSP supervision timer timeout
- `dupCep = nrOfDupCeps` : current number of duplicated CEPs in this RD

More info found in `gradsl` printout description in M-MGW Traffic Control Troubleshooting Guideline 25/1553-AXM 101 01/7

| Sr | Slot | Lnh    | Board | GRA  | DSP | GMD | DeviceType  | SW                 | Set | ResId | nRes | nIdle | graCap | dspCap | totCap | rej | nRest | dupCep |
|----|------|--------|-------|------|-----|-----|-------------|--------------------|-----|-------|------|-------|--------|--------|--------|-----|-------|--------|
| 3  | 7    | 730700 | MSB3  | 7304 | 1   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 2   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 3   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 4   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 5   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 6   | 111 | MFC         | CXC1327801/8_R3D01 | 326 | 11088 | 24   | 24    | 55542  | 55542  | 55542  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7303 | 7   | 111 | IM          | CXC1327799/8_R3E01 | 327 |       | 100  | 100   |        |        |        |     |       |        |
| 3  | 7    | 730700 | MSB3  | 7304 | 8   | 111 | CSD_GSM_MFH | CXC1327794/8_R3D01 | 328 | 11112 | 36   | 36    | 55650  | 55650  | 55650  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 9   | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 10  | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 11  | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 12  | 111 | UMTS_MFD    | CXC1327790/8_R3L01 | 325 | 9594  | 166  | 166   | 57300  | 57300  | 57300  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 13  | 111 | CSD_DIGITAL | CXC1327791/8_R3D01 | 329 | 11148 | 16   | 16    | 55650  | 55650  | 55650  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 14  | 111 | CSD_DIGITAL | CXC1327791/8_R3D01 | 329 | 11148 | 16   | 16    | 55650  | 55650  | 55650  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 15  | 111 | CSD_MODEM   | CXC1327792/8_R3D01 | 330 | 11180 | 16   | 16    | 55650  | 55650  | 55650  | 0   | 0     | 0      |
| 3  | 7    | 730700 | MSB3  | 7304 | 16  | 111 | CSD_FAX     | CXC1720519/8_R3D02 | 331 | 11196 | 5    | 5     | 55650  | 55650  | 55650  | 0   | 0     | 0      |

The second table shows the device status and availability for each MSB board.

The `MD` field indicates the status of the MSB ("M") and DSPs ("D").

The MSB status corresponds to the state of the corresponding `PlugInUnit`: L=locked, 1=enabled, 0=disabled.

The DSP status is found from the command `pingdsp` on MSB3 and `mmpp pingdp` on MSB4. If all DSPs are ALIVE, the state is 1, otherwise it is 0.

The remaining fields are read from the action `getBoardDetails` on `MsDeviceGroup`:

- `nDev = nrOfRds` : The number of Root Devices (RD) on the board, configured with the same devices (set of services)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

as pointed out by the deviceType attribute.

- %Lock = capacityDependencyLockedDev : The fraction (
- %Dis = capacityDisabledDev : The fraction (
- maxDev = maxNrOfDev : A theoretical maximum number of configured devices on the board.

More info in MOM "mom dev getboarddetails".

| Sr | Slot | Lnh    | Board | MD | SwAllocation        | DeviceType | nDev | %Lock | %Dis | maxDev |
|----|------|--------|-------|----|---------------------|------------|------|-------|------|--------|
| 3  | 7    | 730700 | MSB3  | 11 | MGW_MSB3_Profile_16 | CSDDigital | 2    | 0     | 0    | 32     |
| 3  | 7    | 730700 | MSB3  | 11 | MGW_MSB3_Profile_16 | CSDGSMFH   | 1    | 0     | 0    | 36     |

The third table shows the device status and availability for each device pool.

In MGW R2/R3, the usage is given as a number, in MGW R4, it is given as a percentage.

| DevPool        | Total | %Idle | %Busy | %Failed | %DepLock | %DepFail |
|----------------|-------|-------|-------|---------|----------|----------|
| CSDDigitalPool | 256   | 100   | 0     | 0       | 0        | 0        |
| CSDGSMFHPool   | 288   | 100   | 0     | 0       | 0        | 0        |
| CSDModemPool   | 128   | 100   | 0     | 0       | 0        | 0        |

### std on RNC

Shows the configuration, status and usage of RNC devices, CC, DC, PDR.

State information is abbreviated in the following way:

- 1 = unlocked enabled
- 0 = unlocked disabled
- L = locked (enabled or disabled)
- I = idle (the device is not handling calls)
- A = active (the device is handling calls)
- B = busy (the device is fully used and cannot handle more calls)

Options:

- r: re-read the device configuration information. Without r, only device state and usage is updated.
- a: show additional device information relating to the device usage. Note: the use of this option means the moshell will run the command `drh_dcrh_topdata all` which may cause module restart on certain RNC SW releases, see TR WRNae26272. If the RNC is running a SW release containing the correction of TR WRNae26272, then it is safe to run `stda`. Otherwise, just run `std` without the `a` option.  
The correction for this TR is included in RNC SW P6.1.4 (CXP9012842\_R3BD) and P7.0.1 (CXP9012995\_R6CF).

### Printout description for device tables:

Note: some of the fields are only shown when running option a (`stda`).

#### Common fields:

- The MOD and GPB column show the module MP controlling the device.
- The SPM and DEV columns show the MO id of the Spm and Device MOs.
- The G column shows the state of the module MP (GPB).
- The D column shows the state of the Device MO.
- The S column shows the state of the Spm MO.
- The U column shows the usageState of the Device MO.

#### CC device specific fields:

| TYP | MOD | GPB    | SPB   | SPM      | DEV      | GDS | U | LNH            | CPU | UEs | Cells (DRH/CCS/max) |
|-----|-----|--------|-------|----------|----------|-----|---|----------------|-----|-----|---------------------|
| CC  | 4   | ES1-15 | SPB21 | ES1-10-1 | ES1-10-1 | 111 | A | 011000/sp0.lnh | 18% | 5   | 52 52 96            |

- CPU shows the CPU load of the Device and is read from the variable `spCpuLoad` in the `drh_ccrh_topdata` printout in module MP.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- UEs shows the number of UEs handled by the device and is read from the `uelist` printout in CC SP.
- Cells/DRH shows the number of Cells handled by the device and is read from the `drh_ccrh_topdata` printout in module MP.
- Cells/CCS also shows the number of Cells handled by the device but it is read from the `celllist` command in CC SP.
- Cells/max shows the maximum number of Cells supported by the device (shown on RNC >= P7) The two values Cells/DRH and Cells/CCS should always be the same, otherwise it indicates a discrepancy between DRH and CCS.

#### DC device specific fields:

| TYP | MOD | GPB   | SPB   | SPM     | DEV | GDS | U | LNH            | CPU | Res | HsUEs | UEsDcs | UEsDch | UEsDrh | gbrResourcePoints |
|-----|-----|-------|-------|---------|-----|-----|---|----------------|-----|-----|-------|--------|--------|--------|-------------------|
| DC  | 1   | MS-14 | SPB21 | MS-19-5 | 3   | 111 | A | 001900/sp4.lnh | 26% | 10% | 14    | 22     | 15/150 | 16/150 | 530/5100          |
| DC  | 1   | MS-14 | SPB21 | MS-20-3 | 4   | 111 | A | 002000/sp2.lnh | 34% | 11% | 23    | 22     | 21/150 | 23/150 | 540/5100          |

- CPU is read from the variable `cpuLoad` in the `drh_dcrh_topdata` printout in module MP.
- Res is the percentage of resourcePoints used out of maxResourcePoints (read from `drh_dcrh_topdata all`)
- UEsDcs is read from the `uelist` printout in DC SP.
- the following columns are read from the `drh_dcrh_topdata all` printout in module MP: HsUEs->noOfHsCapableUes, UEsDch->noOfUesOnDch/maxNoOfUesOnDch, UEsDrh->noOfUesOnDch/maxNoOfUesOnDch, gbrResourcePoints->gbrResourcePoints/maxResourcePoints.
- UEs/DCS is read from the `uelist` printout in DC SP.

#### PDR device specific fields:

| TYP | MOD | GPB   | SPB   | SPM     | DEV     | GDS | U | LNH            | CPU  | UEs | RABs | aal5 | usedCapacity |
|-----|-----|-------|-------|---------|---------|-----|---|----------------|------|-----|------|------|--------------|
| PDR | 1   | MS-14 | SPB21 | MS-19-1 | MS-19-1 | 111 | A | 001900/sp0.lnh | 3.7% | 55  | 406  | 95   | 3796/215000  |

- CPU is read from the command `spp -p xxxx00/spx.lnh sp procload 1` on central MP or `capi prio` and `capi core 0` on PDR devices (depending on RNC SW release)
- UEs is read from the `uelist` printout in PDR SP.
- RABs and aal5 correspond to the variables `noOfRabs` and `noOfAal5Conns` in the `drh_pdrh` printout in module MP.
- usedCapacity is read from `usedCapacity/maxCapacity` in `drh_pdrh` in module MP.

#### Module summary table:

These table show the device usage on module basis.

The fields are the same as in the tables above except for the DC summary table which contains some additional fields, read from the printout `lh mod drh_trbr_data`:

- ATM: noOfAtmTrBr
- IP: noOfAtmTolpTrBr
- BEE: noOfBeesTrBr

Also in the CC summary table, the field "max" is replaced by the field "GPB" which indicates the number of cells that are handled by the GPB of that RncModule. The field "GPB" is read via the attributes `lubLink::rncModuleRef` and `lubLink::reservedBy`

#### SPB summary table:

In this table we get an overview of all the SPB boards, their device states, usage, and module allocation.

The State column shows first the state of the SPB PlugInUnit, then the state and usage of its devices.

The Module column shows which module is handling each device.

| Sr | Slot | Ln timer | Board | SwAlloc    | Type  | State   | Usage | Module     |
|----|------|----------|-------|------------|-------|---------|-------|------------|
| MS | 19   | 001900   | SPB21 | SPB_TYPE_A | PCDDD | 1-11111 | AAAAA | 1 1 13 8 8 |

#### 4.2.6 stv[b][r] [<Filter>] [<stateFilter>]

Display state, user, and bandwidth usage for ATM ports and channels.

Options:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **b**: to get the bandwidth usage for AtmPorts. Requires running some COLI commands.
- **r**: to refresh the data (ie. re-read from node).

Arguments:

- the first argument matches on the whole line
- the second argument matches only the state field ("VU")

Examples:

- `stvb ms-6-1` print ATM data for AtmPort=MS-6-1 and all VPs/VCs underneath it
- `stv p4500` print ATM data for all VPs/VCs using a traffic descriptor with peak cell rate 4500
- `stv . 0|L` print all Ports/VPs/VCs with state disabled or locked

Printout format:

- The field `ResrvBw/TraffDesc` shows:
  - the `AtmTrafficDescriptor` for `VclTp` MOs. For VC's used by `Aal2Path`, the `Aal2 QoS` is shown in brackets.
  - the reserved bandwidth (in cells/s) for `AtmPort` and `VplTp` MOs. E.g. 353000/353207 means 353000 cells/s reserved out of 353207 available. The available bandwidth for `AtmPort` is obtained from the COLI command `aet_atmmp etatmportfro <fro>` and is only shown when option **b** is specified. The reserved bandwidth for `VplTp` is obtained from the traffic descriptor of `VplTp` (total bandwidth) and the sum of the traffic descriptors of `VclTp` (used bandwidth). The cellrate used by a traffic descriptor is the `PeakCellRate` in case of CBR and the `MinimumCellRate` for UBR/UBR+.
- The field `User` shows the MO using the `VclTp`. For `Aal2PathVccTp` MOs, the information in brackets shows the `aal2PathId` and reserving `Aal2Ap`. For `PacketDataRouter` MOs, the information in brackets shows the `PdrDevice` position, the `Aal5TpVccTp MOid`, the `rnclpAddress`, and the `cnluLinklpAddress`. For `Mtp3bSI` MOs, the information in brackets shows the `SignallingLinkCode (SLC)`, the `linkState`, the `proceduralState`, and the `usageState`, same as in `stt` printout. Refer to `stt` help for more info about the `Mtp3bSI` information.
- The field `M` shows the `RncModule` handing the `User`.
- The field `VU` shows the state of the VP/VC followed by the state of the `User`. L=locked, 0=disabled, 1=enabled. E.g. VU=10 means that the `VclTp` is enabled but the `User` MO is disabled. For `Aal2PathVccTp` MOs, there is an extra digit which represents the `remoteBlockingState` (0=remotely\_blocked, 1=remotely\_unblocked, ?=undefined). Note that the `remoteBlockingState` is not shown when the `Aal2PathVccTp` is locked. For `PacketDataRouter` MOs, there is an extra digit which represents the state of the `PdrDevice` MO.

Example printout:

```
=====
VclTp                               VPI/VCI  ResrvBw/TraffDesc  VU  M  User
=====
AtmPort=MS-6-1                      /                1
AtmPort=MS-6-1,VplTp=1              2  224000/353000    1
AtmPort=MS-6-1,VplTp=1,.*VclTp=vc34 2/34  U3P4500M3000    11  Mtp3bSpItu=Iu1,Mtp3bSls=Iuc-1-2300-3,Mtp3bS1Itu=1
AtmPort=MS-6-1,VplTp=1,.*VclTp=vc50 2/50  U3P4500M3000    11  Mtp3bSpItu=Iu1,Mtp3bSls=Iup-2-2810-3,Mtp3bS1Itu=1
AtmPort=MS-6-1,VplTp=1,.*VclTp=vc90 2/90  C2P12000 (AB)    111 1  Aal2PathVccTp=Iu1-1-1 (1, Aal2Ap=Iu1)
AtmPort=MS-6-1,VplTp=1,.*VclTp=vc91 2/91  C2P12000 (AB)    111 1  Aal2PathVccTp=Iu1-1-2 (2, Aal2Ap=Iu1)
...<cut>...
AtmPort=MS-6-1,VplTp=1,.*VclTp=Pdr1Gtpu1 2/230 U3P66600M5000    11  RncModule=1,PacketDataRouter=Pdr1Gtpu1
AtmPort=MS-6-1,VplTp=1,.*VclTp=Pdr1Gtpu2 2/231 U3P66600M5000    11  RncModule=1,PacketDataRouter=Pdr1Gtpu2
AtmPort=MS-26-1                      /                1
AtmPort=MS-26-1,VplTp=1              1  13804/14650      1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc34 1/34  C1P5             11 1  IubLink=1,NodeSynchTp=1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc35 1/35  C1P5             11 1  IubLink=1,NodeSynchTp=2
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc36 1/36  U3P1000M80       11 1  IubLink=1,NbapCommon=1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc37 1/37  U3P1000M80       11 1  IubLink=1,NbapDedicated=1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc38 1/38  U3P1000M80       11 1  Aal2Sp=1,Aal2Ap=Iub1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc39 1/39  C2P6657 (AB)     111 1  Aal2PathVccTp=Iub1-1 (101, Aal2Ap=Iub1)
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc40 1/40  C2P6657 (AB)     110 1  Aal2PathVccTp=Iub1-2 (102, Aal2Ap=Iub1)
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc43 1/43  U3P1000M80       11 1  IubLink=1,NbapCommon=1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc44 1/44  U3P1000M80       11 1  IubLink=1,NbapDedicated=1
AtmPort=MS-26-1,VplTp=1,.*VclTp=vc45 1/45  U3P1000M80       11 1  Aal2Sp=1,Aal2Ap=Iub1
...<cut>...
```

#### 4.2.7 stt[r] [<Filter>] [<stateFilter>]

Purpose: Display state and user of Physical Ports and Ds0Bundles.

Options:

- **r** : to refresh the data (ie. re-read from node).

Arguments:

- the first argument matches on the whole line
- the second argument matches only the state field ("PUI")



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Examples:

- `stt ms-26-1` print all ports and their users matching ms-26-1
- `stt . 0|L` print all ports and their users who have state disabled or locked
- `stt mtp3 busy` print all mtp3 links that are in usage state busy
- `stt mtp3 deact` print all mtp3 links that are in linkstate deactivated

Printout format:

- **PUI** : the first digit represents the state of the PhysicalPort/Ds0Bundle. The second digit (if present) represents the state of the User. The third digit (if present) represents the state of the intermediate layer which can be IMA (when user is AtmPort) or MTP2 (when user is Mtp3bSI). L=locked, 1=enabled, 0=disabled.
- **CG/KLM**: circuit group and K.L.M (for channelised STM-1)
- **SLC/PCM**: SignallingLinkCode for Mtp3bSI or pcmSystemNr for TdmTermGrp
- **User** : the layer that is using the physical port. Usually an AtmPort in Utran nodes. Can also be a TdmTermGrp or an Mtp3bSI in MGW. In case of Mtp3bSI, the linkState, proceduralState, and usageState are also shown.

Example printout RXI:

```
=====
Port                                     CG/KLM   PUI  USER
=====
Subrack=MS,Slot=7,...,Os155SpiTtp=pp1,Vc4Ttp=1          11  AtmPort=MS-7-1
...
Subrack=MS,Slot=24,...,Os155SpiTtp=pp1,Sts1SpeTtp=1,Vt15Ttp=1,T1Ttp=1  1/1.1.1  11  AtmPort=MS-24-1-1-1
Subrack=MS,Slot=24,...,Os155SpiTtp=pp1,Sts1SpeTtp=1,Vt15Ttp=2,T1Ttp=1  1/1.1.2  111 AtmPort=MS-24-ima1
Subrack=MS,Slot=24,...,Os155SpiTtp=pp1,Sts1SpeTtp=1,Vt15Ttp=3,T1Ttp=1  1/1.1.3  111 AtmPort=MS-24-ima1
=====
```

Example printout MGW:

```
=====
Port                                     CG/KLM   SLC/PCM PUI  USER
=====
2,Slot=25,...,ElPhysPathTerm=2251,Ds0.*=22511          1  LL  TdmTermGrp=El_MSC3_Slot25_Port1_TS1-29_31
2,Slot=25,...,ElPhysPathTerm=2252,Ds0.*=22521          2  1L  TdmTermGrp=El_MSC3_Slot25_Port2_TS1-29_31
2,Slot=25,...,ElPhysPathTerm=2253,Ds0.*=22531          3  11  TdmTermGrp=El_MSC3_Slot25_Port3_TS1-29_31
2,Slot=25,...,ElPhysPathTerm=2253,Ds0.*=22532          0  111 Mtp3bSpItu=2.*Sls=msc3.*SlItu=msc3_0 (available,initialized,active)
2,Slot=25,...,ElPhysPathTerm=2254,Ds0.*=22541          4  11  TdmTermGrp=El_MSC3_Slot25_Port4_TS1-29_31
=====
```

#### 4.2.8 ste[gr] [<Filter>] [<stateFilter>]

Purpose: Display state and configuration of Ethernet Ports.

Options:

- `r`: to refresh the data (clear cache and re-read from node).
- `g`: to print the RSTP tree in graphical format.

Arguments:

- the first argument matches on the whole line.
- the second argument matches only the state field ("S12" or "STL")

Command examples:

- `ste !nolink` : show all lines except those containing the word "nolink"
- `ste . 0|L` : show all lines containing a resource whose state is disabled or locked
- `ste forwarding` : show all lines containing the word "forwarding"

**Printout format, first table:** This table shows the properties for ethernet ports where IP distribution is performed, ie, where IpInterfaces are defined. Each line corresponds to a GigabitEthernet MO (ETIPG/ETMFG) or a InternalEthernetPort MO (ETMFx). Each column corresponds to an attribute of these MOs. Refer to the MOM for more detail on each attribute.

- Position: the subrack, slot, and port number - GigabitEthernet::portNo or 0 for InternalEthernetPort
- Speed: GigabitEthernet::actualSpeedDuplex
- Conf: GigabitEthernet::configuredSpeedDuplex
- AutoNg: GigabitEthernet::autoNegotiation
- Mastr: GigabitEthernet::masterMode
- Prot: GigabitEthernet::protectiveMode
- DfRSw: GigabitEthernet::defRoutersLinkSwitch



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- S: GigaBitEthernet::administrativeState&operationalState (0=unlocked&disabled, 1=unlocked&enabled, L=locked)
- 1: GigaBitEthernet::link1State
- 2: GigaBitEthernet::link2State
- ActiveLink: GigaBitEthernet::activeLink
- Link: GigaBitEthernet::linkType
- Frame: GigaBitEthernet::frameFormat or InternalEthernetPort::frameFormat
- Vlans: the list of vlan ids for all IpInterfaces defined on that port (read from attribute vid or vlanRef on the children IpInterface MOs). A vlan id value of -1 means that vlan is not used on that port ("vlan=false")
- DscpPbitMap: GigaBitEthernet::dscpPbitMap or InternalEthernetPort::dscpPbitMap (only the pbit values are listed)

Example:

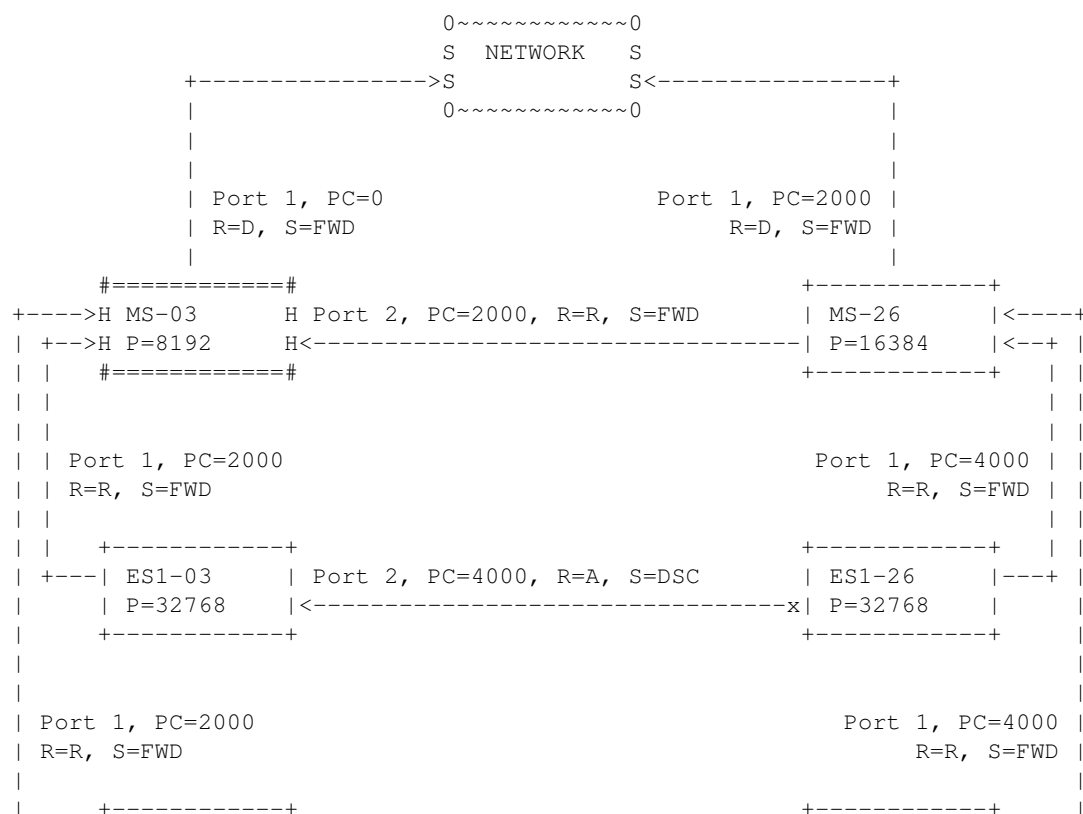
```

=====
Board Position Speed Conf AutNg Mastr Prot DfRsw S12 ActiveLink Link Frame Vlans DscpPbitMap
=====
MFG MS-06-1 1G_F 1G_F true true true false 111 1 (PRIMARY) FRONT 2DIX 20 0000000000101010003030300040404000505050000000600000000000000000
MFG MS-07-1 1G_F 1G_F true true true false 111 1 (PRIMARY) FRONT 2DIX 20 0000000000101010003030300040404000505050000000600000000000000000
MFX12 MS-24-0 2DIX 19 0000000000101010003030300040404000505050000000600000000000000000
MFX12 MS-25-0 2DIX 19 0000000000101010003030300040404000505050000000600000000000000000
=====

```

**Printout format, second table:** Each line corresponds to a EthernetSwitch or EthernetSwitchPort MO (ETMFX), or a EthernetSwitchModule or EthernetSwitchModulePort MO (CMXB), or a EthernetBridgePort (CMXB/CCIB in CAX subrack), as well as the corresponding children MOs SwitchStp/SwitchPortStp. Each column corresponds to an attribute of these MOs. Refer to the MOM for more detail on each attribute.

- Position: the subrack, slot, and port number:
  - 0: EthernetSwitch(Module)
  - 1-7: EthernetSwitchPort::portNo
  - 1-8: EthernetSwitchModulePort::portNo
- Lag: The position of the port specified in masterPort of the connected Lag MO, when applicable.
- lagSp: Lag::aggregatedPortSpeed, the speed of the Lag
- Remote: SwitchStp::rootBridgeld or SwitchPortStp::remoteBridgeld -> if the Bridge Id is pointing to a switch inside the node, the MAC address of the remoteBridge/rootBridge will be translated into the position of the switch. If the MAC address is not found within the node but pointing to an outside switch, then the MAC address given in the remoteBridgeld will be printed.
- Speed: EthernetSwitch(Module)Port::actualSpeedDuplex
- Conf: EthernetSwitch(Module)Port::operatingMode:configuredSpeedDuplex
- AutoNg: EthernetSwitch(Module)Port::operatingMode:autoNegotiation
- Sys/Ext: EthernetSwitchPort::systemPort or EthernetSwitchModulePort::externalPort or EthernetBridgePort::externalPort
- S: EthernetSwitch(Module)Port::administrativeState&operationalState (0=unlocked&disabled, 1=unlocked&enabled, L=locked)
- T: EthernetSwitch(Module)Port::trafficState
- L: Lag:administrativeState&operationalState. Only applicable when the Port(s) are part of a Lag MO.
- Prio: SwitchStp::bridgePriority or SwitchPortStp::priority
- Cost: SwitchPortStp::actualPathCost
- RtCost: SwitchPortStp::rootPathCost
- Role-State: SwitchPortStp::stpRole and SwitchPortStp::stpState
- Edge: SwitchPortStp::edgePortMode
- PbitQMap: EthernetSwitch(Module)(Port)::pbitQueueMap
- UnIng: EthernetSwitchPort::untaggedIngressVid&untaggedIngressPriority or EthernetSwitchModulePort::untaggedIngressVlanRef&untaggedIngressPriority. Shows the vid and priority that will be assigned to untagged ingress frames.
- Vlans: EthernetSwitchPort::vlanMembership or EthernetSwitchModulePort::vlanRef&egressUntagVlanRef. Shows the list of vlan ids supported by the port. Vlans on which egress frames will be untagged will be marked with a "U", eg "23U" means that vlan id 23 will be untagged on egress.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```

+-----| ES2-03      | Port 2, PC=4000, R=A, S=DSC      | ES2-26      |-----+
| P=32768    | <-----x-----| P=32768    |
+-----+          +-----+

```

#### 4.2.9 sti[bcfopr] [<Filter>] [<stateFilter>]

Purpose: Display state and configuration of IP interfaces.

The printout consists of up to 7 tables:

- the IpInterfaces table, one line per IpInterface MO.
- the IpAccessHostEt table, one line per IpAccessHostEt MO.
- the IpAccessHostGpb/Spb table, one line per IpAccessHostGpb or IpAccessHostSpb MO.
- the IpAccessHostPool table, one line per IpAccessHostEt MO part of an IpAccessHostPool (applicable to RNC and MGW only).
- the IpEthPacketDataRouter table, one line per IpEthPacketDataRouter MO (applicable to RNC38xx only).
- the M3uA table, one line per M3uAssociation MO (applicable to RNC and MGW only).
- the lub/S1/X2 table, one line per MO of type lubLink,lub,TermPointToMMe,TermPointToENB (applicable to RNC/RBS/ERBS only).

Options:

- p: ping the remote destinations (printout will take more time to complete)
- r: to refresh the data (clear cache and re-read from node).
- f: only show the IpInterface/IpAccessHost overview table. Without this option, all tables are shown.
- o: show the IpAccessHost frolds in the above tables.
- b: only show the lub/S1/X2 signalling interfaces (in RNC/RBS/ERBS). Without this option, all tables are shown.
- c: only show the Core signalling interfaces (in RNC/MGW: SIGTRAN). Without this option, all tables are shown.

Arguments:

- the first argument matches on the whole line
- the second argument matches only the state field ("GS12MUP" or "GS12ISP" or "IRP")

#### Printout format, IpInterfaces table:

Each line corresponds to a IpInterface MO. Each column corresponds to an attribute of these MOs. Refer to the MOM for more detail on each attribute.

- Board: The type of ET board on which the IpInterface is located (ETIPG, ETMFG, ETMFX)
- Interface: the subrack and slot of the ET board, followed by a sequential number to distinguish between numerous IpInterfaces defined on the same board.
- Vid: the vlan ID, read from the attribute IpInterface::vid or IpInterface::vlanRef
- Subnet: the value of the attributes IpInterface::subnet and IpInterface::networkPrefixLength
- DefaultRouter: the number in brackets shows which is the active defaultRouter according to the attribute IpInterface::defaultRouterTraffic. Also shown is the ip address of the active default router, read from the attribute IpInterface::defaultRouterX (where X is 0, 1, or 2)
- rps: the value of IpInterface::rps
- I: the value of IpInterface::operationalState (0=disabled, 1=enabled)
- R: the value of IpInterface::defaultRouterXState (where X is 0, 1, or 2). Only applicable when rps=true, otherwise a "-" is shown.
- P: the ping status to the active defaultRouter (0=unreachable, 1=alive)
- H: the state of each IpAccessHost connected to this IpInterface.
- IpHosts: the list of IP hosts connected to this IpInterface. G=IpAccessHostGpb, Et=IpAccessHostEt, S=IpAccessHostSpb. For each IpAccessHost, the location of the host is shown, not the MO name. To see the mapping of the host location vs MO name, check the following two tables. In brackets next to IpAccessHostEt is shown the IpAccessHostPool using this host, when applicable.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

| Board | Interface | Vid  | Subnet            | DefaultRouter      | Rps   | IRP | HHH | IpHosts                      |
|-------|-----------|------|-------------------|--------------------|-------|-----|-----|------------------------------|
| IPG   | MS-04-1   | 632  | 10.164.233.64/29  | (0) 10.164.233.70  | false | 1-1 | 1   | G=MS-5-1                     |
| IPG   | MS-04-2   | 652  | 10.164.233.0/26   | (0) 10.164.233.61  | true  | 111 | 111 | S=MS-10-2 S=MS-23-1 S=MS-9-1 |
| IPG   | MS-04-3   | 662  | 10.164.233.128/26 | (0) 10.164.233.188 | false | 1-1 | 1   | Et=MS-04-3 (IuB)             |
| IPG   | MS-04-4   | 682  | 10.164.233.96/27  | (0) 10.164.233.125 | true  | 111 | 1   | Et=MS-04-4 (IuR)             |
| IPG   | MS-04-5   | 2001 | 192.168.101.0/24  | (0) 192.168.101.1  | false | 1-1 | 1   | Et=MS-04-5 (intraNode)       |

**Printout format, IpAccessHostEt table:**

Each line corresponds to a IpAccessHostEt MO. Description of the columns:

- ET: The type of ET board on which the IpAccessHostEt is located (ETIPG, ETMFG, ETMFX)
- Host: the subrack and slot of the ET board, followed by a sequential number to distinguish between numerous IpAccessHostEt defined on the same board.
- MOName: the name of the IpAccessHostEt MO.
- Lnh: the linkhandler address of the ET board. Needed in order to run the EtHostMo\_startPing/EtHostMo\_startTraceRoute command.
- Ntp: the value of attribute ntpDscp. Only shown when ntpServerMode is enabled on this host. If ntpServerMode is disabled then a dash is shown instead.
- Fro: the frold of the IpAccessHostEt. Needed in order to run the EtHostMo\_startPing/EtHostMo\_startTraceRoute command. Only shown with option o.
- HostIp: the IP address of the IpAccessHostEt.
- Vid: the vlan ID of the IpInterface connected to this IpAccessHostEt.
- H: the state of the IpAccessHostEt (L=locked, 1=enabled, 0=disabled).
- IRP: the state of the IpInterface MO connected to this IpAccessHostEt. See description in table above.
- IpAccessHostPool/IpAccessSctp: the list of MOs using this host, first the Pools are listed, then the IpAccessSctp. For the IpAccessSctp, it shows the position of the GPB on which the SCTP is located, not the MO name.

| ET  | Host       | MOName  | Lnh    | Ntp | Fro | HostIp         | Vid  | HIRP | IpAccessHostPool/IpAccessSctp |
|-----|------------|---------|--------|-----|-----|----------------|------|------|-------------------------------|
| IPG | Et=MS-04-2 | MS-4-1  | 000400 | -   | 1   | 192.168.101.4  | 2001 | 11-1 | intraNode                     |
| IPG | Et=MS-04-3 | MS-4-3  | 000400 | 49  | 2   | 10.212.48.5    | 700  | 11-1 | Iub MS-06,MS-08,MS-12,MS-16   |
| IPG | Et=MS-04-4 | MS-4-2  | 000400 | -   | 3   | 10.202.212.3   | 812  | 1111 | Iu_Iur                        |
| IPG | Et=MS-25-2 | MS-25-1 | 002500 | -   | 4   | 192.168.101.25 | 2001 | 11-1 | intraNode                     |
| IPG | Et=MS-25-3 | MS-25-3 | 002500 | 49  | 5   | 10.212.48.6    | 700  | 11-1 | Iub MS-06,MS-08,MS-12,MS-16   |

**Printout format, IpAccessHostGpb/Spb table:**

Each line corresponds to a IpAccessHostGpb/Spb MO. Description of the columns:

- Host: the subrack and slot of the GPB/SPB where the IP host is located.
- Board: the type of board where the IP host is located.
- MOName: the name of the IpAccessHostGpb/Spb MO.
- HostIp1/2: the IP addresses of the IP Host.
- Interface1/2: the position of the IpInterfaces connected to the IP Host.
- Vid1/Vid2: the vlan ID of the IpInterfaces connected to the IP Host.
- H: the state of the IP Host (L=locked, 1=enabled, 0=disabled).
- IRP1/2: the state of the IpInterfaces connected to the IP Host (see detailed description of the IRP state in first table).

| Host     | Board | MOName  | HostIp1      | HostIp2      | Interface1 | Interface2 | Vid1 | Vid2 | H | IRP1 | IRP2 |
|----------|-------|---------|--------------|--------------|------------|------------|------|------|---|------|------|
| G=MS-08  | GPB53 | MS-8    | 10.159.22.2  | 10.159.22.18 | MS-26-2    | MS-27-2    | 929  | 929  | 1 | 1-1  | 1-1  |
| G=ES1-12 | GPB53 | ES-1-12 | 10.212.0.27  | 10.212.0.28  | ES1-03-1   | ES1-26-1   | 100  | 100  | 1 | 1-1  | 1-1  |
| G=ES1-13 | GPB53 | ES-1-13 | 10.212.0.29  | 10.212.0.30  | ES1-03-1   | ES1-26-1   | 100  | 100  | 1 | 1-1  | 1-1  |
| S=MS-19  | SPB3  | MS-19   | 10.159.18.7  | 10.159.18.8  | MS-26-3    | MS-27-3    | 945  | 945  | 1 | 111  | 111  |
| S=MS-20  | SPB3  | MS-20   | 10.159.18.9  | 10.159.18.10 | MS-26-3    | MS-27-3    | 945  | 945  | 1 | 111  | 111  |
| S=MS-21  | SPB3  | MS-21   | 10.159.18.11 | 10.159.18.12 | MS-26-3    | MS-27-3    | 945  | 945  | 1 | 111  | 111  |

**Printout format, IpAccessHostPool table (RNC/MGW):**

Each line corresponds to a IpAccessHostEt MO used by an IpAccessHostPool. Each column corresponds to an attribute of these MOs.

- Pool: the name of the IpAccessHostPool
- HostEt: the position (Subrack-Slot) of the IpAccessHostEt MO, followed by a sequential number to distinguish between numerous IpAccessHostEt defined on the same board.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- HostIp: the attribute IpAccessHostEt::ipAddress
- ET: the board type (ETIPG, ETMFG, or ETMFX)
- Vid: the vlan ID of the IpInterface connected to this IpAccessHostEt.
- P: the state of the IpAccessHostPool MO (L=locked, 1=unlocked&enabled, 0=unlocked&disabled)
- H: the state of the IpAccessHostEt MO
- I: the state of the IpInterface MO connected to this IpAccessHostEt, same as shown in the first table
- R: the state of the active defaultRouter, same as shown in the first table
- P: the ping status to the active defaultRouter, same as shown in the first table
- Users: read from the attribute IpAccessHostPool::reservedBy. Shows "Rnc" if the pool is used by the RncFunction MO (applicable to intraNode/BEES pool), else shows the number of lubLinks/lurLinks/luLinks using this pool.

RNC:

| Pool      | HostEt  | HostIp         | ET  | Vid  | PH | IRP | Users: | Iuc | Iur | Iub |
|-----------|---------|----------------|-----|------|----|-----|--------|-----|-----|-----|
| IUB       | MS-04-3 | 10.164.233.129 | IPG | 662  | 11 | 1-1 |        | 0   | 0   | 34  |
| IUB       | MS-25-3 | 10.164.233.130 | IPG | 662  | 11 | 1-1 |        | 0   | 0   | 34  |
| IUR       | MS-04-4 | 10.164.233.97  | IPG | 682  | 11 | 111 |        | 0   | 24  | 0   |
| IUR       | MS-25-4 | 10.164.233.98  | IPG | 682  | 11 | 111 |        | 0   | 24  | 0   |
| intraNode | MS-04-5 | 192.168.101.4  | IPG | 2001 | 11 | 1-1 | Rnc    | 0   | 0   | 0   |
| intraNode | MS-25-5 | 192.168.101.25 | IPG | 2001 | 11 | 1-1 | Rnc    | 0   | 0   | 0   |

MGW:

| Pool | Type | HostEt    | HostIp       | ET  | Vid  | PH | IRP |
|------|------|-----------|--------------|-----|------|----|-----|
| A    | A    | MAIN-06-2 | 10.52.211.10 | IPG | 1180 | 11 | 1-  |
| A    | A    | MAIN-14-1 | 10.52.211.11 | IPG | 1180 | 11 | 1-  |
| A    | A    | MAIN-17-1 | 10.52.211.12 | IPG | 1180 | 11 | 1-  |
| A    | A    | MSE1-14-2 | 10.52.211.13 | IPG | 1180 | 11 | 1-  |
| Iu   | IU   | MAIN-06-2 | 10.52.211.10 | IPG | 1180 | 11 | 1-  |
| Iu   | IU   | MAIN-14-1 | 10.52.211.11 | IPG | 1180 | 11 | 1-  |
| Iu   | IU   | MAIN-17-1 | 10.52.211.12 | IPG | 1180 | 11 | 1-  |
| Iu   | IU   | MSE1-14-2 | 10.52.211.13 | IPG | 1180 | 11 | 1-  |

### Printout format, IpEthPacketDataRouter table (RNC):

Each line corresponds to a IpEthPacketDataRouter MO. Each column corresponds to an attribute of these MOs.

- IpEthPdr: the position (Subrack-Slot) of the PdrDevice, followed by the attribute IpEthPacketDataRouter::ipAddressSelection. Value 1 means IP\_ADDRESS\_1, value 2 means IP\_ADDRESS\_2. Value 11 means that both IpEthPacketDataRouter of that PdrDevice are using IP\_ADDRESS\_1 while 22 means that both are using IP\_ADDRESS\_2 (which would mean that all traffic of the PdrDevice is routed to the same IpInterface instead of being load-shared on both IpInterfaces)
- SPB: the board type of the SPB hosting the PdrDevice
- HostSpb: the position of the IpAccessHostSpb connected to this IpEthPacketDataRouter, followed by the number 1 or 2, depending on the value of IpEthPacketDataRouter::ipAddressSelection
- HostIp: the ip address of the IpAccessHostSpb, could be ipaddress 1 or 2, depending on the value of IpEthPacketDataRouter::ipAddressSelection
- ET: the board type of the IpInterface connected to this IpAccessHostSpb (ETIPG, ETMFG, or ETMFX)
- Vid: the vlan ID of the IpInterface connected to this IpAccessHostSpb
- E: the state of the IpEthPacketDataRouter MO (L=locked, 0=unlocked&disabled, 1=unlocked&enabled)
- H: the state of the IpAccessHostSpb MO
- U: the value of the usageState of the PdrDevice hosting this IpEthPacketDataRouter (I=Idle, A=Active, B=busy)
- I: the state of the IpInterface MO connected to this IpAccessHostEt, same as shown in the first table
- R: the state of the active defaultRouter, same as shown in the first table
- P: the ping status to the active defaultRouter, same as shown in the first table
- Iup: the value of IpEthPacketDataRouter::reservedBy. Shows which IP-based luPS links can use this IpEthPacketDataRouter for userplane connections. When empty, means that it can be used by all IP-based luPS links.

| IpEthPdr | SPB   | HostSpb | HostIp          | ET  | Interface | Vid  | EH | U | IRP | Iup |
|----------|-------|---------|-----------------|-----|-----------|------|----|---|-----|-----|
| MS-19-1  | SPB21 | MS-19-1 | 114.126.135.176 | MFG | MS-07-1   | 2000 | 11 | A | 1-1 | any |
| MS-19-2  | SPB21 | MS-19-2 | 114.126.135.181 | MFG | MS-26-1   | 2000 | 11 | A | 1-1 | any |
| MS-20-1  | SPB21 | MS-20-1 | 114.126.135.177 | MFG | MS-07-1   | 2000 | 11 | A | 1-1 | any |
| MS-20-2  | SPB21 | MS-20-2 | 114.126.135.182 | MFG | MS-26-1   | 2000 | 11 | A | 1-1 | any |

### Printout format, M3uAssociation table (RNC/MGW):

Each line corresponds to a M3uAssociation MO. Each column corresponds to an attribute of these MOs.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- the HostGpb column identifies the IpAccessHostGpb MO.
- the Sctp column identifies the Sctp MO
- the M3uA column identifies the M3uAssociation MO. At the beginning of the string is the identity of the parent Mtp3bSp MO, then comes the identity of the M3uAssociation MO. A wildcard (.) separates the two identities.
- the Assoc column identifies the sctp association, given by the the gpb position and the association number. The association number can be used in the printout `sctphost_info -assoc <assocnumber>`.
- the GS12MUP column shows the various states, where: G=IpAccessHostGpb, S=Sctp, 1=IpInterface1, 2=IpInterface2, M=M3uAssociation, U=User, P=ipac\_ping (1=alive, 0=notalive). The ping is done only when "sti" is run with the option "p".
- the A column shows the associationState of the M3uAssociation, where: I=inactive, A=active, E=established, D=down.
- the LocalInterface column shows the local interface, can be a IpInterface MO or a FastEthernet MO. In the case of FastEthernet, the letter "f" is appended to the identity as shown in the third line of the printout below. In brackets is indicated whether the chosen interface is interface 1 or interface 2. The association tries to setup on interface 1 but if that fails then it uses interface 2.
- the RemoteInterface column shows which interface is used on the remote side, 1 or 2.
- the LocalIp:Port and RemIp:Port columns show the ipaddress and sctp port of the association, for each side.
- ther User column shows which application part is using this association. Could be GCP (Vmgw), Q.AAL2 (Aal2Ap), RANAP, RNSAP.

#### Example MGW:

```

=====
HostGpb   Sctp   M3uA       Assoc   GS12 MUP A LocalInterface,Ip:Port   RemInterface,Ip:Port   User
=====
IPACGPB_2_6 Sctp_2_6 2..msc2qa 0006:148 1111 111 A (1) 2-19 10.201.0.31:2905 (2) 10.202.10.20:2905 Vmgw=VMGW92
IPACGPB_2_6 Sctp_2_6 2..nwp141a 0006:60 1111 111 D (2) 2-20 10.201.10.31:2905 (1) 10.201.12.141:2905 Aal2Sp=1,Aal2Ap=nwp141
SIG12_13 SIG12_13 2..nwp206a 0012:5 1111 111 A (1) 1-12f 10.202.0.30:2905 (1) 10.202.2.206:2905 Aal2Sp=1,Aal2Ap=nwp206
=====

```

#### Example RNC:

```

=====
HostGpb   Sctp   M3uA       Assoc   GS12 MUP A LocalInterface,Ip:Port   RemInterface,Ip:Port   User
=====
MS-8-1 MS-8 Iu.*mgw7-2521-1720 0008:21 1111 111 A (1) MS-7 10.207.2.121:2905 (1) 10.207.2.245:2905 Aal2Ap=Aal2routing-mgw7-2521-1720
MS-8-1 MS-8 Iu.*mgw71-2521-1730 0008:4 1111 111 A (1) MS-7 10.207.2.121:2905 (1) 10.207.2.246:2905 Aal2Ap=Aal2routing-mgw71-2521-1730
MS-8-1 MS-8 Iu.*Iuc-2521-1700 0008:18 1111 111 A (1) MS-7 10.207.2.121:2905 (2) 10.207.12.240:2905 Cn.*=23591,Iu.*=Iuc-1700,Ranap=Iuc-1700
MS-8-1 MS-8 Iu.*Iuc-2521-1710 0008:2 1111 111 A (1) MS-7 10.207.2.121:2905 (1) 10.207.2.242:2905 Cn.*=23591,Iu.*=Iuc-1710,Ranap=Iuc-1710
MS-8-1 MS-8 Iu.*Iup-2521-2207 0008:33 1111 111 A (1) MS-7 10.207.2.121:2905 (2) 10.207.4.1:2905 Cn.*=23591,Iu.*=Iup-2207,Ranap=Iup-2207
MS-8-1 MS-8 Iu.*Iup-2521-2271 0008:34 1111 111 A (1) MS-7 10.207.2.121:2905 (2) 10.207.4.17:2905 Cn.*=23591,Iu.*=Iup-2271,Ranap=Iup-2271
MS-8-1 MS-8 Iu.*Iur-2521-2522 0008:22 1111 111 A (1) MS-7 10.207.2.121:2905 (2) 10.207.12.122:2905 Iur.*=Iur-2521-2522,Rnsap=Iur-2521-2522
=====

```

**Printout format, lubLink table (RNC):** This table shows the configuration and status of control plane connections for IP-based lubLinks (for lub user plane connections refer to the IpAccessHostPool table). Each line corresponds to a lubLink MO in RNC. Each column corresponds to an attribute of these MOs.

- Sctp: the value of lubLink::sctpRef. The position of the GPB hosting this Sctp is shown.
- Mod: the value of lubLink::rncModuleRef.
- Host: the value of Sctp::ipAccessHostGpbId or Sctp::ipAccessSctpRef. Shows "Gpb" if IpAccessHostGpb is used or "Et" if IpAccessSctp is used.
- Interf1/Interf2: the position of the IpInterfaces used by the IpAccessHost(s).
- LocalIp1/LocalIp2: the ip addresses of the IpAccessHost(s) used by the Sctp.
- Remotelp: the value of lubLink::remoteCplpAddress1
- lubLink: the name of the lubLink MO.
- G: the state of the IpAccessHostGpb or IpAccessSctp used by the Sctp MO (0=locked, 1=unlocked&enabled, 0=unlocked&disabled)
- S: the state of the Sctp MO
- 1: the state of IpInterface 1
- 2: the state of IpInterface 2
- I: the state of the lubLink MO
- S: the state of the NodeSynch MO
- PP: the ping status from each IpInterface to the Remotelp (0=unreachable, 1=alive). The first "P" corresponds to the ping status from IpInterface 1, the second "P" corresponds to the ping status from IpInterface 2.
- the NbapC\_Assoc and NbapD\_Assoc fields identify the sctp associations for NbapCommon and NbapDedicated. First number is the local port number, then the remote port number, then the SCTP association reference number according to

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

the printout of `sctphost_info -assoc -all`. The number in brackets identifies the active `IpInterface` used for this association.

| Sctp  | Mod | Host | Interf1 | Interf2 | LocalIp1     | LocalIp2     | RemoteIp     | IubLink | GS12 | ISPP | NbapC_Assoc   | NbapD_Assoc   |
|-------|-----|------|---------|---------|--------------|--------------|--------------|---------|------|------|---------------|---------------|
| MS-14 | 1   | Gpb  | MS-25-1 | MS-26-1 | 10.100.0.140 | 10.100.1.140 | 10.100.2.150 | Iub-1   | 1111 | 1111 | 1:5101:30 (1) | 2:5102:23 (1) |
| MS-14 | 1   | Gpb  | MS-25-1 | MS-26-1 | 10.100.0.140 | 10.100.1.140 | 10.100.2.150 | Iub-10  | 1111 | 1111 | 1:5119:33 (1) | 2:5120:26 (1) |

#### Printout format, lub table (RBS):

- LocalIp: ipaddress of the `IpAccessHostGpb` or `IpAccessHostEt` used for lub control plane
- RemoteIp1/RemoteIp2: ip addresses used on the remote side (RNC) according to the printout of `sctphost_info -assoc -all`
- G: state of `IpAccessHostGpb` or `IpAccessSctp` MO
- S: state of `Sctp` MO
- I: state of `IpInterface` MO
- C: state of `NbapCommon` MO
- D: state of `NbapDedicated` MO
- P: ping status, one ping for each remotelp (0=unreachable, 1=alive)
- the `NbapC_Assoc` and `NbapD_Assoc` fields identify the sctp associations for `NbapCommon` and `NbapDedicated`. First number is the local port number, then the remote port number, then the SCTP association reference number according to the printout of `sctphost_info -assoc -all`. The number in brackets identifies the active `IpInterface` used for this association.

| LocalIp     | RemoteIp1  | RemoteIp2  | IubLink | GS1 | CDPP | NbapC_Assoc | NbapD_Assoc |
|-------------|------------|------------|---------|-----|------|-------------|-------------|
| 10.2.35.143 | 10.2.35.16 | 10.2.35.17 | Iub=1   | 111 | 1111 | 5113:1 (2)  | 5114:2 (1)  |

#### Printout format, S1/X2 interfaces in ERBS:

- T: state of the `TermPoint` MO
- S: state of the `Sctp` MO
- H: state of the `IpAccessHostEt` MO
- P: ping status to the active remote IP address (0=unreachable, 1=alive)
- Assoc: the reference number of the SCTP association according to the printout of `sctphost_info -assoc -all`
- TermPoint: "ENB" refer to `TermPointToENB` MO (X2), "Mme" refer `TermPointToMME` MO (S1).

| LocalIp:Port      | RemoteIp:Port      | StandbyRemoteIp | TSHP | Assoc | TermPoint           |
|-------------------|--------------------|-----------------|------|-------|---------------------|
| 10.62.11.34:36422 | 10.62.11.33:36422  | 10.62.11.34     | 1111 | 71    | ENB=104023          |
| 10.62.11.34:36422 | 10.64.193.81:36412 | 10.62.11.82     | 1111 | 68    | Mme=MME010064193081 |
| 10.62.11.34:36422 | 10.64.193.91:36412 | 10.62.11.92     | 1111 | 70    | Mme=MME010064193091 |

#### 4.2.10 sts

Purpose: Display state and configuration of Network Synchronization.

The printout is read from the Synchronization MO. The first line corresponds to the value of the attribute `nodeSystemClock`. The remaining lines correspond to the values of the attributes `syncReference`, `syncRefPriority`, `syncRefActivity`, `syncRefStatus`.

Examples:

```
RNC01> sts
```

```
SystemClock: LOCKED_MODE
```

| Prio | Activity | RefState | AdmState | OpState | SyncReference                                              |
|------|----------|----------|----------|---------|------------------------------------------------------------|
| 1    | ACTIVE   | OK       | UNLOCKED | ENABLED | Subrack=MS,Slot=4,PlugInUnit=1,TimingUnit=1,TuSyncRef=1    |
| 2    | INACTIVE | OK       | UNLOCKED | ENABLED | Subrack=ES-2,Slot=2,PlugInUnit=1,ExchangeTerminal=1,Os155  |
| 3    | INACTIVE | OK       | UNLOCKED | ENABLED | Subrack=ES-2,Slot=27,PlugInUnit=1,ExchangeTerminal=1,Os155 |



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```

4      INACTIVE  OK          UNLOCKED  ENABLED   Subrack=ES-3,Slot=2,PlugInUnit=1,ExchangeTerminal=1,Os155
5      INACTIVE  OK          UNLOCKED  ENABLED   Subrack=ES-3,Slot=27,PlugInUnit=1,ExchangeTerminal=1,Os155

```

```
RBS14> sts
```

```
SystemClock: HOLD_OVER_MODE
```

```

-----
Prio  Activity  RefState  AdmState  OpState   SyncReference
-----
1      INACTIVE  FAILED    UNLOCKED  DISABLED   IpAccessHostEt=1,IpSyncRef=1
2      INACTIVE  FAILED    UNLOCKED  DISABLED   IpAccessHostEt=1,IpSyncRef=2

```

#### 4.2.11 str

Print status of the lubLinks/AbisLinks and their associated Cells and Channels (RNC/BSC only).

The command has two syntaxes, depending on the type of node.

##### **CDMA BSC: str [ | <unix-cmds>]**

```

-----
SITE C1 C2 C3 ABIS BACKHAUL          ATMPORTS
-----
96   11 11 11 11   RBS1_Backhaul BHRBS1_Backhaul_BHSBackhaulSpan_1
-----

```

The states of the channels are shown for each cell, as well as the states of the **AbisCommon** and **AbisDedicated**.

The MO-id of the **BackHaul** and the **AtmPorts** are also shown for each site.

State abbreviation: L means Locked, 0 means Disabled, and 1 means Enabled.

##### **UTRAN RNC: str[12ft] [<csvfile>] [<filter-options>] [ | <unix-cmds>]**

To see the state of all or part of the cells/iubs/channels in the node, one line per site.

The filter options (-m, -s, -i, -c, -g, -t) allow to get states on only part of the sites/cells, in order to speed up the output. For example:

- `str -m 7,8,9` - print states only for modules 7, 8, and 9
- `str -s ms,es-1` - print states only for subracks ms and es-1
- `str -i 9012` - print states only for the MO lublink=9012 and its connected cells
- `str -c 90121,90131` - print states only for the lublink MOs connected to the MO UtranCell=90121 and UtranCell=90131
- `str -g clusterNorth` - print states only for the iublins or utrancells defined in the MO group "clusterNorth"
- `str -t a` - print states only for ATM-based iublins
- `str -t i` - print states only for IP-based iublins
- `str -t ai` - print states for dual stack iublins

There are four possible output formats:

The `str` printout uses an abbreviated naming of the cells where it is assumed that the last digit is identifying the sector. For networks where the sector is not identified by the last digit, it can be handy to use `str1` or `str2` since the whole cell name will then be shown for each sector.

The `strt` command shows the AtmPorts used by each site.

- `str`

```

-----
MOD  IUBLINK      CELLNAME      CFRPHEU1  CFRPHEU2  CFRPHEU3  ICDS  TN  R
-----
1    Iub_3011      3011-1/2/3    1111111  L000000   1000000   1111  I  P
-----

```

- `strt`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

| MOD | IUBLINK  | CELLNAME   | CFRPHEU1 | CFRPHEU2 | CFRPHEU3 | ICDS | TN | TNPORTS         |
|-----|----------|------------|----------|----------|----------|------|----|-----------------|
| 1   | Iub_3011 | 3011-1/2/3 | 1111111  | L000000  | 1000000  | 1111 | AI | MS-25-1 MS-26-1 |
| 8   | Iub_3012 | 3012-1/2/3 | 1111111  | 1111111  | 1111111  | 1111 | I  | MS-23 MS-24     |

- str1

| MOD | IUBLINK  | CELLNAMES         | CFRPHEU1 | CFRPHEU2 | CFRPHEU3 | ICDS |
|-----|----------|-------------------|----------|----------|----------|------|
| 1   | Iub_3011 | 30111 30112 30113 | 1111111  | L000000  | 1000000  | 1111 |

- str2

| MOD | IUBLINK  | ICDS | CELL1 | CFRPHEU | CELL2 | CFRPHEU | CELL3 | CFRPHEU |
|-----|----------|------|-------|---------|-------|---------|-------|---------|
| 1   | Iub_3011 | 1111 | 30111 | 1111111 | 30112 | 1111111 | 30113 | 1111111 |

- **MOD:** The RNC module that is handling the control plane for this lub, corresponds to the attribute lubLink::rncModuleRef
- **IUBLINK:** The MO name of the lubLink
- **CELLNAME:** The name of the cells that are connected to that lubLink. The cell names correspond to the respective sectors. Eg: 6306-1/2/3 means that: cell 63061 is connected to sector 1, cell 63062 is connected to sector 2, cell 63063 is connected to sector 3.
- **CFRPHEU:** The first digit is the state of the UtranCell MO. The following three digits are the state of the common channels (Fach/Rach/Pch). The fifth digit (if present) represents the state of the hsdpa channel (Hsdpsch). The sixth digit (if present) represents the state of the enhanced uplink channel (Eul). The seventh digit (if present) represents the state of the EulFach channel (EulFach).
- **ICDS:** The first digit is the state of the lubLink. The second digit is the state of NsapCommon MO or SctpAssociation MO handling Nsap Common. The third digit is the state of NsapDedicated MO or SctpAssociation MO handling Nsap Dedicated. The fourth digit is the state of NodeSynch MO.
- **TN:** The type of transport network used by the lubLink. A=ATM, I=IP, AI=DualStack
- **R:** The lub redundancy configuration of the lubLink, read from the attribute lubLink::poolRedundancy. P=Primary, S=Secondary, N=No Redundancy. Only applicable for RNC in Pool with lub Redundancy feature.
- **TNPORTS:** The Subrack and Slot of the ETIP (in case of IP lub) or ATMPORT (in case ATM or DualStack lub)

States values:

- **L** = Locked
- **S** = ShuttingDown
- **T** = TPS power locked (corresponds to the attribute tpsPowerLockState)
- **0** = Unlocked & Disabled
- **1** = Unlocked & Enabled

The output can be piped in external unix commands such as grep.

If a filename is given as argument, the output will be saved into this file in csv format (as well as being printed on the screen).

The *f* option (*fast*) is for printing without fetching the data. I.e. the data can be fetched once, then displayed in many different ways without having to fetch it again each time.

Examples:

```
strf | grep 3011
strlf | sort -k 2      (sort on the second field)
```

#### 4.2.12 hc

This command runs a general *healthcheck* on the node. Obsolete ! Use dcg command instead.

If no logfile is currently open, then a logfile will be automatically opened to capture the output of the hc command.

Please refer to the command file in **moshell/commonjars/scripts/hc\_datacollection.mos** to view the various commands that are run for the health check.

More info about each command can be found by typing `h <command>`.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.2.13 `dcg[meiasrfkx] [-m <rophours>] [-d <logdays>] [-b <boards|boardgroup>|all] [-k <nrdumps>] [-f <mofilter>] [<logdir>]`

Fetch data for TRs/CSRs, according to the Data Collection Guidelines.

The `dcg` command offers a number of options, it is possible to combine several options, eg: "`dcgmsr`"

Options:

- `m`: mandatory data. Includes mandatory printouts as well as offline files (modump, logfiles, pm ropfiles, dbdat)
- `e`: subset of the mandatory data which can be taken in case of emergency, before doing board/node restart. This option will usually be run on its own.
- `i`: IP printouts
- `a`: ATM/AAL2 printouts. By default only the AAL2 printouts are collected. To collect ATM printouts, use option "`-b`", see below.
- `s`: SS7 printouts
- `x`: SPAS printouts
- `r`: RNC specific printouts.
- `f`: fetch logfiles, ropfiles, and CV. Three zipfiles are produced which can be used in offline mode in `pmr`, `pmx`, `lg`, and `dbc`. Refer to the chapter "Offline mode" for more info.
- `k`: take MO dump (kget format). A zipfile is produced containing the MO dump and MOM of the node, it can be used in offline mode by running "`moshell <zipfile>`". Refer to the chapter "Offline mode" for more info.

Switches:

- `-m <rophours>`: the number of hours of ROP files to collect with `pmrf`, eg "`-m 2`". Default is 8 hours in `dcge` and 48 hours in `dcgm/dcgf`
- `-d <logdays>`: the number of days of logfiles to collect with `lgf`, eg "`-d 30`". Default is 60 days in `dcgm/dcgf`.
- `-b <boards|boardgroup>|all`: the ET boards on which `dcgi/dcga` will be run. When this option is not specified, `dcga` collects no ET board data, while `dcgi` collects all ET boards data. Example: `dcgi -b 000600,000700`
- `-k <nrdumps>`: the number of ENB DSP dumps to collect. (Corresponds to the option "`-d`" in `lg` command. Default: 0)
- `-f <mofilter>`: the MO filter for MO dump collection. Eg "`-f !relation=`" to skip MOs such as `UtranRelation/GsmRelation` in the MO dump.

Argument:

- the directory where the collected data will be stored. If no directory is given, the directory `~/moshell_logfiles/logs_moshell/dcg/<node>/<date>_<time>` is used.

Please refer to the command file in `moshell/commonjars/scripts/dcg_datacollection.mos` to view the various commands that are run for each option. More info about each command can be found by typing `h <command>`

Note: for CPP nodes, if the MO layer is unavailable, it is possible to run `dcg` anyway by typing `uv nocorba=1` before executing `dcg`. Using `nocorba=1` means that moshell will not attempt to connect to the MO service and will only run commands via telnet/ftp/ssh/sftp.

#### 4.2.14 `diff[a][d][m][o]/ldiff[a][d][m][o]`

Parameter auditing or MO dump comparisons.

**Syntax 1:**

Compare two or three MOs side by side. MOs must be of same *MO class*. All attribute values that are different between the MOs will be printed.

Example:

```
dif 4 32 17
```

Where 4, 32, 17 are the proxy identities of the MOs that should be compared.

**Syntax 2:**

```
diff[a][d][m][o]/ldiff[a][d][m][o] <moGroup>|<moFilter>|<proxy>|<modumpFile>|<modumpDir>
[<baselineFile>|<modumpFile2>|default] [<outputDir>]
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Purpose: To compare an MO dump with a parameter baseline file or with another MO dump.

Options:

- a: show the list of MOs and attributes that are in the reference but not in the node or dump.
- d: compare against the default values in MOM (read from command "momb")
- m: when comparing against parameter baseline, any parameter not found in the baseline will be compared against the MOM default values instead.
- o: when comparing two MO dumps, only the differences in configuration parameters will be shown (= attributes that are not readOnly)

Note: It is currently not supported to combine several options together.

When no option is specified, the attributes are compared against the recommended values in baseline (files moshell/commonjars/pm/PARAM\*)

User variables:

- `diffexclude_attributes` : to exclude certain attributes from the MO dump comparison (diff <dump1> <dump2>)
- `diffm_exclude_moclasses` : to exclude certain MO classes from the MOM default value comparison (diffd and diffm)
- `diffm_exclude_attributes` : to exclude certain attributes from the MOM default value comparison (diffd and diffm)
- `diffm_exclude_structs` : to exclude certain structs from the MOM default value comparison (diffd and diffm)

First Argument:

- `<moGroup>|<moFilter>|<proxy>`: to specify the MOs in the current node which should be used for the comparison.
- `<modumpFile>`: to specify an modump, taken from this node or another node. The modump file should be a zipfile taken by the command `dcgk`. Alternatively it can be a text or gzipped file containing the printout from the `get` or `kget` command.
- `<modumpDir>`: to specify a directory containing modump logfiles. Only files with the extension ".log" will be used in the comparison.

Second Argument (optional):

- empty or "default": a baseline parameter file from moshell/commonjars/pm/PARAM\* will be chosen according to the MOM type and version of the current node or of the modump specified in the first argument. Note: typing "default" is only necessary if one wants to specify a third argument (the outputDir).

If the uservariable `custom_param_file` is set then the file(s) specified in `custom_param_file` are read after the PARAM\* file, meaning that their values will override those of the PARAM\* file.

- `<baselineFile>`: a specific baseline file is used as reference, instead of the default one. In this case the file(s) specified in the uservariable `custom_param_file` are not read.

The format of a baseline file must consist of three words on each line: `<mo> <attribute> <value>`

The `<mo>` field can be either an MO class (e.g. `PlugInUnit`) or an MO LDN (e.g.

`Subrack=MS,Slot=1,PlugInUnit=1`)

It is also possible to write a tilde sign ( `~` ) in front of the MO LDN, in which case it is treated as a regular expression.

For instance: `~UniSaalTp=.*q[ab]$ maxSduSize 128` means that the reference value only applies to the MOs whose LDN matches that string.

The recommended value can be a single value or several values separated by commas. For instance `-1,2,25,300` : means any of these values is accepted as recommended value. If the recommended value is a sequence of Structs or sequence of moRefs, then it shall be written on one single line. Example:

```
EthernetSwitch      pbitQueueMap [8] = 0,1;1,0;2,1;3,1;4,2;5,2;6,3;7,3
```

More information about the reference file format can be found in the document

moshell/examples/audit\_files/EAB\_FJW-08\_0071.doc

- `<modumpFile>`: the two modump logfiles are compared and the following differences will be highlighted: differences in attribute values, MOs found in one dump but not the other, attribute names found in one dump but not the other.

Third Argument:

- `<outputDir>`: to specify the directory where to store the result files (csv comparison file and correction command file). If not specified, a default directory and file names are chosen.

Result:

The result is shown on screen with space-separated fields and also saved in the result directory with comma-separated fields (CSV). Also, in the result directory is a command file to align the current values to the baseline values. The path to the CSV result file is saved in the moshell variable `$diffcsvfile` and the path to the command file is saved in the moshell variable `$diffcmdfile`.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Please refer to the document `moshell/examples/audit_files/EAB_FJW-08_0071.doc` for more information.

Note about parameter baseline files:

Parameter baseline files are taken from the Winnow database and stored in CDM in excel format:

- RNC: 3/19059-HSD10102
- RBS: 4/19059-HSD10102
- RXI: 5/19059-HSD10102
- ENB: 19706-CXP102051/\*

To use these files as reference for comparison, they need to be converted to text. This can be done by copy pasting the excel sheet to a text file. Moshell keeps a text copy of the latest version of each of these files in the folder `moshell/commonjars/pm`. By not specifying the baseline parameter file in the "diff" command will make moshell choose the best suited file for the node type and mom version of the current node or modump file.

Examples:

- `diff .` - Compare all MOs with the relevant baseline parameter file stored in `moshell/commonjars/pm`.
- `diffa .` - Same as above but showing MO/attributes found in reference but not in node
- `diff . default ~/audit\_070110` - Same as above but store the results in the directory `/audit_070110`
- `diff ~/moshell_logfiles/logs_mobatch/2007-01-10/mysites/11-21` - Compare all modumps under that directory against the relevant baseline parameter file stored in `moshell/commonjars/pm`
- `ldiff msplatform=1 mgw\_parameters\_r4.txt` - Compare all MOs under `msplatform=1` in the current node with the baseline file "mgw\_parameters\_r4.txt"
- `diff . rnc10\_before\_upgrade.txt` - Compare all MOs in the current node with the MO dump "rnc10\_before\_upgrade.txt"
- `diff rnc10\_before\_upgrade.txt rnc10\_after\_upgrade.txt` - Compare the MO dumps "rnc10\_before\_upgrade.txt" and "rnc10\_after\_upgrade.txt"
- `diff rnc10_before_upgrade.txt rnc10_after_upgrade.txt` - Same as above but only the configuration parameters are compared
- `diffd utrancell` - compare all utrancell parameters against MOM default values.

#### 4.2.15 lkr[a]

Print RNC lub resources allocation.

##### Purpose:

- display the repartition of lubLinks and UtranCells across rncModules and Subracks, to identify any uneven resource allocations, for instance: rncModules that are handling more lub/Cells than others, or CC devices that are handling different Cells than their controlling module MP.
- uneven allocation of lubLinks across RncModules can be corrected by the command `resub iublink`
- uneven allocation of UtranCells across CC devices can be corrected by locking/unlocking the cells using command `b1` and `deb` on the UtranCell MOs.
- with the option `a`, an additional table shows the lubLinks whose AtmPort(s) are located in a different Subrack than the lub module resources. lubLinks can be moved to a different AtmPort or Subrack with the command `resub <iublink> <subrack> or resub <iublink> <atmport/vp>`. Type `h resub` for info.

##### Printout example and description:

The first table, only printed with option `a` (`lkra`), shows the list of lubLinks whose AtmPort(s) are in a different Subrack than the lub module resources. The `Sr` column shows the Subrack containing the module resources, the `Mod` column shows the module number, the `IubLink` column shows the lub, and the `AtmPort(s)` column shows the port(s) used by the lub. Example:

| Sr | Mod | IubLink | AtmPort(s)  |
|----|-----|---------|-------------|
| MS | 1   | Iub-198 | ES-1-27-2-1 |
| MS | 13  | Iub-203 | ES-1-27-2-1 |
| MS | 8   | Iub-208 | ES-1-27-2-1 |
| MS | 1   | Iub-213 | ES-1-27-2-1 |

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

|    |    |         |             |
|----|----|---------|-------------|
| MS | 13 | Iub-214 | ES-1-27-2-1 |
| MS | 1  | Iub-87  | ES-1-3-1-1  |
| MS | 8  | Iub-88  | ES-1-3-1-1  |
| MS | 1  | Iub-89  | ES-1-3-1-1  |
| MS | 13 | Iub-90  | ES-1-3-1-1  |
| MS | 8  | Iub-91  | ES-1-3-1-1  |
| MS | 1  | Iub-92  | ES-1-3-1-1  |
| MS | 13 | Iub-93  | ES-1-3-1-1  |
| MS | 8  | Iub-94  | ES-1-3-1-1  |
| MS | 1  | Iub-95  | ES-1-3-1-1  |
| MS | 13 | Iub-96  | ES-1-3-1-1  |

The second table (which is the first table when option "a" is not used) shows the resources allocation, module by module.

- **Sr** : the subrack containing the module resources
- **Mod** : the module number
- **S** : the state of the module MP: L=locked, 1=enabled, 0=disabled
- **GPB** : the board type of the module MP
- **nIub** : the number of lubLinks handled by the module MP. Having an equal number of lubLinks/Cells on each module gives a better spreading of the load.
- **CellGPB**: the number of UtranCells handled by the module MP
- **CellCC** : the number of UtranCells handled by the CC devices controlled by that module MP.
- **nCC** : the number of CC devices allocated to this RncModule.

Note: if the cell to CC device allocation has changed since the moshell session was started, the command "bor" needs to be run in order to refresh the moshell cache, otherwise the values in "CellCC" field could be wrong.

Cell repartition by rncModule:

| Sr  | Mod  | S | GPB   | nIub | CellGPB | CellCC | nCC |
|-----|------|---|-------|------|---------|--------|-----|
| MS  | 60   | 1 | GPB65 | 14   | 90      | 117    | 1   |
| MS  | 80   | 1 | GPB65 | 13   | 81      | 114    | 1   |
| MS  | 110  | 1 | GPB65 | 14   | 90      | 119    | 1   |
| MS  | 140  | 1 | GPB65 | 14   | 93      | 117    | 1   |
| ES1 | 21   | 1 | GPB65 | 14   | 99      | 118    | 1   |
| ES1 | 81   | 1 | GPB65 | 14   | 99      | 114    | 1   |
| ES1 | 151  | 1 | GPB65 | 14   | 111     | 119    | 1   |
| ES1 | 181  | 1 | GPB65 | 14   | 96      | 106    | 1   |
| ES1 | 211  | 1 | GPB65 | 14   | 105     | 103    | 1   |
| ES1 | 2121 | 1 | GPB75 | 13   | 102     | 114    | 1   |
| ES1 | 2122 | 1 | GPB75 | 13   | 87      | 114    | 1   |
| ES2 | 3021 | 1 | GPB75 | 14   | 93      | 140    | 1   |
| ES2 | 3022 | 1 | GPB75 | 14   | 99      | 0      | 0   |
| ES2 | 3081 | 1 | GPB75 | 14   | 90      | 140    | 1   |
| ES2 | 3082 | 1 | GPB75 | 14   | 93      | 0      | 0   |
| ES2 | 3121 | 1 | GPB75 | 15   | 99      | 140    | 1   |
| ES2 | 3122 | 1 | GPB75 | 14   | 96      | 0      | 0   |
| ES2 | 3151 | 1 | GPB75 | 14   | 99      | 114    | 1   |
| ES2 | 3152 | 1 | GPB75 | 14   | 96      | 113    | 1   |
| ES2 | 3181 | 1 | GPB75 | 14   | 102     | 120    | 1   |
| ES2 | 3182 | 1 | GPB75 | 14   | 102     | 99     | 1   |
| ES2 | 3211 | 1 | GPB75 | 14   | 93      | 87     | 1   |
| ES2 | 3212 | 1 | GPB75 | 14   | 93      | 0      | 0   |

The third table (which is the second table when option "a" is not used) is identical to the previous one but aggregated on Module Board level. In the case of GPB5/GPB6 it will give the same figures as the previous table but in the case of multicore boards (GPB75/EPB) it gives aggregated values for all RncModules of that board.

Cell repartition by Board:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

| Sr  | Slot | S | GPB   | nIub | CellGPB | CellCC | nCC |
|-----|------|---|-------|------|---------|--------|-----|
| MS  | 06   | 1 | GPB65 | 14   | 90      | 117    | 1   |
| MS  | 08   | 1 | GPB65 | 13   | 81      | 114    | 1   |
| MS  | 11   | 1 | GPB65 | 14   | 90      | 119    | 1   |
| MS  | 14   | 1 | GPB65 | 14   | 93      | 117    | 1   |
| ES1 | 02   | 1 | GPB65 | 14   | 99      | 118    | 1   |
| ES1 | 08   | 1 | GPB65 | 14   | 99      | 114    | 1   |
| ES1 | 12   | 1 | GPB75 | 26   | 189     | 228    | 2   |
| ES1 | 15   | 1 | GPB65 | 14   | 111     | 119    | 1   |
| ES1 | 18   | 1 | GPB65 | 14   | 96      | 106    | 1   |
| ES1 | 21   | 1 | GPB65 | 14   | 105     | 103    | 1   |
| ES2 | 02   | 1 | GPB75 | 28   | 192     | 140    | 1   |
| ES2 | 08   | 1 | GPB75 | 28   | 183     | 140    | 1   |
| ES2 | 12   | 1 | GPB75 | 29   | 195     | 140    | 1   |
| ES2 | 15   | 1 | GPB75 | 28   | 195     | 227    | 2   |
| ES2 | 18   | 1 | GPB75 | 28   | 204     | 219    | 2   |
| ES2 | 21   | 1 | GPB75 | 28   | 186     | 87     | 1   |

The fourth table (which is the third table when option "a" is not used) shows an aggregated view of the previous table, on Subrack level. This is interesting to see if any Subracks are more loaded than others.

- **Sr** : the Subrack identity
- **nMod** : the number of rncModules contained in the Subrack
- **nCC** : the number of CC devices contained in the Subrack
- **nIub** : the number of lubLinks handled by all module MPs of the Subrack
- **CellGPB** : the number of UtranCells handled by all module MPs of the Subrack
- **CellCC** : the number of UtranCells handled by all CC devices of the Subrack
- **avIub** : the average number of lubLinks per module MP in that subrack
- **avCell** : the average number of UtranCells per module MP in that subrack
- **avCellCC** : the average number of UtranCells per CC device in that subrack

Cell repartition by Subrack:

| Sr   | nMod | nCC | nIub | CellGPB | CellCC | avIub | avCell | avCellCC |
|------|------|-----|------|---------|--------|-------|--------|----------|
| MS   | 4    | 4   | 55   | 354     | 467    | 14    | 88     | 117      |
| ES1  | 7    | 7   | 96   | 699     | 788    | 14    | 100    | 113      |
| ES2  | 12   | 8   | 169  | 1155    | 953    | 14    | 96     | 119      |
| Tot: | 23   | 19  | 320  | 2208    | 2208   |       |        |          |

#### 4.2.16 resub <lubLink> [<VplTp>|<Subrack>] [<VplTp>]

Moving lub resources within or across Subracks (RNC >= P5).

There are three ways to run the command:

##### a) Respreading of lub resources within a subrack: resub <iublink(s)>

The **lkr** command shows the repartition of Cells/lubLinks across rncModules, Cc devices and Subracks. The performance is affected when the resources allocation is uneven. The Cells/lubLinks resources can be evenly reallocated by using the command **resub <iublink(s)>**. Example:

- **resub iublink=-** - Reallocate all iublink resources evenly in the node
- **ma iub\_es1 iublink subrackref subrack=es-1 ; resub iub\_es1** - Reallocate all iublink resources evenly in subrack ES-1

##### b) Moving lub resources to a different subrack: resub <iublink(s)> <subrack>



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The `lkra` command shows the list of `lubLinks` whose `AtmPort(s)` are located in a different Subrack than the `lubLink`. This causes higher ISL load. The `lubLink` can then be moved to a different Subrack with the command `resub <iublink(s)> <subrack>`. Example:

- `resub iublink=iub-10 subrack=es-1` - Move `lubLink` `iub-10` to subrack `ES-1`

### c) Moving lub resources to different AtmPort/Vp: `resub <iublink> <vp> [<vp>]`

When there is a need for more `Atm` bandwidth it may be necessary to move an `lub` to a different `Atm` port. This can be done with the command `resub <iublink> <vp> [<vp>]`. The second `vp` is optional and only needed for `lubLinks` that use two `AtmPorts` for redundancy.

- `resub iublink=3040 atmport=ms-27-2,vpltp=vp2` - Move the `lubLink` `3040` to `AtmPort=MS-27-2,VplTp=vp2`
- `resub iublink=3040 atmport=ms-27-2,vpltp=vp2 atmport=ms-26-2,vpltp=vp8` - Move the `lubLink` `3040` to `AtmPort=MS-27-2,Vpltp=vp2` and `AtmPort=MS-26-2,VplTp=vp8`

### Result:

- **a) and b)** In these cases, the command sets the attribute `preferredSubrackRef` (P5/P6) or `atmUserPlaneTermSubrackRef` (P7) to the required subrack. In the case of a), this is the current subrack, equivalent to running `setc <iublink(s)> preferredsubrackref`. In the case of b), this is the given subrack, equivalent to running `set <iublink(s)> preferredsubrackref <subrack>`
- **c)** In the case where one or two `VplTp(s)` have been specified, a command file is generated containing all the commands necessary for moving the `lubLink` to the new `VplTp(s)`. The command can be executed with the `run` command or can be converted to EMAS/MoTester format using the `u!` command. In that case, it can be executed with the `trun` command.

## 4.2.17 `ir[cdps] [<lubLink>] [<CM>]`

`lub` Redundancy operations for RNC in pool.

Arguments (both optional):

- the first argument is for specifying the `lubLink(s)` on which the command will apply. It can be a `MO` filter, a proxy ID, or a `MO` group. If no `lubLink` is specified then the command will apply to all. `lubLinks` using `ATM` transport will be silently ignored. Note: the `c` option will always ignore this argument.
- the second argument is for specifying the address of the other Cluster Member, which can be an ip address or the path to an `MO` dump. The other Cluster member can also be specified in the uservariable `cluster_members` in which case it is not necessary to specify this argument. The uservariable `cluster_members` shall contain the list of ip addresses or `MO` dump paths for all nodes of the cluster. It can be specified from the command line with `uv` command or saved in the `ipdatabase` file.

Options:

- `c`: Consistency check of the `lub` Redundancy configuration. Will generate a printout listing all the checks and their result. See printout description further down.
- `p`: Protect one or more `lubLinks`. Only `lubLinks` with `poolRedundancy` `PRIMARY` or `NON-REDUNDANT` will be considered by the command. The `poolRedundancy` attribute will be set to `PRIMARY` and a command file will be generated to define the secondary `lubLink(s)` and associated cell data in the other Cluster Member. The command file shall be executed in the other Cluster Member and the file path can be accessed from the variable `$irpcommandfile`.
- `s`: Synchronise the configuration data of one or more secondary `lubLinks`. Only `lubLinks` with `poolRedundancy` `SECONDARY` and `operationalState` `DISABLED` will be considered by the command. A command file will be generated to align the secondary `lub/Cell` data to its counterpart in primary side. The command file shall be executed in the same Cluster Member where the `irs` command was run, and the file path can be accessed from the variable `$irscommandfile`. Any secondary `lubLinks` without a primary counterpart will be saved in the variable `$irs_orphan_iubs` and can be removed with the command `ird $irs_orphan_iubs`.
- `d`: Delete one or more secondary `lubLinks`. Only `lubLinks` with `poolRedundancy` `SECONDARY` and `operationalState` `DISABLED` will be considered by the command. A command file will be generated to remove the secondary `lubLink(s)` and associated cell data. The command file shall be executed in the same Cluster Member where the `ird` command was run, and the file path can be accessed from the variable `$irdcommandfile`. The corresponding primary `lubLinks` can be set to non-redundant with the command `set $ird_iubs poolRedundancy 0`, to be run in the other Cluster Member.

Notes:

- when a secondary `UtranCell` is created by `irp` or `irs`, the previously existing `RemoteUtranCell` with that `cld` will be removed, and any cell relations terminating to the `RemoteUtranCell` will be redirected towards the newly secondary `UtranCell`.
- when a secondary `UtranCell` is deleted by `ird` or `irs`, a `RemoteUtranCell` will be created in its place in order to terminate

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

any cell relations that were pointing to the newly deleted UtranCell.

- secondary UtranCells created by `irp` or `irs` will use a dedicated Ura common for all cells of that secondary lubLink. The Ura MO used by the primary UtranCells will not be used as it is currently not supported by RNC to use the same UraIdentity in several members of the same cluster.
- any attributes that shall not be copied over from primary to secondary lub/Cells can be specified in the uservariable `ir_nosync`.
- the secondary lubLink created by `irp` will be created in `administrativeState=LOCKED` if `redundancySwitchoverWaitTime=-1` on the s-CM.

Example printout from `irc` command:

```
=====
Iub Redundancy Consistency Check for nodes: RNC01 (CM1) and RNC02 (CM2)
=====
1) Same SW level in all CMs: NOT OK
CurrentUpgradePackage: CXP9021776_R4FC40 (CM1) CXP9021776_R4FC39 (CM2)
-----
2) Same features operational in all CMs: NOT OK
RncFeature=RabCombination009 serviceState: 0 (CM1) 1 (CM2)
-----
3) All primary/non-redundant IubLinks/UtranCells have unique rbsId/cId within the cluster: NOT OK
cId=2205 UtranCell=cell11106-Iub-1523-6 (CM1) UtranCell=cell2205-Iub-1525-5 (CM2)
-----
4) All secondary IubLinks/UtranCells have unique rbsId/cId within the cluster: NOT OK
cId=2202 UtranCell=cell12202-Iub-1525-2 (CM1) UtranCell=cell2201-Iub-1525-1 (CM1)
-----
5) No Utran/Coverage Relations between two cells with same cId: NOT OK
cId=1201 UtranCell=cell12204-Iub-1525-4,UtranRelation=Rem-Inter-Iub-1526-1 (CM1)
cId=1201 UtranCell=cell11201-Iub-1526-1,UtranRelation=Rem-Intra-Iub-1525-4 (CM1)
cId=2202 UtranCell=cell12201-Iub-1525-1,UtranRelation=Intra-Softer-cell2202-Iub-1525-2 (CM1)
cId=2202 UtranCell=cell12202-Iub-1525-2,UtranRelation=Intra-Softer-cell2201-Iub-1525-1 (CM1)
-----
6) No RemoteUtranCells with same cId as a local UtranCell or another RemoteUtranCell within each CM:
cId=2202 UtranCell=cell12202-Iub-1525-2 UtranCell=cell12201-Iub-1525-1 (CM1)
cId=2205 UtranCell=cell11106-Iub-1523-6 UtranCell=cell12205-Iub-1525-5 (CM1)
cId=1201 UtranCell=cell11201-Iub-1526-1 UtranCell=cell12204-Iub-1525-4 (CM1)
-----
7) No unlocked disabled secondary IubLinks when redundancySwitchoverWaitTime=-1: NOT OK
IubLink=Iub-1523 (CM1)
IubLink=Iub-1525 (CM2)
-----
```

#### 4.2.18 `tg[r][c][d]`

Print Resource Object information for all MOs in LmCell (RNC only).

Command Syntax:

```
tg[c][r] [<moFilter>|<mogroup>|<fro>][:<actorChildren>] [<rrt-cmd>] [|<unix-cmds>]
tg[d]      [<moFilter>|<mogroup>|<fro>][:<actorChildren>] [<cell-parameter(s)>]
```

Purpose:

- To print the relation MO $\longleftrightarrow$ FRO $\longleftrightarrow$ ACTOR $\longleftrightarrow$ CCDEVICE for **IubLink** and **UtranCell** MOs.
- To send RRT commands to Cell/lub actors or their children in the actor tree.

Arguments: By specifying the first argument, it is possible to filter the Cell/lubs matching a specific Frold or MOId.

By specifying a second argument consisting of an RRT command or a list of cell parameters, it is possible to run an RRT-command or display certain cell parameters on all the actors that are matched by the first argument.

The list of available RRT commands can be seen by typing:

- `lhsh 001400 ? rrt`

The list of available cell parameters can be seen by typing:

- `lhsh 001400 rrt-CXC132xxxx_Ryyyy 1/1/1/1/2/1.1 info` (for cell parameters)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `lhsh 001400 rrt-CXC132xxxx_Ryyyy 1/1/1/1/7/1.1/4/1 info` (for nbap common parameters)

The printout can be piped into unix commands, like `grep` and `sort`.

Options:

- `r` : to refresh the MO/FRO/ACTOR data. Otherwise this data is reused within the moshell session and from session to session, using a cache on the workstation disk. A `tg refresh (tgr)` needs to be done after a node upgrade or if some **lubLinks** or **UtranCells** have been added/removed/remoduled. Otherwise you may get error messages such as `unknown command rrt-CXC1328831_Rxxx` (eg, the `rrt` LM has changed after an upgrade).
- `c` : print the following extra extra fields:
  - Common Channel Device (**ccDevice**): shows which SPM is used to handle the common channels of this cell
  - ccDevice Module (**ccMod**): shows if a ccDevice is running on an SPM that doesn't belong to the same module as the Cell/Iub. This can be fixed by locking/unlocking the cell. It is always best to make sure that all cells are using a ccDevice located in their own module so that the load will be spread equitably on all ccDevices.

Note that ccDevice data is not kept in the cache, only MO/fRO/Actor relation is kept in the cache.

- `d` : to print certain cell-parameters for all cells matching the first argument.

Examples:

1. `tg` - to view the fRO/Actor data for each **UtranCell** and **lubLink**
2. `tg cell=302` - to view the fro/actor data for all UtranCell MOs matching regex `cell=302`
3. `tg cellmod1` - to view the fro/actor data for all MOs belonging to the MO group `cellmod1` (use **ma** command to make a MO group)
4. `tg 67` - to view the mo/actor data for the MOs that have `fro=67`
5. `tgr` - to refresh the fro/actor data
6. `tg c` - to view all cells/iubs and their respective fro/actor and ccDevice (Note: ccDevice data is not cached so if it is not necessary to do `tgr` to refresh ccDevice data)
7. `tg iublink=3.*1$ info` - to run the `rrt info` command on all actors whose MO matches `iublink=3.*1$`
8. `tg c | grep 0019` to see all cells that are on ccdevice of board 001900
9. `tg cell getattr cellData` - to send the `rrt` command `getattr cellData` on all actors whose MO matches `cell`
10. `tg iubmod3 state` - to run the `rrt` command `state` on all MOs belonging to the MO group `iubmod3` (use **ma** command to make a MO group)
11. `tg d cell ulinterference celldata:errorstatus cellRoState` - to view the cell parameters `ulinterference, celldata:errorstatus, etc.` on all cells (P3/P4)
12. `tg d cell:/8/3 cellLoadMonitor:totAseDl cellLoadMonitor:totAseUl cellLoadMonitor:currDlCode`  
- to view the admission tree usage on all cells (P5 and after)
13. `tg d cell ulinterference celldata:errorstatus celldata:spconfigflag cellRoState`  
- to view the cell parameters `ulinterference, celldata:errorstatus, etc.` on all cells.
14. `tg d cell cellLoadMonitor:totAseDl cellLoadMonitor:totAseUl cellLoadMonitor:currDlCode`  
- to view the admission tree usage on all cells.
15. `tg d cell ulinterference celldata:spblocked celldata:spconfigflag cellRoState celldeleted cellTraceActivated celldata:errorstatus`
16. `tg d iub:/4/1 activeStatus standbyStatus currentCause currentAvailabilityStatus rncRbsLinkLossOfRedundancy rncRbsLinkDown rncRbsDeactivated`  
- to view those attributes on all the NsapCommon actors

Output examples:

- `tg c`

| MOD | MFRO | ModMP  | UtranCellId | IubLinkId | CfRO | IfRO | CellActor     | IubActor        | CcDev   |
|-----|------|--------|-------------|-----------|------|------|---------------|-----------------|---------|
| 1   | 0    | 001400 | 90121       | 9012      | 5    | 1    | 1/1/1/1/2/1.3 | 1/1/1/1/7/1.1/4 | 0019SP2 |

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```

1 0 001400 90122          9012 4      1 1/1/1/1/2/1.2 1/1/1/1/7/1.1/4 0020SP0
1 0 001400 90123          9012 3      1 1/1/1/1/2/1.1 1/1/1/1/7/1.1/4 0020SP0

```

- `tgdcell=9012 ulinterference celldata:spblocked celldata:spconfigflag cellRoState  
celldeleted cellTraceActivated celldata:errorstatus`

| MO              | ulinterference | spblocked | spconfigflag | cellrostate | celldeleted | errorstat                | celltraceactivated |
|-----------------|----------------|-----------|--------------|-------------|-------------|--------------------------|--------------------|
| UtranCell=90121 | -106           | 0         | 0            | 2           | 0           | 0                        | 0                  |
| UtranCell=90122 | -100           | 0         | 0            | 1           | 0           | 16002:CellNoDrhResources | 0                  |
| UtranCell=90123 | -100           | 0         | 0            | 1           | 0           | 16002:CellNoDrhResources | 0                  |

## Description:

- `spblocked=1` means that the CCdevice has crashed and Rnh got the signal `clearResourceReq`.
- `spconfigflag` says if the cell has allocated SP resources or not. Should be 0 if `spblocked=1`.
- `cellrostate` is defined in the header file `rllibRncConst.h`: `roStateNOK=1`, `roStateOK=2`, `roStateDepNOK=3`
- `celldeleted` means that the cell has been deleted by the operator, i.e. FRO has sent a `deleteInd`.
- `celltraceactivated` says if the feature "Selective Cell Tracing" is active or not. See CR WRNac20241
- `errorstatus` is different to 0 if there is a fault in the cell unlock procedure. Range 16000-16014 is specified in the file `rllibEventNr.h`. `Cellrostate` shall be equal to 1 if the `errorstatus` is different to 0.
- `tgdiub:/4/1 activeStatus standbyStatus currentCause  
currentAvailabilityStatus rncRbsLinkLossOfRedundancy  
rncRbsLinkDown rncRbsDeactivated`

to view those attributes on all the NbpCommon actors

| MO           | activestatus | standbystatus | currentcause | currentavailabilitystatus | rncrbslinkdown | rncrbsdeactivated |
|--------------|--------------|---------------|--------------|---------------------------|----------------|-------------------|
| IubLink=1001 | 1            | 1             | 2            | 2                         | 0              | 0                 |
| IubLink=1002 | 1            | 1             | 2            | 2                         | 0              | 0                 |

## Description:

- `activeStatus` and `standbyStatus` are boolean (0 or 1). It's the FRO's way to say if it's ok or not to attach to USAAL. `UserPlaneCepId` for active and standby shall be different to -1 if active/standby status equals 1.
- `currentCause` is sent by FRO in the signals `opStateChdInd` and `setAttribInd` and is defined by the dataclass `CpxUsaalEfi_OpStateChangeCauseD`. Valid values are:
  - `CELLO_USAALEFI_SERVER_RESTARTED == 1`
  - `CELLO_USAALEFI_OTHER == 2`
  - `CELLO_USAALEFI_SERVER_MOVED == 3` (introduced in the feature Moveable CEP in P5MD)
- `currentAvailabilityStatus` comes together with `opStateChdInd` and is about "regular" state propagation, i.e. if bit 5 is set to NBAP RO Dependency Failed.
- `rncRbsLinkLossOfRedundancy` & `rncRbsLinkDown` are internal flags that say if the events `linkDown` and `linkLossOfRedundancy` are active or not. If `rncRbsLinkDown=1` then `currentAvailabilityStatus` will be "Dependency Failed" and `linkUsedForTraffic` will be equal to -1.
- `linkUsedForTraffic` says which `iublink` we are using at the moment. Active == 0, Standby == 1. If `linkUsedForTraffic` equals 1 (standby) then `rncRbsLinkLossOfRedundancy` is also equal to 1.
- `rncRbsDeactivated` says if the event `linkDeactivated` is active or not. All 3 events are defined in header file `rllibEventNr.h`.
- `tgdcell:/8/3 cellLoadMonitor:totAseDl cellLoadMonitor:totAseUl cellLoadMonitor:currDlCode`

to view those attributes on all the NbpCommon actors

| MO              | totasedl | totaseul | currdlcode |
|-----------------|----------|----------|------------|
| UtranCell=30101 | 320      | 160      | 3750       |
| UtranCell=30102 | 0        | 0        | 3593       |
| UtranCell=30103 | 0        | 0        | 3593       |
| UtranCell=30104 | 1940     | 969      | 4218       |
| UtranCell=30105 | 2064     | 1568     | 4375       |
| UtranCell=30106 | 991      | 911      | 3984       |

## Description:

- `totAseDL` is the total amount of DL ASE that is currently in use in the cell. The value is scaled by a factor 100 to get a resolution of 0.01 ASE, i.e. `totalAseDL = 1280` means 12.80 ASE DL load in the cell.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- totAseUI is the same as totAseDI but for UL ASE. DL and UL ASE values are controlled by UEH (by signals admissionRequest, admissionDecreaseLoadInd, admissionAseUIIncreaseInd and admissionAseUIDecreaseInd).
- currDICode reflects the last reported value of DL code tree load utilization in the cell (reported from the RnhCode block by signal codeTreeLoadInd). The value is scaled by factor 100 to get a resolution of 0.01

#### 4.2.19 ueregprint/uer[d][t][i][s][p][v] [-m <mod>|-i <imsi>|-u <ueref>|-n <maxUes>|-c <utrancell>|-r <iublink>] [<attribute-filter>[=<value>]]all]

Print UE registry or UE context data (serving or drifting) for all active calls (RNC only).

The command `ueregprint` is a moshell alias that runs the `coli` command "ueregprint" on all RNC RANAP boards. It allows all the same arguments as the regular `ueregprint` command but does not require to specify the board on which it should run. This is especially useful for EvoC node where the UE registry is running in all the blades.

Example:

- `ueregprint sum`
- `ueregprint all`

The command `uer` is a moshell command that prints the UE contexts in each module MP in order to show the details of call.

The following COLI commands are used: `ueregprint` (on C2/Ranap MP), `drh_trbr rab` (on PDR module MPs), and `printUeCtxt` (on module MP). No RRT commands are used.

Switches:

- **s**: for showing the traffic summary tables at the end of the printout.  
When no filtering options are used (e.g. `uers` command is run on its own) then only the summary tables are shown, not the individual calls.  
When filtering options are used but the **s** switch is not given (e.g. `uer [-option <filter>]`), then the summary tables are not shown, only the individual calls.
- **d**: for printing drift UEs.
- **i**: for printing UEs that have a IMSI only.
- **t**: for printing UEs that have a TMSI or IMEI only.
- **p**: for skipping the PDR device check, faster.
- **v**: for printing UE context attributes vertically, only applies when used with the `attribute-filter`

Filtering Options:

- **-m <mod>**: to show calls belonging to certain module(s) only. Eg: `-m 1,8 ==>` calls in modules 1 and 8 are printed.
- **-i <imsi>**: to show calls whose IMSI match a specific filter only. Eg: `-i 3014235`.
- **-u <ueref>**: to show the calls related to a specific UE ref. Eg: `-u 4728`.
- **-n <maxUes>**: to show a maximum number of calls only. Eg: `-n 50 ==>` a maximum of 50 calls are printed.
- **-c <utrancell>**: to show the calls that have a radio link or common channel in certain utrancells.
- **-r <iublink>** : to show the calls that have a radio link or common channel in certain iublins.

Note regarding the options `-c/-r`:

- on RNC  $\geq$  P6, a regular expression filter can be used, e.g. `uer -c 30.*a`
- on RNC  $\leq$  P5, the exact utrancell or iublink must be given, e.g. `uer -r iub_10`

The attribute-filter can be:

- empty. Some default tables will be printed, see below.
- "all". All UeContext data for the UEs matching the first argument will be printed
- a regular expression matching one or more UeContext attributes. Only the attributes matching the string will be printed.
- an attribute name followed by "=" and a value (e.g. `sccpConnId=6`). Only the UEs that have an attribute matching that value will be printed.

If the attribute-filter is empty, then three tables are printed:

- The first table contains for each UeContext:
  - MOD: the module handling the call

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- IMSI: the IMSI of the UE
  - CTXT: the UeContext (UeRef) id of the UE
  - SRNC: the Serving RNC. Read from the variable sRncId.
  - CommCh: the rbsid and cell id (cid) of the serving cell providing the Common Channels for this UE. Read from the variable commonResData.cld
  - RL1-4: the rbsid and cell id (cid) of the serving cells providing the radio links for this UE (up to 4 radio links per UE). Read from the variables radioLink[0-3].cld
  - DCdev: the DC device handling the call. Read from the variable drhRcSpId.
  - PDRdev: the PDR device and PacketDataRouter frold handling the call. Read from the command `lh mod drh_trbr rab.`
  - CoreNetId: the Type of Core Network connecting the call: CS or PS. Read from the variables isCNConnected.\*Circuit and isCNConnected.\*Packet. Also shows the Id of the Core Network used for the call (mcc/mnc/cnld). Read from the variables globalCnldPs and globalCnldCs.
  - AGE: the duration of the call. Read from `lh ranapmp ueregprint all`
  - UERC: the UeRc used by the call. Read from the variable connType.
- the second table shows the current number of Ue Contexts associated to each UeRc type in each module.
  - the third table shows the same as the second table but for the whole node.

Examples (with empty attribute-filter):

- `uer` - print UeContext data for all UEs in the node
- `uer -n 25` - print UeContext data for 25 UEs only (randomly selected across the node and proportionately spread across all the modules)
- `uer -i 0001` - print UeContext data for all UEs whose IMSI match 0001.
- `uer -m 8` - print UeContext data for all UEs in module 8.
- `uer -m 8,9,10` - print UeContext data for all UEs in modules 8, 9 and 10.
- `uer -r iub-17` - print UeContext data for all UEs that have a radio link or common channel in a cell of lubLink "lub-17"
- `uer -c iub-17-1` - print UeContext data for all UEs that have a radio link or common channel in the UtranCell "lub-17-1"
- `uer -r iub.*1$` - print UeContext data for all UEs that have a radio link or common channel in the lubLinks whose name matches `iub.*1$` (works only on RNC P6 and above)

See output example below:

```
=====
MOD      IMSI/TMSI  CTXT   CommCh   RL1      RL2      RL3   DCdev   PDRdev  CoreNetId  AGE  UERC + ESTAB_CAUSE
=====
1  301020430130001  1826      746:2907  20:1174      0923sp3
1  301001920070001  287        68:1178      0022sp2  0019sp0:1  P 24099
1  301000310030000  309          298:1875      0021sp3  0019sp0:0  CP 24099 00:00:52 5 Interact. PS (64/64) (2:origInteractive)
1  301010330100000  5778 680:2951      0923sp3  0019sp0:0  P 24099 00:01:28 9 Conv. CS speech 12.2 + Interact. PS (0/0) (2:origInteractive)
1  301001630000000  5565      582:2848  71:1205      0023sp2  0019sp0:0  CP 24099 00:01:59 4 Interact. PS (RACH/FACH) (2:origInteractive)
1  301001630000000  5565      582:2848  71:1205      0023sp2  0019sp0:0  C 24099 00:00:14 2 Conv. CS speech 12.2 (0:origConversational)
...<cut>...

=====
UeRc  M1  M8  M13  userLabel
=====
1  14  13  4  Standalone RRC on DCH
2  75  55  55  Speech
3  18  30  22  64kbps CS data, fixed rate
4  13  9  17  Packet RACH/FACH
5  5  1  2  PACKET 64/64
7  1  0  0  Packet 64/384
9  0  0  1  Speech + Packet 0kbps
14  0  0  1  CS data 64kbps + Packet 8/8
16  1  0  0  PS Interactive 384/HS - HS-DSCH
18  1  0  0  Packet 128/128
Tot: 128 108 102

=====
Cause  M1  M8  M13  EstablishmentCause
=====
0  65  46  57  origConversational
2  22  13  22  origInteractive
5  30  40  19  termConversational
12  11  9  4  registration
Tot: 128 108 102

=====
UeRc  Total  %  userLabel
=====
1  31  9.2  Standalone RRC on DCH
2  185  54.7  Speech
3  70  20.7  64kbps CS data, fixed rate
4  39  11.5  Packet RACH/FACH
5  8  2.4  PACKET 64/64
7  1  0.3  Packet 64/384
9  1  0.3  Speech + Packet 0kbps
```



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```

14      1      0.3  CS data 64kbps + Packet 8/8
16      1      0.3  PS Interactive 384/HS - HS-DSCH
18      1      0.3  Packet 128/128
=====
Tot:    338    100

```

```

=====
Cause  Total      %  EstablishmentCause
=====
0      168    49.7  origConversational
2       57    16.9  origInteractive
5       89    26.3  termConversational
12      24     7.1  registration
=====
Tot:    338    100

```

Examples (with attribute-filter not empty):

- `uer -i 301001800040001 all -print all UeContext data for the UE with IMSI 301001800040001`

```

=====
MOD      IMSI  ATTRIBUTES
=====
8  301001800040001  $ UehUexCtxtD[1022]=0x4FE0D058
8  301001800040001  [1022].isActive() = 1
8  301001800040001  [1022].isCNConnected(uehCNidCircuit) = 0
8  301001800040001  [1022].isCNConnected(uehCNidPacket) = 1
8  301001800040001  [1022].sRncId = 301
8  301001800040001  [1022].softHoDone = 1
8  301001800040001  [1022].isPmRecordingActive = 0
8  301001800040001  [1022].recordingProt = 1
8  301001800040001  [1022].diPcMethod = 3
8  301001800040001  [1022].diRefPwrVal = -165
8  301001800040001  [1022].supportOfGsm = 1
8  301001800040001  [1022].tmpRanapConnId = -1
8  301001800040001  [1022].recordingMeas = 0
8  301001800040001  [1022].measBERrequested = 0
8  301001800040001  [1022].measBLERrequested = 0
etc.....

```

- `uer . aal2.*cepId$|softho|\rcIndex$ - for all UEs, print UeContext attributes that match the regexp "aal2.*cepId$|softho|rcIndex$"`

```

=====
MOD      IMSI  ATTRIBUTES
=====
1  301001800060003  softHoDone=1  rcIndex=4
1  301001701000000  softHoDone=0  rcIndex=2  aal2Arr[0].cepId=477  aal2Arr[1].cepId=494
1  301001810040000  softHoDone=1  rcIndex=4
1  301001700000001  softHoDone=1  rcIndex=2  aal2Arr[0].cepId=76  aal2Arr[1].cepId=77  aal2Arr[2].cepId=141  aal2Arr[3].cepId=142
1  301001720000003  softHoDone=1  rcIndex=2  aal2Arr[0].cepId=507  aal2Arr[1].cepId=508  aal2Arr[2].cepId=332  aal2Arr[3].cepId=333
1  301001720110001  softHoDone=1  rcIndex=2  aal2Arr[0].cepId=424  aal2Arr[1].cepId=425
1  301001810000003  softHoDone=1  rcIndex=2  aal2Arr[0].cepId=476  aal2Arr[1].cepId=482  aal2Arr[2].cepId=166  aal2Arr[3].cepId=167
1  301001711000002  softHoDone=1  rcIndex=2  aal2Arr[0].cepId=52  aal2Arr[1].cepId=57  aal2Arr[2].cepId=170  aal2Arr[3].cepId=171

```

- `uer -i 001 sccpConnId=6 - for all UEs whose IMSI match "001", print the UeContact attributes that match sccpConnId=6`

Note: to abort the printout, do `Ctrl-z`, then: `touch <stopfile> ; fg`

The path to the *stopfile* can be found in the window title.

#### 4.2.20 `ced[h][p][s][g][r] [-m <module(s)>|-c <utrancell>|-r <iublink>] [ | <unix-cmds>]`

Print consumption of cell resources and rbs hw, cell supported features, cell coordinates (RNC only).

Options:

- `ced` : consumption of air interface resources for each UtranCell. Read from the `coli` command "celldata" on module MPs.
- `cedh` : CE consumption (Channel Element) and number of RadioLinks/UEs for each lubLink. Read from the `coli` command "hwm" on module MPs.
- `cedhp`: same as above but the CE usage is expressed in percentage.
- `cedg` : state and geographical coordinates of each UtranCell. Read from MO data on UtranCell and children.
- `ceds` : state and supported features of each UtranCell. Read from MO data on UtranCell and children.

The `r` option can be used together with any of the above, in order to clear the moshell cache and fetch latest values from the node.

Filters:

- `-m <module(s)>`: only fetch/parse the data for certain RncModules, e.g. `-m 8` or `-m 1,2,3`
- `-c <utrancell>`: only fetch/parse the data for specific UtranCells, e.g. `-c cell1304A` (not case sensitive)
- `-r <iublink>` : only fetch/parse the data for specific lubLinks, eg. `-r iub_304` (not case sensitive)

The output can be filtered by piping through some unix commands, eg. `grep` or `sort`.

Examples:

- `ced | grep 0019sp0` : print all cells configured on CC device 001900/sp0



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `ced | sort -k 5` : print cell data sorted on the fifth field
- `cedh -m 8,9,10` : print CE usage for all sites in modules 8, 9, 10.
- `ced -r iub_304` : print cell data for all cells belonging to `IubLink=Iub_304`
- `cedh -c cell1304a` : print CE usage for the site connected to `UtranCell=cell1304A`
- `cedg -c cell140` : print state and geographical coordinates for all cells matching "cell140"
- `ceds -r iub_56` : show state and supported features for all cells connected to lubs matching "iub\_56"

Printout format:

```
> ced
-----
MOD   CELL  cid  fro  ro  PwrDl/Adm  dlCode  ulInt  sf8d  sf8u  sf16d  sf16u  sf32d  sf4u  dlASE  ulASE  CPMcnt  HScnt  EULs  EULns  Eul2  Spch  Fdcph  Crn  Hrn  Ern  CC_SP
-----
1 Iub-1-1 1031  0  0  7.0% 75%   6.2% -111.8  0/8  0/8  0/16  0/16  0/32  0/4  0/500  0/500  0/15  0/10  1/32  0/100  0/4  0  0/0  0  0  0  1:0019spl
1 Iub-1-2 1032  1  2  7.0% 75%   6.2% -111.8  0/8  0/8  0/16  0/16  0/32  0/4  0/500  0/500  0/15  0/10  4/32  0/100  0/4  0  0/0  0  0  0  1:0021spl
1 Iub-1-3 1033  2  3  7.0% 75%   22.9% -111.8  1/8  0/8  0/16  1/16  0/32  0/4  39/500  8/500  0/15  0/10  3/32  0/100  0/4  0  0/0  0  0  0  1:0019spl
1 Iub-2-1 1037  6  1  7.0% 75%   24.6% -111.8  1/8  0/8  0/16  2/16  1/32  0/4  53/500  20/500  0/15  0/10  0/32  0/100  0/4  0  0/0  0  0  0  1:0019spl
1 Iub-2-2 1038  7  4  7.0% 75%   10.8% -111.8  0/8  0/8  0/16  2/16  1/32  0/4  12/500  19/500  0/15  1/10  0/32  0/100  0/4  0  0/0  0  0  0  8:0023spl(!)
```

The fields in the "ced" printout correspond to the following variables in the "celldata" printout on Module MP:

- MOD: RNC Module
- cid: cld
- fro: cellFrold (facade resource object, a unique id in the node)
- ro: cell RO (resource object, a unique id in the module. Same as capsule index.)
- PwrDl/PwrAdm: Filtered DL Power / pwrAdm
- dlCode: DL Code Allocation Level
- ulInt: UL Interference
- sf8dl: SF8 RL DL Count / sf8Adm
- sf8ul: SF8 RL UL Count / sf8AdmUI
- sf16dl: SF16 RL DL Count / sf16Adm
- sf16ul: SF16 RL UL Count / sf16AdmUI
- sf32dl: SF32 RL DL Count / sf32Adm
- sf4ul: SF4 RL UL Count / sf4AdmUI
- dlASE: Total ASE DL / aseDIAdm
- ulASE: Total ASE UL / aseUIAdm
- HScnt: HSDPA UE Count / hsdpaUsersAdm
- EULs: EUL UE Count serving / eulServingCellUsersAdm
- EULns: EUL UE Count non-serving / eulNonServingCellUsersAdm
- EUL2ms: EUL2ms UE Count serving / eulServingCellUsersAdmTti2
- Spch: Speech Only UE Count
- CPMcnt: CPM RL Count / compModeAdm
- Crn: Crnti allocation count
- Hrn: Hrnti allocation count
- Ern: Ernti allocation count
- Fdcph: This columns is read from the command "admtimeposdata". The first number corresponds to the number of opened codes and the second number is the number of time positions.
- CC\_SP: The RncModule and SPM of the CC device handling the Cell. If the RncModule controlling the CC device is different from the one controlling the cell, an exclamation mark (!) is printed. This used to be an issue in older RNC SW (PLM info 510), but is now fixed with TR HN89801, refer to PLM info 749.

```
> cedh
-----
MOD GRP  IUBLINK  fro  ro  nCell  usedCEdl  usedCEul  gHoCEdl  gHoCEul  OtherCEdl  OtherCEul  dlRL  ulRL  nrUE  leakingCellFroIds
-----
1  0  b0157MDN  0  0  3  34/64  98/256  30/64  158/256  29/63  155/253  46  46  46
1  4  b0423MDN  5  2  2  2/64  6/128  62/64  122/128  61/63  120/126  3  3  3
1  8  b0407MDN  8  4  1  8/64  42/256  56/64  214/256  55/63  211/253  15  15  15
1  10 b0106MDN  9  5  3  40/96  125/256  56/96  131/256  55/95  128/253  48  48  49  147
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

&gt; cedhp

| MOD | GRP | IUBLINK  | fro | ro | nCell | usedCEdl | usedCEul | gHoCEdl | gHoCEul | OtherCEdl | OtherCEul | dIRL | uIRL | nrUE | leakingCellFrolds |
|-----|-----|----------|-----|----|-------|----------|----------|---------|---------|-----------|-----------|------|------|------|-------------------|
| 1   | 0   | b0157MDN | 0   | 0  | 3     | 53.1%    | 38.3%    | 46.9%   | 61.7%   | 46.0%     | 61.3%     | 46   | 46   | 46   |                   |
| 1   | 4   | b0423MDN | 5   | 2  | 2     | 3.1%     | 4.7%     | 96.9%   | 95.3%   | 96.8%     | 95.2%     | 3    | 3    | 3    |                   |
| 1   | 8   | b0407MDN | 8   | 4  | 1     | 12.5%    | 16.4%    | 87.5%   | 83.6%   | 87.3%     | 83.4%     | 15   | 15   | 15   |                   |
| 1   | 10  | b0106MDN | 9   | 5  | 3     | 41.7%    | 48.8%    | 58.3%   | 51.2%   | 57.9%     | 50.6%     | 48   | 48   | 49   | 147               |

The fields in the "cedh/cedhp" printout correspond to the following variables in the "hwm print grp" printout on Module MP:

- MOD: RNC Module
- GRP: Cell Group (one to four per lubLink)
- IUBLINK: the MO id of the lubLink
- fro: the frold of the lubLink (facade resource object id, a unique id in the node)
- ro: the RO Id of the lubLink (resource object id, unique in the module. Same as capsule index).
- nCell: the number of UtranCells in the Cell Group
- usedCEdl/usedCEul: Consumed Credit / Capacity Credit . The number of used CE out of the total amount of available CE. The total amount of available CE is calculated in each RBS, based on the HW capability and the licensed capacity.
- gHoCEdl/gHoCEul: Guaranteed, HO (Channel Elements availability for Guaranteed, Handover traffic)
- OthrCEdl/OthrCEul: Other (Channel Elements availability for Guaranteed, Other traffic)
- dIRL, uIRL: number of Radio Links used in the RBS. The two values (downlink and uplink) should always be equal.
- nrUE: number of UEs in the RBS. This should be equal to the number of Radio Links.
- leakingCellFrolds: the frold of Cells where dIRL, uIRL, nrUE are not equal. This indicates that some resource has not been released properly in the cell. To release the resources in a leaking Cell, the corresponding lubLink MO must be locked and unlocked.
- leakingCellFrolds: the frold of Cells where the following condition is not met:
  - until W10: dIRL=uIRL=nrUE
  - from W11: nrUE=nrDIRL and and nrUIRL >= nrDIRL and nrDIRL >= nrDchUIRL and nrUIRL not more than double nrDIRL

This indicates that some resource has not been released properly in the cell. To release the resources in a leaking Cell, the corresponding lubLink MO must be locked and unlocked.

&gt; cedg

| Mod | UtranCell | CFRPHEMU | Antenna and Cell Coordinates, feed into <a href="http://maps.google.com/maps?q=">http://maps.google.com/maps?q=</a> |                    |                    |                    |                    |                    |  |  |  |  |  |  |  |
|-----|-----------|----------|---------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--|--|--|--|--|--|--|
| 21  | CTU20847  | 111111-- | 41.2383,-73.1937                                                                                                    | 41.2383,-73.193665 | 41.2424,-73.140621 | 41.2157,-73.149569 | 41.1996,-73.179159 | 41.2016,-73.215508 |  |  |  |  |  |  |  |
| 21  | CTU20848  | 111111-- | 41.2383,-73.1937                                                                                                    | 41.2383,-73.193665 | 41.2016,-73.215508 | 41.2208,-73.241665 | 41.2482,-73.245378 | 41.2710,-73.224885 |  |  |  |  |  |  |  |
| 21  | CTU20849  | 111111-- | 41.2383,-73.1937                                                                                                    | 41.2383,-73.193665 | 41.2710,-73.224885 | 41.2785,-73.189759 | 41.2672,-73.156478 | 41.2424,-73.140621 |  |  |  |  |  |  |  |
| 21  | CTV20841  | 111111-- | 41.2383,-73.1937                                                                                                    | 41.2383,-73.193665 | 41.2424,-73.140621 | 41.2157,-73.149569 | 41.1996,-73.179159 | 41.2016,-73.215508 |  |  |  |  |  |  |  |

The fields in "cedg" printout correspond to:

- MOD: RNC Module
- CFRPHEMU: status of UtranCell, Fach, Rach, Pch, HsdSch, Eul, MbmsCch, EulFach (L=locked, S=ShuttingDown, T=tpsPowerLocked, 1=unlocked&enabled, 0=unlocked&disabled)
- first coordinates: value of attribute UtranCell::antennaPosition
- all following coordinates: value of attribute UtranCell::utranCellPosition

The coordinates are expressed in degree of latitude and longitude and can be fed in google maps to see the location of the cell, eg: <http://maps.google.com/maps?q=41.2383,-73.1937>

&gt; ceds

| Mod  | UtranCell  | CFRPHEMU | ABCDEFGHIJKLMNQRST    | cpc | dBMC | edchT2 | enhDrx | enhL2 | eulDch | eulTd | fdpCh | hsAqm | hsFach | impL2 | lBHo | mC | mCMimo | q64 | q64Mimo | eulMC | hsThp | hs3MC | db3MC |
|------|------------|----------|-----------------------|-----|------|--------|--------|-------|--------|-------|-------|-------|--------|-------|------|----|--------|-----|---------|-------|-------|-------|-------|
| 1111 | cell117711 | 111111-L | 001000000001100000100 | 0   | 0    | 1      | 0      | 0     | 0      | 0     | 0     | 0     | 0      | 0     | 1    | 1  | 0      | 0   | 0       | 0     | 1     | 0     | 0     |
| 1111 | cell117712 | 111111-- | 001000000001100010100 | 0   | 0    | 1      | 0      | 0     | 0      | 0     | 0     | 0     | 0      | 0     | 1    | 1  | 0      | 0   | 0       | 1     | 0     | 0     | 0     |
| 1111 | cell117713 | 111111-- | 001000000001100010100 | 0   | 0    | 1      | 0      | 0     | 0      | 0     | 0     | 0     | 0      | 0     | 1    | 1  | 0      | 0   | 0       | 1     | 0     | 0     | 0     |
| 1111 | cell117714 | T00000-- | 001000000001100000100 | 0   | 0    | 1      | 0      | 0     | 0      | 0     | 0     | 0     | 0      | 0     | 1    | 1  | 0      | 0   | 0       | 0     | 1     | 0     | 0     |
| 1111 | cell117715 | T00000-- | 001000000001100010100 | 0   | 0    | 1      | 0      | 0     | 0      | 0     | 0     | 0     | 0      | 0     | 1    | 1  | 0      | 0   | 0       | 1     | 0     | 0     | 0     |

The fields in "ceds" printout correspond to the following attributes:

- MOD: RNC Module
- CFRPHEMU: status of UtranCell, Fach, Rach, Pch, HsdSch, Eul, MbmsCch, EulFach (L=locked, S=ShuttingDown, T=tpsPowerLocked, 1=unlocked&enabled, 0=unlocked&disabled)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- A=cpc: UtranCell::cpcSupport
- B=dBMC: MultiCarrier::dualBandMultiCarrierSupport
- C=edchT2: Eul::edchTti2Support
- D=enhDrx: Hsdsch::enhUeDrxSupport
- E=enhL2: Hsdsch::enhancedL2Support
- F=eulDch: Eul::eulDchBalancingSupport
- G=eulTd: Eul::eulTdSchedulingSupport
- H=fdpch: UtranCell::fdpchSupport
- I=hsAqm: Hsdsch::hsAqmCongCtrlSupport
- J=hsFach: Hsdsch::hsFachSupport
- K=impL2: Eul::improvedL2Support
- L=IBHo: UtranCell::loadBasedHoSupport
- M=mC: MultiCarrier::multiCarrierSupport
- N=mCMimo: MultiCarrier::multiCarrierMimoSupport
- O=q64: Hsdsch::qam64Support
- P=q64Mimo: Hsdsch::qam64MimoSupport
- Q=eulMC: MultiCarrier::eulMultiCarrierSupport
- R=hsThp: UtranCell::ueHsThpMeasSupport
- S=hs3MC: MultiCarrier::hsdpa3McSupport
- T=db3MC: MultiCarrier::hsdpaDb3McSupport

#### 4.2.21 al[atkcg][u] [-a|-u <alarm-id>] [ | <unix-cmds>]

Print the list of active alarms. Acknowledge/Unacknowledge an alarm.

Output can be piped through external unix utilities like "sort", "grep", "less", "more", etc.

It is possible to combine several options, eg: al, ala, altk, alatk, altkc, etc. These options are as follows:

By default the timestamps are printed in local timezone. For UTC time, use option "u", eg altu.

- al : active alarm list is printed in overview format, only four fields are shown per alarm.
- ala : same as al, but the full detailed list is added underneath the overview table.
- alt : same as al, but the time field is added to the table and the alarms are sorted chronologically.
- alk : same as al, but the list is separated in two parts, one for the unacknowledged alarms, and one for the acknowledged alarms.
- alc : same as al, but each alarm is displayed in CSV format and all fields are shown for each alarm.
- alg : same as al, but the MO instances with alarms are saved to a MO group called al\_group

The options -a/-u can be used for acknowledging/unacknowledging an alarm. The alarm is identified by its alarm id which can be printed with ala or alc.

Examples:

- altk : sort active alarm list by timestamp and show alarms in two separate tables depending on acknowledgment state
- altkc : same as above but alarms are displayed in CSV format and more fields are shown
- al | grep -i atmport : only print the alarms that match "atmport" (case insensitive)
- al -a 18 : acknowledge alarm number 18 (the alarm id is shown in "alc" or "ala")

Note: in alt and alk, the severity field is shortened to one character:

- C for Critical
- M for Major
- m for minor

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **w** for warning

Moshell generated alarms: on CPP nodes, the alarm list is read from the Alarm Client Interface (ALCI), however, some additional alarms are generated from moshell in the case where some CPP features are activated and disabled, since these MOs would not generate an alarm otherwise. This moshell feature is controlled by the uservariable "mosalarm" and is turned on by default which means that the "al" command takes about 1 or 2 seconds more to execute due to reading the status of those CPP features from the MO Client Interface.

#### 4.2.22 lg[abcdefghijklmnpqrstuvwxyz12345] [-l <logdirectory|logfile|zipfile>] [-m <minustime>] [-p <plustime>] [-s <startdate>] [-e <enddate>] [-g <boardgroup>] [-n <nodefilter>] [-x <XBlog-filter|ESlog-filter>] [-b <xb>] [-d <nrumps>] [<unix-cmds>]

Fetching and processing of node logs

All options can be combined together, except options **d**, **w**, **x**, **f** which can only be combined with options **r** and **c**.

##### CPP Log Options:

- **1** : Print the list of logs from the node.
- **2** : T&E disk log (/d/usr/cello/telogs)
- **3** : RNC SON log (/c/logfiles/SON/ANR\_EVENTLOG.xml and /c/logfiles/SON/TPS\_EVENTLOG.xml)
- **4** : MGW BGF log (/c/logfiles/BGF)
- **5** : MGW IPCS log (/c/logfiles/ipcs\_logs)
- **a** : Alarm log (ALARM\_LOG.xml). History of alarms raising and ceasing.
- **j** : Alarm durations (ALARM\_LOG.xml). Same as option **a** except that raising and ceasing are combined into one entry, together with the total duration of the alarm.
- **x** : Active alarms (ALARM\_LOG.xml). Snapshot of alarms active on a specific date/time given in -m/-s option.
- **e** : Event log (EVENT\_LOG.xml). History of MO events.
- **v** : Availability log (CELLO\_AVAILABILITY2\_LOG.xml). History of node/board/program restarts.
- **s** : System log (/c/logfiles/systemlog). History of node/board/program restarts.
- **p** : Post Mortem Dumps (/c/pmd) and LTE ENodeB error files (/c/logfiles/troubleshooting/error). History of board/program crashes. PMD files are saved permanently in moshell\_logfiles/logs\_moshell/pmdfiles/<nodeaddress>/pmd.
- **u** : Upgrade log (Trace.log/Trace.txt). History of system upgrades.
- **d** : Downtime log. History of node outages.
- **o** : MO command log (CORBA\_AUDITTRAIL\_LOG.xml and PNP\_LOG.xml). History of MO write commands (set/action/create/delete).
- **q** : MO command log in "trun" format (CORBA\_AUDITTRAIL\_LOG.xml). Useful for recovering configuration data which was not saved to CV before node restart.
- **l** : COLI command log (SHELL\_AUDITTRAIL\_LOG.xml). History of COLI commands.
- **n** : Moshell command log. To specify different or all nodes, use the -n <node-filter> option. If -m or -s option are not specified, the default is to show command history of the last 30 days.
- **y** : SecurityEvent log (CELLO\_SECURITYEVENT\_LOG.xml). History of O&M connection setups.
- **w** : Active O&M connections (CELLO\_SECURITYEVENT\_LOG.xml). Snapshot of O&M connections on a specific date/time given in -m/-s option.
- **z** : IP Transport log (CELLO\_IPTRAN\_LOG.xml).
- **t** : Trace and Error log (lh all te log read. to specify a different boardgroup than all, use the -g option).
- **g** : Board Restart error log (lh allpd llog -l ; lh ru llog -l -n 5).
- **h** : HW Inventory log (CELLO\_HWINVENTORY\_LOG.xml). This file must first be generated with the command hili mk on O&M MP.
- **k** : XB log. Fetches CMXB logs (HCS and Evo nodes) and SCXB logs (Evo nodes). Use -b <xb> to limit log fetching from a single board. Use -x <xblogfilter> to determine which logs are processed (see below).
- **b** : RLIB log (/c/logfiles/Rlib/RLIB\_PM\_LOG.xml), applicable to RNC only.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

**Pico Log Options:**

- a: Alarm log (/var/permanent/log/oss/alarmlog.log)
- e: FmEvent log (/var/volatile/log/fmevents.log)
- g: Runtime log (/var/volatile/log/runtime)
- h: AutoIntegration log (/var/permanent/log/oss/AutointegrationLog.txt)
- s: SystemEvent log (/var/permanent/log/oss/sysevent)
- u: Upgrade log (/var/permanent/log/oss/SWUpgradeLog.txt)
- y: SecurityEvent log (/var/volatile/log/security)

**RCS Log Options (DUS gen2):**

- a: Alarm log (AlarmLog)
- b: TN Application log (TnApplicationLog)
- h: AutoIntegration log (AiLog)
- k: Ericsson Support Information log (EsiLog)
- l: COLI command log (AuditTrailLog)
- o: MO command log (AuditTrailLog)
- v: Availability log (RBS\_CS\_AVAILABILITY\_LOG)
- u: Upgrade log (SwmLog)
- y: Security log (SecurityLog)
- z: TN Network log (TnNetworkLog)

Note: If firewall restrictions are preventing log export from the RCS node, try changing the uservariable `export_method`. Refer to the description in the file `moshell/moshell`.

**Format Options:**

- m : merge the different logs together (eg: lgaevm will merge alarm/event/availability logs).
- i : inverse chronological order.
- r : refetch the logs from the node. Logs are only fetched once and kept in cache. This option is used to refresh the session cache.
- c : print the output in csv format (semicolon separation).
- f : fetch the logs only and store them in a directory on the workstation. Different number of DSP dumps can be specified with option "-d <nrdumps>", eg: "lgf -d 2" to collect the last 2 dsp dumps. By default, no ENB DSP dumps will be fetched and no XB logs will be fetched.
- f1: fetch all logs including XB logs and the last ENB DSP dump.
- f2: fetch only XB logs.

**Time filtering:**

- The -s and -e options are used for specifying an absolute timespan: -s gives the starting date and -e gives the ending date. The format is `yyymmdd[.hhmm]`, for instance `20071230`, or `20071230.0800`.
- The -m and -p options are used for specifying a timespan relative to today's date: -m gives how long time backward and -p gives how long time forward. The format is in days, hours, or minutes, eg. `10d` (10 days), `2h` (2 hours), `30m` (30 minutes).

Note: the switch -s/-m can sometimes be omitted. E.g. `lgo 14` can be used instead of `lgo -m 14`, and `lgo 20080701.1200` can be used instead of `lgo -s 20080701.1200`. This works only when the options -e/-p are not used.

**Offline usage:**

The -l option allows to process the logfiles in offline mode, when not connected to the node. The argument of the -l option specifies the location where the logfile(s) are locally stored on the workstation. It can be:

- a single logfile
- a directory containing several logfiles.
- a zipped archive containing one or more logfiles.

By running the command `lgf` while connected to the node, it is possible to download all the logfiles to a local directory for later offline processing. The local directory can be specified as argument. If not specified, a default location is chosen

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

(~/moshell\_logfiles/logs\_moshell/lg/nodeaddress/date\_time). The local directory is then automatically compressed and saved in a zipped archive.

**XB log filters:** The XB log filter is specified with `-x <filter>` in the command `lgk` on nodes containing CMXB/SCXB boards, to specify the type of logs that will be displayed. The XB log filter shall be given as a combination of one or more of the following letters:

- o: OS log (default)
- s: SNMP log
- c: COLI command log
- b: Board manager log
- w: Switching event log
- m: Software management log
- e: Security log
- f: Firewall log
- h: Shelf manager log
- a: Application log
- t: Timing unit manager log
- p: Power and fan log
- x: All of the above listed logs

Example:

- `lgk -x oscb` - show the XB log entries from OS log, SNMP log, COLI command log and Board manager log

**ESI log filters:** The ESI (Ericsson Support Information) log filter is specified with `-x <filter>` in the command `lgk` on RCS nodes (TCU03/DUSgen2), to specify the type of logs that will be displayed. The XB log filter shall be given as a combination of one or more of the following strings, separated by commas:

- ai: AiLog
- al: AlarmLog
- avc: AvcLog
- capi: cpu\_load.log
- com: com.log
- coma: com\_alarm.log
- comi: ComInterfaceLog
- erl: erlang.log
- ev: NotificationLog
- lic: LicensingLog
- ltt: LttngLog
- nl: nl\_log (network loader)
- sec: SecurityLog
- swmi: SwmInternalLog
- sys: SystemLog
- tri: TriLog
- upg: SwmLog

Example:

- `lgk -x coma,erl,tri` -> show the log entries from com\_alarm.log, erlang.log and TriLog in the ESI

#### Notes:

- The output of the `lg` command can be filtered by piping to a unix command such as `grep`, `sort`, `less`, etc.
- In options `a` and `x`, the alarm severity field is shortened to one character: C=Critical, M=Major, m=minor, w=warning, \*=cleared.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- In option `j`, the alarm severity field consists of one letter if the alarm is still active. If the alarm is ceased then we see a character followed by a star, eg: `M*` means Alarm was raised with severity Major, then ceased.
- In `lgd`, the reason code for manual restarts can be translated with the command `mom restartreason`
- In `lgd`, the downtime values correspond to the following stages:
  - CPP downtime is the time elapsed between the row `CRIT Node down` and the row `Node operational` in `syslog` or the row `NODE IN Operational` in `avlog` (whichever row comes first).
  - Application downtime is the time elapsed between the row `CRIT Node down` in `syslog` and the row `RNC Node Restart Completed (RNC)`, `Cell .* enabled (WRBS)`, `VMGWs Unlocked/First VMGw Enabled (MGW)`, or `NODE IN Operational RestartCompleted (ERBS)` in `avlog`.
  - Jvm downtime is the time elapsed between the row `CRIT Node down` in `syslog` and the row `JVM Load Module is now operational` in `avlog` or the row `The Configuration Service is up and running` in `upgradelog`.
  - JvmRestart: For individual Jvm restarts, the Jvm downtime is the time elapsed between the row `Program CXCxxxx started` in `syslog` and the row `The Configuration Service is up and running` in `upgradelog`.
  - The node downtime figures in the summary table at the end of the printout represent the highest value between CPP downtime and Application downtime. The partial downtime figures are weighted against percentage of availability (when applicable).

#### Examples:

- `lga -s 20050705 -e 20050710` - show alarm log entries between the dates 20050705 and 20070710
- `lgaemc | grep -i atmport` - show all entries from the alarm/event logs matching the word `atmport` (case insensitive), display in CSV format (semicolon separated), and pipe to `grep`
- `lgx -m 14` - show alarms that were active 14 days ago
- `lgxc 20080704.1330` - show alarms that were active on the 2008-07-04 at 13:30 and print output in CSV format
- `lgvsm -s 20050705.1000` - show all entries from system log and availability log since the 20070705 at 10:00, merged in chronological order
- `lgar -m 10d -p 30m` - refetch the alarm log and show all its entries starting from 10 days ago and until 30 minutes from then.
- `lgf` - fetch all logs from the node and put them in the default location  
`~/moshell_logfiles/logs_moshell/lg/nodeaddress/date_time/node_logfiles.zip`
- `lgf /home/user/logs/rnc10` - fetch all logs from the node and put them in the zipped file  
`/home/user/logs/rnc10/<node>_logfiles.zip`
- `lgaemic -m 10h -l ~/moshell_logfiles/logs_moshell/lg/rnc10/20071122_1425` - parse the last 10 hours of the alarm and event logs stored in the folder `~/moshell_logfiles/logs_moshell/lg/rnc10/20071122_1425`, merge them and display them in CSV format and reverse chronological order
- `lgd -m 30d` - show all node restarts and related downtime from the past 30 days
- `lgt -g mp` - show the T&E logs of the boards of the board group "mp", sorted in chronological order
- `lgtaom -m 12h` - show the T&E logs of all boards merged with the alarm log and audit trail, for the past 12 hours
- `lgn -m 5 -n 137.58` - show the moshell command log for the past 5 days for all nodes whose address matches 137.58
- `lgk -m 5d -b 000100` - show OS status on SCXB in 000100 for past 5 days
- `lgk -m 10d -x ce` - show security events and COLI command history on XBs for past 10 days
- `lgk -x coma,erl,swmi -m 30` - shown ESI log entries from `com_alarm.log`, `erlang.log`, and `swminternal.log` for the last 30 days



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

## 4.3 Other commands

### 4.3.1 `uv [<string>|<var>=value]`

Display or change moshell configuration settings (also called "user variables").

The `uv` command used without any argument displays the values of all user variables that are usually specified in the **moshell** file and/or the `~/.moshellrc`. See Section 2.5 and **moshell** file for more info about the functionality of these variables.

If a string is given as argument, then only the variables matching the string will be displayed.

The `uv` command also allows to change a variable's value from within the moshell session.

For instance, if the variable `secure_shell` is set to 0 in the `~/.moshellrc`, it will be possible to run an moshell session in secure shell mode by just typing `uv secure_shell=1` at the moshell prompt. From that point on, all node connections that would have been performed using telnet will be performed using ssh.

Example:

1. `uv` - to print all variables
2. `uv sec` - to print all variables matching the string "sec"
3. `uv secure_shell=1` - to change a variable

Similar to the `get` command, you can also store the output of this command into a variable Example:

```
uv ^credential > $credential
```

### 4.3.2 `pv [<string>]`

To print all scripting variables or just those matching the `<string>`.

For example:

- `pv` print all scripting variables and their current value
- `pv ver` just print the scripting variables whose name match `ver`

The number of variables printed is saved to the variable `$nr_of_vars` Please refer to the "scripting" chapter for how to set a variable.

Note: To print a variable it is also possible to do: `l echo $variable`

### 4.3.3 `!/l <unix-command>`

Execute a unix command on the PC/workstation

Either the `!` or `l` can be used.

Examples:

1. `l pwd` (or the alias `lpwd`) - to know your current unix working directory
2. `l cd scripts/rbs3` (or the alias `lcd`) - to change your current working directory
3. `! less define_sectors.mo` - to view the command file you are about to run
4. `! vi define_sectors.mo` - to make a modification in your command file

Note:

- unix commands that are called with `!"` are never logged
- unix commands that are called with `!"` are logged if the user has started the logging with `l+`

### 4.3.4 `l+[m][m][s][o]/l-/l? [<logfile>]`

Open/close moshell logfiles.

`l+` is to *open a logfile*. If no logfile is given, then a default unique logfile is chosen.

The path of the default logfile is:

`~/moshell_logfiles/logs_moshell/sessionlog/<DATE>_<NODE>.log`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The `m` option is for *mute*, i.e.: no output will be displayed on the screen until the log is closed. All output will go to the logfile.

(the `mm` option is for extra mute, even less will be displayed on screen than with `l+m`).

The `s` option is for not printing the header "log open/log close". Can also be set with the `loginfo_print` user variable.

The `o` option is for overwriting the logfile, otherwise it is appended.

`l-` is for closing the logfile.

`l?` is for checking if a logfile is currently open.

It is possible to open several logfiles but only one at a time will be active. When one logfile is closed, logging will resume in the previous one. Examples:

```
l+ logfile1 #starts logging to logfile1, $logfile set to "logfile1"
get
pr
l+ logfile2 #stops logging to logfile1 and starts logging to logfile2
            #$logfile is set to "logfile2"
vii
l-          #stops logging to logfile2 and resumes logging to logfile1
            #$logfile set to "logfile1"
vols
l-          #stops logging to logfile1, $logfile stays set to "logfile1"
```

In this example, **logfile1** will contain the printouts from `get`, `pr` and `vols`, while **logfile2** will contain the printout from `vii`.

#### 4.3.5 `dbc[s][a] [<cvname>|<dbdat-file>|<cvzip-file>|<mobatch-folder>]`

Database consistency check

##### Purpose

When there are inconsistencies in the SQL database, it can lead to problems such as traffic performance degradation, upgrade failures, or cyclic restarts. Often the symptoms appear only after the next node restart or upgrade so it is not easy to know since how long the data has been corrupted and how far back we should rollback to find a non-corrupted CV. Therefore it is a useful preventative measure to regularly perform a database consistency check.

Please refer to RAN PLM info 662, 664, 808, 914, for more details on the different types of corruptions and possible remedies.

##### Command argument

When the command is run on its own without option nor argument, it will make a CV (`cvmk`), fetch the `db.dat` file (`ftget`), remove the cv (`cvrn`), then analyse the `db.dat` on the workstation and look for corruptions.

The argument (optional) can either be:

- the name of the CV to fetch and process.
- the path to a `db.dat` file or `cv.zip` that has been fetched previously and is located on the workstation.
- the path of a folder containing logs of the `dbc` printout run from `mobatch` ("`a`" option).

##### Command options

- `s` (`dbcs`): to fetch and check the startable CV. Only applicable in online mode.
- `a` (`dbca`): to analyse existing `dbc` logs, taken with `mobatch` from many nodes or `db.dat` files. The `dbca` command combines the multiple `dbc` printouts into one single `dbc` printout showing all exceptions found in the different nodes or `db.dat` files.

##### Running a consistency check on the whole network

- If the `db.dat` or `cv.zip` files have already been collected, it is possible to audit them all in parallel with `mobatch`. The syntax is:

```
mobatch [-r] -d /path/to/folder dbc
```

In this case `mobatch` will run the `moshell` sessions in `sql` mode against the files instead of against the nodes.

The first argument must specify the path to the folder containing the `db.dat` and/or `cv.zip` files.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The `-r` option is for recursive search in the folder, otherwise only the files directly under the folder are examined.

- If the `dbdat/cvzip` files have not previously been collected, it is possible to run `:mobatch /path/to/sitefile dbc`. In this case `mobatch` will run the `moshell` sessions in online mode against each node specified in the sitefile and audit the current CV.

## Background info

The configuration data of the node is kept in a SQL database in RAM memory on the central MP and can be backed up on disk (`/d/configuration/cv`) for permanent storage. The main purpose of the database is to store the persistent data of the MOs. An MO is made of up to three layers:

- the MAO layer (Management Adaption Object)
- the FRO layer (Facade Resource Object)
- the RO layer (Resource Object)

An MO always consists of one MAO. There is a one to one relation between the MO and its MAO.

Underneath the MAO there can be one or more FROs, or in some cases no FRO. Examples: The `SwAllocation` MO consists only of a MAO without FRO/RO. The `Mtp3bSpltu` MO consists of one MAO and one FRO/RO. The `Aal2PathVccTp` MO consists of one MAO and two FROs/ROs. The FROs are used for controlling the actual resources, the ROs.

The MAOs and FROs use separate SQL tables for data storage. The ROs do not store any persistent data as this is handled by FRO.

The MAOs keep their persistent data in the SQL table `modata_r2` or `modata_r3`. For the FROs, there is a separate table for each MO class. For instance the FROs of `PlugInUnit` MOs use the table `cspiuresource`, the FROs of `Program` MOs use the table `cspgmresource`, etc.

## Printout example and comments

Important to keep in mind: the same fault can have appear in several of the checks below. For instance, an MO which has not been correctly deleted from the node could result in corruption shown in checks 8, 12, 13. A `UtranRelation` with incorrect `frequencyRelationType` will result in corruption shown in checks 17 and 18. Usually when the same fault appears in several checks, the fault that is higher up in the list is the one that should be fixed first.

```
-----
1) MAOs with non-recommended characters in the MAO name
      (recommended range: -_/.A-Za-z0-9!%*:): YES
2039 TransportNetwork=1,AtmTrafficDescriptor=UBR+_230_QoS3 MAO name: UBR+_230_QoS3
5642 RncFunction=1,IubLink=iub 45 MAO name: iub 45
```

### \*\*\*\*\* Comment \*\*\*\*\*

The MOs above contain a "+" and space sign in their MO id. These characters are not known to have a system impact on the node but it is not recommended to use them as they may not be handled properly by the O&M client (`moshell/AMOS/OSSRC`). Having MOs in this list will give the result "OK with warnings". MOs can be renamed by using the "rset" command.

```
-----
2) MAOs with dangerous characters in the MAO name ` , = ^ " | ' Ã (HL11572/UABtr75948): YES
5631 TransportNetwork=1,Aal2PathVccTp=TransportNetwork=1,Aal2PathVccTp=88
      MAO name: TransportNetwork=1,Aal2PathVccTp=88
```

### \*\*\*\*\* Comment \*\*\*\*\*

The MO above contains strictly forbidden characters in the MO id: ";" and "=". Using these characters can cause cyclic node restarts, see TR HL11572/UABtr75948.

```
-----
3) MAOs without FROs: YES
3321 Aal2PathVccTp=Iuc-2300-2351-7-95 : aal2pathepfroid=5
      not found in table aal2pathepfrotable_6
5644 IubLink=503 : theclientsuniqueid=33
      not found in table roamfroiublinkdbtable_09
```

### \*\*\*\*\* Comment \*\*\*\*\*

Here we have some MOs where the MAO is pointing to a non-existing FRO. Start `moshell` in `sql` mode towards the `db.dat` file (`moshell -d /path/to/db.dat`) and run the "get" command on these MOs. For instance, the "get" command will show that the `IubLink` MO has `froid` value 33 but when looking in the corresponding FRO table `roamfroiublinkdbtable_09` with "sql select" command, we will see that there is no entry with this id.

```
-----
4) FROs without MAOs: YES
theclientsuniqueid=198 in roamfroexternalgsmcelldbtable_06 (ExternalGsmCell)
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

## \*\*\*\*\* Comment \*\*\*\*\*

Here we have the reverse situation than above. There is no entry in the MAO table which has a frold pointing to this FRO. We can run the command "fro externalgsmcell" to print the frold value of all MAO instances and we will see that none has the frold 198.

## 5) MAOs with duplicate LDN: YES

```
13 SystemFunctions=1,Licensing=1 (known issue: TR UABtr63243 - no impact)
65 SystemFunctions=1,Licensing=1 (known issue: TR UABtr63243 - no impact)
85 SystemFunctions=1,Licensing=1 (known issue: TR UABtr63243 - no impact)
```

## \*\*\*\*\* Comment \*\*\*\*\*

This indicates that there are several entries in the table modata\_r2 which refer to the same MAO. Sometimes this can be a problem, especially if the MAO is connected to a FRO. In this case, Licensing MO is made only of a MAO layer so no FRO are affected. This particular problem is known in CPP5/PP6 and fixed in CPP7. It has no system impact.

## 6) MAOs with duplicate froId: YES

```
3440 IubLink=Iub-11 frold=30
3443 IubLink=Iub-22 frold=30
```

## \*\*\*\*\* Comment \*\*\*\*\*

Here we have two different MAOs that point to the same FRO, this is not allowed since a FRO can only be connected to one MAO. We could check this by starting moshell in sql mode and running the "get" or "fro" command on these MOs to print the frold.

## 7) Mismatch between number of MAO and FRO instances: YES

```
Aal2PathVccTp      : 21 MAOs, 41 FROs
                    (For Aal2PathVccTp, there should be 2 FROs per MAO).
ExternalGsmCell    : 1133 MAOs, 1134 FROs
IubLink            : 33 MAOs, 32 FROs
```

## \*\*\*\*\* Comment \*\*\*\*\*

This check is a summary of issues found in points 3,4,5,6. We count the number of MAO and FRO instances for each MO class and show those where there is a mismatch. The faulty MO instances can be found in one of the previous four checks.

## 8) MAOs referring to non-existent MAOs: YES

```
429 Subrack=MS reservedBy RncFunction=1,IubLink=80
1709 RncModule=1 reservedBy RncFunction=1,IubLink=80
```

## \*\*\*\*\* Comment \*\*\*\*\*

This indicates that some MOs have a reference to an MO that does not exist on the node. This can happen when a MO is deleted from the node, sometimes the system fails to remove it from reference attributes pointing to it. In example above, the reservedBy attribute of some MOs did not get updated properly when the MO IubLink=80 was deleted from the node.

## 9) MAOs defined under a different parent than FRO: YES

```
3441 IubLink=Iub-11,NodeSynch=1 maoParent: IubLink=Iub-11 (30)
                                froParent: 30 (IubLink=Iub-11 IubLink=Iub-22)
3444 IubLink=Iub-22,NodeSynch=1 maoParent: IubLink=Iub-22 (30)
                                froParent: 31 ()
3568 UtranCell=Iub-54-1,UtranRelation=9875a maoParent: UtranCell=Iub-54-1 (145)
                                froParent: 223 (UtranCell=Iub-57-2)
```

## \*\*\*\*\* Comment \*\*\*\*\*

The MAO knows its parent from the LDN. In some cases, the parent address is also stored in FRO by an attribute giving the frold of the parent MO in the MO tree. This check shows if the parent reference stored in MAO is different to the parent reference stored in FRO.

In the first example, the discrepancy is due to the fact that two MAOs have the same frold (problem highlighted in check 6), this issue has a repercussion here since the children of this MO do not know which of the two MOs with frold 30 are the parent. In the second example, we see the frold of the parent points to a FRO that either does not exist or is not connected to any MAO. In the third example, we see that the frold of the parent points to a different MO than the one given in the LDN by MAO. Note: the number in brackets next to the LDN is the frold of that MAO.

## 10) Inconsistent MO references between MAO and FRO: YES

```
3443 IubLink=Iub-22 sctpRef: Sctp=MS-15 (3) sctpfroid: 2 (Sctp=MS-14)
4135 UtranCell=Iub-11-1 iubLinkRef: IubLink=Iub-11 (30)
                                iublinkfroid: 30 (IubLink=Iub-11 IubLink=Iub-22)
5789 UtranCell=U30717,UtranRelation=U05938
                                utranCellRef: IurLink=rncka62,ExternalUtranCell=U05938 (1743)
                                nutrancellfroid: 1999 ()
2406 IpAccessHostPool=Iub
                                ipAccessHostRef: IpAccessHostEt=ES1-27 IpAccessHostEt=MS-26 IpAccessHostEt=MS-7
                                ipaccesshostfroid: IpAccessHostEt=ES1-02 IpAccessHostEt=ES1-27 IpAccessHostEt=MS-26
```

## \*\*\*\*\* Comment \*\*\*\*\*

MO references are sometimes kept in MAO or FRO only, but sometimes they are kept in both parts. It is important that an MO

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

reference kept both in MAO and FRO should be the same in both.

In the first example, the MO has a reference stored in MAO which is different to the one stored in FRO.

In the second example, the reference stored in FRO points to two different MAOs, this is due to the problem highlighted in check 6 with duplicated froid.

In the third example, the FRO reference points to a FRO that either does not exist or does not have a MAO.

In the fourth example, the list of MO references is different in the MAO attribute ipAccessHostRef compared to the FRO attribute ipaccesshostfroid Note: the number in brackets next to the LDN is the frold of that MAO.

```
-----
11) MAOs missing from reservedBy list: YES
2401 IubLink=Tub-1226 sctpRef Sctp=MS-15 reservedBy
```

\*\*\*\*\* Comment \*\*\*\*\*

This check indicates that the MO on the left has a reference to the MO on the right (via the attribute stated in the middle) but does not appear in the reservedBy list of the MO on the right. To check this we start moshell in sql mode (moshell -d dbdat/cvzip) and perform the get command on the MO on the right and we see that the MO on the left cannot be found in the reservedBy list eventhough it has a reference to that MO. This is a one-way relation between the MOs and is a fault.

```
-----
12) MAOs found only in reservedBy list: YES
858 RncModule=11 reservedBy IubLink=Iub-198
```

\*\*\*\*\* Comment \*\*\*\*\*

This is the opposite problem than the previous check. It indicates that the MO on the right (in this case the IubLink) appears in the reservedBy list of the MO on the left, even though it has no reference to that MO. To check this we start moshell in sql mode (moshell -d dbdat/cvzip) and perform the get command on the MO on the right. We will see no attribute containing any reference to the MO on the left. When we do the get command on the MO on the left we see that the MO on the right appears in the reservedBy list anyway. This is a one-way relation between the MOs and is a fault.

```
-----
13) Inconsistent sequence of moRefs in MAO: YES
1003 SectorAntenna=1,AuxPlugInUnit=RRU-1 persistentReservers: 5, actual: 7
```

\*\*\*\*\* Comment \*\*\*\*\*

This check applies to MAO attributes of type sequence:moRef. It indicates if there is a discrepancy between the announced number of MO references in the attribute (shown at the beginning of the attribute value, in square brackets, eg: [5]), and the actual number of MOs listed in the attribute. In the example above, the attribute is supposed to be a sequence of 5 MOs but in fact it contains 7 MOs. To check, start moshell in sql mode and perform the "get" command on the MO.

```
-----
14) FROs referring to non-existent FROs (CSR1473974): YES
6513 SpDevicePool=DcDevice,DcDevice=1 subrackfroid 9 ()
6494 RncModule=13 piuofroid 45 ()
```

\*\*\*\*\* Comment \*\*\*\*\*

This check shows the FROs that have a reference to a FRO that does not exist on the node. Only MOs which have not already been displayed in check number 10 will be displayed here. To find out the faulty FRO table and row, start moshell in sql mode and perform the "get" command on the MO.

```
-----
15) Remaining old FRO table versions (HL93894/WRNae89948/HM76376/HS48645/HR63086): YES
cspgmresource_03 (current): cspgmresource_02 (old)
csxpresource_01 (current): csxpresource (old)
ecnprsectordata_4 (current): ecnprsectordata_3, ecnprsectordata_2, ecnprsectordata_1 (old)
```

\*\*\*\*\* Comment \*\*\*\*\*

This check shows the FRO tables which exist in multiple versions: the latest table version is shown on the left-hand side, and the list of old table versions is shown on the right-hand side. The old table versions should normally be removed at system upgrade.

```
-----
16) Corrupted MAO entries in modata table: YES
3280 Aal2QosProfile=adoffbdoff reservedBy incorrect nrOfElements:
qF3||rF1=1,5=1,141=aal2pathvcctp=99||rF1=1,5=1,141=TransportNetwork=1,141=88|rF1=1,5=1,141=TransportNetwork=1,169=8
```

\*\*\*\*\* Comment \*\*\*\*\* This check looks at the syntax of the attribute data of the MAOs listed in the modata\_r2/modata\_r3 table. Any corrupted attribute data will be shown as an exception in the get printout for that MO. The correct syntax of each entry should be: MoType^Revision^LDN^PrimaryKey^attributeName^attributeData^ And the syntax of the attribute data should be: dataType AVCflag Data . With:

```
dataType:  s=String, r=Reference, t=Struct, f=Float, q=Sequence,
           i=Integer, l=Long, b=Boolean
AVCflag:   T=isAVCNotifier , F=notAVCNotifier
Sequence:  dataType AVCflag noOfElements|attributeName|attributeData|....
Struct:    dataType AVCflag noOfElements attributeName attributeData ....
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

-----  
17) MAOs without parent: YES  
14498 AtmPort=ES-1-2-1-ima55,VplTp=vp1,VpcTp=1 missingParent: AtmPort=ES-1-2-1-ima55,VplTp=vp1

\*\*\*\*\* Comment \*\*\*\*\*

This check reports the list of MAOs whose parent does not have an entry in the modata\_r2/modata\_r3 table.

-----  
18) MAOs with invalid froId: YES  
22458 Subrack=ES-2,Slot=21,PlugInUnit=1,GeneralProcessorUnit=1,LoadControl=1 froId=  
22485 Subrack=ES-2,Slot=20,PlugInUnit=1,GeneralProcessorUnit=1,LoadControl=1 froId=

\*\*\*\*\* Comment \*\*\*\*\*

This check reports the list of MAOs that have a invalid FRO id, e.g. empty value instead of integer.

-----  
19) Instance-based PM Scanners containing missing MO instances (HR95951/CSR2375943/CSR2375969/ER690205): YES  
533 USERDEF.RNC\_CNHH\_RNC71\_3.Profile=982.Continuous=Y.STATS missing MO instances:  
UtranCell=BU317L,UtranRelation=BU980L  
536 USERDEF.RNC\_CNHH\_RNC71\_4.Profile=983.Continuous=Y.STATS missing MO instances:  
UtranCell=BU530L,UtranRelation=BU279L  
UtranCell=BU530L2,UtranRelation=BU279L2  
624 USERDEF.RNC\_CNHH\_RNC71\_2.Profile=981.Continuous=Y.STATS missing MO instances:  
UtranCell=BU279M,UtranRelation=BU279N  
UtranCell=BU279K,UtranRelation=BU279N  
UtranCell=BU279L,UtranRelation=BU279N  
UtranCell=BU279L,UtranRelation=BU530L  
UtranCell=BU279M2,UtranRelation=BU279N  
UtranCell=BU279L2,UtranRelation=BU279N  
UtranCell=BU279L2,UtranRelation=BU530L2

\*\*\*\*\* Comment \*\*\*\*\*

This check reports the list of Instance-based PM scanners that contain MO instances which don't exist anymore in the node. Any such scanners should be deleted and replaced with Class-based scanners or Instance-based scanners containing existing MO instances. The scanners can be printed in dbdat mode with pst/pgets. Refer to TR HR95951 or PLM info 914 for more info.

-----  
20) Instance-based PM Scanners containing more than 1000 MO instances (HR95951/CSR2375943/CSR2375969/ER690205): YES  
533 USERDEF.RNC\_CNHH\_RNC71\_3.Profile=982.Continuous=Y.STATS number of MO instances: 10118  
536 USERDEF.RNC\_CNHH\_RNC71\_4.Profile=983.Continuous=Y.STATS number of MO instances: 10271  
624 USERDEF.RNC\_CNHH\_RNC71\_2.Profile=981.Continuous=Y.STATS number of MO instances: 10602  
627 USERDEF.RNC\_CNHH\_RNC71\_1.Profile=881.Continuous=Y.STATS number of MO instances: 10892

\*\*\*\*\* Comment \*\*\*\*\*

This check reports the list of instance-based PM scanners that contain more than 1000 MO instances. It is recommended to replace these scanners with Class-based scanners due to impact on the node JVM performance. The scanners can be printed in dbdat mode with pst/pgets. Refer to TR HR95951 or PLM info 914 for more info.

-----  
21) Jvm admClassPath containing LoadModules not part of the current UpgradePackage (CSR2447811): YES  
7 Jvm=1 incorrect LMs in admClassPath: CXC1720482\_R73D61 CXC1727628/2\_R5X02 CXC1726723\_R73D03 CXC1725907\_R73D62

\*\*\*\*\* Comment \*\*\*\*\*

This check reports if the Jvm admClassPath contains any LMs that are not listed in the current UpgradePackage (attribute: ConfigurationVersion::currentUpgradePackage::loadModuleList)

-----  
22) Missing FRO tables: YES  
dnsclient (IpOam)  
httpsdb (WebServer)

\*\*\*\*\* Comment \*\*\*\*\*

This check reports if there are MAO instances where the corresponding FRO table is missing. The output shows the name of the missing FRO table and the corresponding MO class. This is a complement to check 3 which detects MAO instances where the FRO entry is missing from the FRO table but does not detect cases where the whole FRO table is missing.

-----  
101) FRO attributes containing the froId field but not the ldn field (HR88263): YES  
1867 EUltraNetwork=1,ExternalENodeBFunction=5051-530675,ExternalEUtranCellFDD=5051-530675-2 parentref,eutranfrequenc  
1868 EUtranCellFDD=SHBDEM2,EUtranFreqRelation=1275,EUtranCellRelation=5051-530675-2 parentref,neighborcellref

\*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RBS/ERBS only and does not appear in other node types (RNC/MGW). This check shows if there are any attributes containing the text "<attr name='froId'" but not the text "<attr name='ldn'".

-----  
101) MTP3 pointCode collision (WRNae82362, SCS695737): YES  
Point code 13749348 used by Mtp3bSpAnsi=1 and Mtp3bSpAnsi=1,Mtp3bSrs=r821s



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

## \*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC/MGW nodes only and does not appear in other node types (RXI/RBS).

This check shows if there are any MTP3 routes that use the same destinationPointCode as the node's own point code. To check this, start moshell in sql mode and run the "get" command on both MOs listed to crosscheck that the pointcode is the same. The fault is described in TR WRNae82362.

-----  
102) Number of Fans mismatch (CSR 2101402/HQ64197): YES

23 Equipment=1,Subrack=MS nrOfFans: 4 nr\_denib: 2

## \*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC/MGW nodes only and does not appear in other node types (RXI/RBS).

In addition, it only applies to the nodes that use Subracks with fanConfiguration BFD528, eg. RNC3820, MGW GMPv4.

The purpose is to check if the number of Fan MOs defined under each Subrack is the same as the number of fans specified in the attribute numberOfDenibDevices (if >0) or in the attribute Subrack::subrackProdType::fanConfiguration (if numberOfDenibDevices <0).

-----  
103) Inconsistency in UtranRelation nodeRelationType or frequencyRelationType

(HP94489/WRNae68940/WRNae72810): YES

5596 UtranCell=Iub-10-1,UtranRelation=Softer-Iub-10-3 nodeRelationType: 1,  
actual: 0 (cellRef:UtranCell=Iub-10-3)

22205 UtranCell=U31618,UtranRelation=U31477 frequencyRelationType: 1,  
actual: 0 (f1=f2=1007)

37068 UtranCell=85276B,UtranRelation=1 frequencyRelationType: 0,  
actual: 1 (f1=10737, f2=10713)

## \*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC only and does not appear in other node types. It is only for MO instances of type "UtranRelation" and cross-checks the value of the attributes nodeRelationType and frequencyRelationType against the real setting.

The first UtranRelation in the example has nodeRelationType set to 1 (Inter-RNC, meaning it is supposed to be a relation to an ExternalUtranCell) but the cellReference to which it is pointing to is an internal UtranCell, so it should actually have the nodeRelationType 0 instead of 1.

The second UtranRelation MOs in the example has a discrepancy in the frequencyRelationType which is set to 1 (Inter-frequency) whereas the originating Cell and destination Cell have the same frequency, so it should actually be 0 (Intra-frequency). The third UtranRelation is the opposite scenario where the frequency of the originating Cell and destination Cell are different but the frequencyRelationType is set to 0 (Intra-frequency).

-----  
104) Inconsistency in UtranCell interFreqRelCntr or intraFreqRelCntr

(HP94489/WRNae68940/WRNae72810): YES

364 UtranCell=85162B intraFreqRelCntr: 27, actual: 28 (by frequencyRelationType and uarfcnDl)

364 UtranCell=85162B interFreqRelCntr: 1, actual: 0 (by frequencyRelationType and uarfcnDl)

680 UtranCell=85276B intraFreqRelCntr: 25, actual: 24 (by uarfcnDl)

680 UtranCell=85276B interFreqRelCntr: 0, actual: 1 (by uarfcnDl)

## \*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC only and does not appear in other node types. It is only for MO instances of type "UtranCell" and cross-checks the value of the attributes intraFreqRelCntr and interFreqRelCntr against the real setting.

The real setting is checked by checking the frequencyRelationType of the UtranRelations defined underneath the UtranCell. Both the frequencyRelationType attribute and the real frequencyRelationType (found out by looking at the uarfcn of originating and destination cell) are checked.

To check this manually, start moshell in sql mode (moshell -d cvzip/dbdat) and run the command "lget utranCell=xxx cntr/freq.\*type". It will show the value of the cell's freqRelCntr and frequencyRelationType of underlying relations. If they mismatch, the cell is shown as mismatching "by frequencyRelationType and uarfcnDl". If they match but some of the frequencyRelationType settings are incorrect (see previous check 17) then the mismatch will be shown "by uarfcnDl". In the first two lines we can see that the UtranCell 85162B has 27 intra-frequency relations and 1 inter-frequency relations. When we check the frequencyRelationType attribute and the uarfcn of the originating and destination cells, both indicate that this is not correct. It appears that there is actually 28 intra-frequency relations and 0 inter-frequency relations. In the next two lines we can see that the UtranCell 85276B has 26 intra-frequency relations and 0 inter-frequency relations, but in reality there are 24 intra-frequency and 1 inter-frequency. In this case the frequencyRelationType setting of the UtranRelation is misleading and has been flagged in check 15.

-----  
105) Inconsistency in RncFunction cellRelCntr (HT37388): YES

RncFunction=1 cellRelCntr=21743, nrOfRelations=21410 (1304 CoverageRelation, 6 EutranFreqRelation, 12 GsmRelation, 2

## \*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC only and does not appear in other node types. It checks that the value of the attribute cellRelCntr in the RncFunction MO should be equal to the total number of MO instances of type CoverageRelation+EutranFreqRelation+GsmRelation+UtranRelation. It can be checked manually with the command: prs relation=

-----  
106) Inconsistency in UtranCell gsmRelHoAndCellReselCntr: YES

5289 UtranCell=U32194C gsmRelHoAndCellReselCntr: 21, actual: 22

17389 UtranCell=G32105C gsmRelHoAndCellReselCntr: 25, actual: 22



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

\*\*\*\*\* Comment \*\*\*\*\*

This check is specific for RNC only and does not appear in other node types. It checks that the value of the attribute `gsmRelHoAndCellReselCntr` in each `UtranCell` MO should be equal to the actual number of children `GsmRelation` MOs with `mobilityRelationType=0`. It can be checked manually with the command:

```
get utrancell=xxx,gsmrelation mobilityRelationType ^0
```

#### 4.3.6 dbd [<cvname>|<dbdat-file>|<cvzip-file>] [<cvname>|<dbdat-file>|<cvzip-file>]

Purpose: To compare the data of two CVs or db.dat files

Arguments:

- the name of the CV to fetch and process.

or:

- the path to a db.dat file or cv.zip that has been fetched previously and is located on the workstation.

Uservariables:

- `dbd_exclude_moclasses` : to exclude certain MO classes from the comparison
- `dbd_exclude_attributes` : to exclude certain attributes from the comparison

Examples:

- `dbd Rb_CXP9021775_R1BF04_121217_2155 Fi_CXP9021775_R1BF04_121217_2201` - comparing the CV before upgrade and the CV after upgrade
- `dbd /home/userid/rnc1.db.dat /home/userid/rnc2.db.dat` - comparing a db.dat from one node against the db.dat of another node

#### 4.3.7 <ose/coli command> [|<unix-cmds>]

Send a COLI command to the CPP node's OSE shell. Type "h ose" for syntax help and "?" to view available commands.

The command are sent to the node using either telnet or ssh, depending on the value of the moshell setting `secure_shell` (See Section 2.6)

The password is not required if it is defined in the `ipdatabase` or the `$password` variable (otherwise, the user will be prompted to enter the password).

To find out all available COLI commands, type `? or lhsh <lnh> ?`

It's possible to pipe the command to any external utility the machine which moshell is running on (such as **grep**). Example:

- `ls -l /d/loadmodules | sort -nk 3` : to sort LMs by size
- `te log read | /home/eric/tools/decoder.pl | /home/eric/tools/flow.pl`
- `lhsh 001400 te log read | grep -i error` : to show errors in TE log

Several commands can be run on the same line by separating each command with a semicolon.

If more than 5 commands are specified on the line, then these will be sent via a command file on the node (quicker).

The user variable `fast_coli_threshold` controls the number of commands above which a command file will be transferred to the node.

By default it is 5 but it is possible to reduce or increase this setting with the `uv` command (see chap 2.5 for more info on uservariables).

For instance by setting `fast_coli_threshold` to 0, COLI commands will always run via command file, this can help to avoid printout corruptions (certain commands producing large printouts can sometimes get corrupted by spurious echo characters when run directly without command file).

Examples:

- `vii ; pwd ; cd /d/loadmodules ; pwd ; llog`
- `lhsh 001100 ; vii ; te e trace1 NBAP*`
- `lhsh 001400 ; te filter set "([1]<8)OR([1]>=9)AND(LEN<>33)" NBAP*`
- `lhsh 001400 ; te filter set "([1]<>8)AND([1]<>$12)AND([1]<>$14)AND([1]<>$11)" NBAP*`
- `lhsh 001400 ; te log freeze -grp state_change`  
`WaitForActivation -> WaitForCapacity" 100`
- `lhsh 012000 ; MsbHostMo_StartPing -d 10.173.137.130 -z 10.173.137.1`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `lhsh 000200 ; EtHostMo_startPing -d 10.164.41.132 -h 1 -c 20 -s 54`

Note that it is safer to do `lhsh <lnh> <command>` instead of `lhsh <lnh> ; command` in case the board is not reachable. Example: if you want to format /d on board 001400: if you do `lhsh 001400 ; formathd /d` and the board 001400 is not reachable then the command is sent to the Hub MP whereas with the command `lhsh 001400 formathd /d`, if the board 001400 is not reachable then the command is not sent at all. But for certain commands like "te filter set", "te log freeze", "EtHostMo\_startPing" or "MsbHostMo\_StartPing", the semicolon has to be entered after the "lhsh" in order to force moshell to actually log into the board.

### Running SQL commands

Examples:

- `sql+` (start the sqlc client on the node)
- `sql select name from tables`
- `sql select * from tables where name like '%iur%'`
- `sql select * from cspgmresource_01 where pno='CXC 132 0784'`
- `sql update cspgmresource_01 set poolsize=20000000  
where pno='CXC 132 0784'; commit;`

Note: If the `osa_coli` program is running on a different board than the hub MP, semicolons must be put around `sqlc`:

- `lhsh 001100 ; sqlc ; select * from tables`

### Running 3GSIM/CORBEN/LOCO commands

Examples:

1. `crb st`
2. `crb rf -f /c/corben/uefile.cmd; crb rf -f /c/corben/cellfile.cmd`
3. `3gsim lb`
4. `3gsim lss`
5. `loco ls`
6. `corben ; ts ; statistics`
7. `corben ts ; corben statistics`

### Running NCLI commands

Note: `ncli` command completion not supported when run from moshell

Examples:

- `ncli alarms` - Active alarm list
- `ncli help ; man search` - List ncli commands. Print help of the ncli command "search"
- `ncli search . ""` - List all MOs
- `ncli search . "" operationalState==0` - List all disabled MOs
- `ncli search . "" operationalState==0 AND administrativeState==1` - List all MOs unlocked and disabled
- `ncli group -a -e ( . "" operationalState==0 ) ; get -group userLabel` - Put all disabled MOs in a group, then read the userLabel attribute of these MOs
- `ncli ; get . userLabel ; set . userLabel=(String)RNC11 ; get . userLabel` - Read and change userLabel attribute on ManagedElement MO

Adding/Removing a static route:

- `ncli action IpOam=1,lp=1,lpRoutingTable=1 addStaticRoute (String)"0.0.0.0" (String)"0.0.0.0" (String)"137.58.152.1" (int)110 (boolean>false`
- `ncli action IpOam=1,lp=1,lpRoutingTable=1 deleteStaticRoute { destinationIpAddr=(String)"0.0.0.0" destinationNetworkMask=(String)"0.0.0.0" nextHopIpAddr=(String)"137.58.152.1" routeMetric=(int)110 }`

Other examples:

- `ncli ; cd TransportNetwork=1 ; search . AtmPort ;  
search . AtmPort operationalState==0`
- `ncli group -a -e(. UpgradePackage);group -l;get -group state;`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
action SwManagement=1,UpgradePackage="CXP9013831_R9YC/6" verifyUpgrade
```

### **Running CMXB commands on HCS node (RNC3820/MGW GMPv4).**

Examples:

- `lhsh 000100 cmxbsh ; help ; ls /bin ; ls /usr/bin ; iss ; help`
- `lhsh 000000 cmxbsh ; iss ; show interfaces status; show mac-address-table`
- `lh scb cmxbsh ; iss ; show interfaces status; show mac-address-table`

### **Running CMXB/SCXB commands on EvoC (RNC8200).**

Examples:

- `xbsh 000200 ; help ; ls /bin ; ls /usr/bin ; iss ; help`
- `lh cmxb help ; ls /bin ; ls /usr/bin ; iss ; help`
- `lh scxb help ; ls /bin ; ls /usr/bin ; iss ; help`
- `lh xb help ; ls /bin ; ls /usr/bin ; iss ; help`

### **Running telnet commands in MSB4 (MGW)**

The telnet username (and password if applicable) must be entered after the telnet command, separated by semicolons and the exit command must be given at the end.

Examples:

- `lhsh 000700 telnet 10.7.0.5 ; root ; shroot ; pwd ; ls -l /var ; exit`

### **Changing shell password**

Examples:

- `passwd ; <old password> ; <new password>`
- `secmode -l 2 ; <new password>`

### **Running commands towards an AXE node (prerequisite: uservariable "lincli" must be set to 3).**

Examples:

- `allip`
- `rxmsp:mo=rxotg-17,subord`
- `rxbli:mo=rxotrx-17-1` (command will be auto-confirmed)

Limitations: it is not possible to release the terminal in order to view "ordered" printouts.

## **4.3.8 coli**

Open an interactive COLI or RCS-COLI session to the node.

## **4.3.9 comcli**

Open an interactive COMCLI session to the node. Only applicable for COM nodes.

## **4.3.10 c+/c1/c2/c-/c?**

Switch between the COM node's linux shell, rcs-coli shell or comcli shell.

To switch between the linux/rcs-coli shell and the comcli shell, use the command `c+/c1/c2` :

- `c+` sets the uservariable comcli to 2, giving access to the comcli shell
- `c1` sets the uservariable comcli to 1 and coli\_shell to 1, giving access to the rcs-coli shell
- `c2` sets the uservariable comcli to 1 and coli\_shell to 2, giving access to the linux shell

If the node does not have a rcs-coli shell then the commands `c1` and `c2` will be equivalent, and it is also possible to use the command `c-`.

The `c?` command is for checking what is the current shell.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.11 <linux/rcs-coli/comcli command> [<unix-cmds>]

Send CLI commands to the COM node's linux shell, rcs-coli shell or comcli shell.

The commands are sent to the node over ssh. The password is not required if it is defined in the ipdatabase or the `$password` variable (otherwise, the user will be prompted to enter the password).

To print the list of linux commands, type "ls" on the list of directories shown in the `$PATH` environment variable (`echo $PATH`). To print the list of rcs-coli commands, type "help". To print the list comcli commands, type "?" in comcli mode.

To switch between the linux/rcs-coli shell and the comcli shell, use the command `c+/c1/c2` :

- `c+` sets the uservariable comcli to 2, giving access to the comcli shell
- `c1` sets the uservariable comcli to 1 and coli\_shell to 1, giving access to the rcs-coli shell
- `c2` sets the uservariable comcli to 1 and coli\_shell to 2, giving access to the linux shell

If the node does not have a rcs-coli shell then the commands `c1` and `c2` will be equivalent, and it is also possible to use the command `c-`.

To switch between linux shell and rcs-coli shell, the uservariable `linux_shell` can be used:

- `linux_shell=0 -> rcs-coli shell`
- `linux_shell=1 -> linux shell`

Within the comcli shell, there are two modes: exec mode and config mode. Exec mode is the default. To switch to config mode type "configure". The comcli allows to perform MO commands (get, set, create, delete, action, etc). More information about the comcli shell can be found in the document 1/1553-FAE 151 01 ("CLI Style"). All MO commands can also be performed using moshell's own MO commands.

It is possible to pipe a shell command to any external unix utility, eg "grep", "sort", etc. The pipe sign must be surrounded by a blank space on each side. It is also possible to use the built-in COMCLI command "filter", in that case no spaces shall be around the pipe sign. See examples below.

Each command line is sent in a separate ssh session, so in order to send several commands within the same ssh session, they need to be run on the same line by separating each command with a semicolon.

Examples:

- `ls -l /d/loadmodules | sort -nk 3 -` to sort files by size
- `ps -ef | grep com-` to see the list of com processes
- `find /bin -ls` - recursive list all files and directories in /bin
- `bash && for file in /bin/*; do echo $file ; done ; exit` - open a bash shell and do a for loop on all files inside the /bin directory
- `c+ -` to switch to comcli shell
- `show ManagedElement=1,Equipment=1,RbsUnit=1 ; configure ; ManagedElement=1,Equipment=1,RbsUnit=1,userLabel="test" ; end`
- `show all | grep Schema`
- `show all|filter Schema`
- `c2` to switch back to linux shell

Refer to the moshell file for more information about the uservariables for COM nodes:

- `comcli`
- `linux_shell`
- `cliss`
- `comcli_columns`
- `comcli_timeout`
- `comcli_cfg`
- `comcli_model`
- `comcli_retry_maxtime`
- `comcli_retry_interval`
- `comcli_port`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- comcli\_mom

#### 4.3.12 mcc/lmcc <moGroup>|<moFilter>|<proxy(s)> <comcli commands(s)> [|<unix-cmds>]

Execute MO Context-sensitive COMCLI commands.

Execute a COMCLI command from within a specific MO.

Examples:

List all COMCLI commands applicable to all Router MOs and their children (the command ? or ^ can be used to list COMCLI commands)

- lmcc router= ?
- lmcc router= \t

Print more help on the ping command in the MO InterfaceIPv4=TNA

- mcc interfaceipv4=tna ping ?

Run the ping command from all AddressIPv4 MOs and pipe the output through grep

- mcc addressip ping --count 3 10.18.30.2 | grep transmitted

#### 4.3.13 bo[ar]/ba[swdpmu]/br[wdm]/be[0-50]/bp

Manage board groups that can be used for running COLI commands on multiple boards.

Syntax:

- bo[a] [r]
- ba[s]/br/bp <boardGroup> <boardLNH>|<boardGroup>|<boardType>
- baw[s]/brw <boardGroup> <swa> [  
<rncMod>]
- bam[s]/brm <boardGroup> <MO Group>
- bad/brd <boardGroup> <devType> [  
<rncMod>]
- bap <boardGroup> <pgm>
- bau <boardGroup> <rpuLabel>
- be[0-50] <boardGroup> <boardGroup>

The following board groups are always created by default after running any of the board commands (bo/ba/br/lh, etc) the first time:

- all contains all boards (MP/BP) as well as PiuDevices (d0000x) and SPMs (spx.lnh).
- allp contains all boards (MP/BP) but not PiuDevices and SPMs.
- allpd contains all boards (MP/BP) and PiuDevices but not SPMs.
- alld contains all PiuDevices and SPMs.
- mp contains all MP boards.
- bp contains all BP boards.
- coremp contains the core MP(s), one or two depending on the configuration.
- ommp contains the O&M MP(s), one or two depending on the configuration.
- sccmp contains the SCCP MP(s) (RNC/MGW only).
- tu contains TU boards.
- scx contains SCB and SXB boards.
- et contains ET boards.
- aal2ap, aal2nccadm, aal2cpsrc, aal2rh contain MP boards running the corresponding aal2 programs.

In RNC, the following board groups are also created by default, containing MPs/SPMs connected to the various RNC modules:

- mod[x]
- cc[x]

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `dc[x]`
- `pdr[x]`

In RBS, the following board groups are created by default: `rax`, `tx`, `ru`, `asc`.

In MGW the following board groups are created by default: `mesc`, `licdb`, `ch`, `stc`, `gra`, `imra`, `raa`, `msb`.

It is recommended to start by running the `bo` command (*board overview*) to view the available boards in the node.

The very first time `bo` is run on a node, it will take more time because it has to fetch data from the node. The following times, the existing data is shown again unless the `r` switch is used (command `bor`), in which case the data is fetched again from the node.

By default the `bo` command only shows slots that contain boards defined in the configuration (i.e. boards which are associated with a PlugInUnit MO) but by using option `a` it is possible to view all slots, even those which do not contain a PlugInUnit.

The `ba` command is used for adding boards into a group. The boards shall be identified by their position or a string matching the board type. The `s` switch adds any related SPMs to the board group.

Examples:

- `ba group1 1 2 4-8 114-119` - boards 000100, 000200, 000400 to 000800, 011400 to 011900 are added to *group1*
- `bas group1 223-226` - boards 022300 to 022600 are added to *group1*, together with their related SPs (if these boards are SPBs).
- `ba group1 spb scb 3 4` - all boards of type matching **spb** and **scb** are added to *group1* as well as boards 000300 and 000400
- `ba group1 coremp mod scb` - all boards belonging to the groups **coremp** and **mod** are added to *group1* as well as boards with type matching **scb**.
- `ba gpb gpb` - all boards of type matching **gpb** are adding to the group called *gpb*.

The `baw` command is similar to the `ba` command except that the boards are identified by their SwAllocation and/or RncModule.

Examples:

- `baw moduleMPs module` - all boards belonging to the SwAllocation matching **module** will be added in the group called *moduleMPs*
- `baw module1 .* 1` - all boards belonging to rncModule 1 will be added to the group called *module1*

The `bad` command is for adding SPMs into a board group based on their device type. This is applicable to RNC only. Examples:

- `bad dc dc` - add all SPMs handling a DC device to the group called *dc*.
- `bad dc1 dc 1` - add all SPMs handling a DC device on module 1 to the group called *dc1*.

The `bap` command is for adding boards into a board group based on what program(s) they are running. The string is matched against the name of the Program MOs. All PlugInUnit MOs which contain a Program MO whose name matches the string are added to the group. Examples:

- `bap mesc cxc1324881|upcf_*mesc` - add all boards containing a Program MO whose name matches `cxc1324881` or `upcf_*mesc` into the board group *mesc*

The `bau` command is for adding boards into a board group based on what RPU(s) they are running. The string is matched against the reliableProgramLabel of the RPU MOs. All PlugInUnit MOs that have a RPU MO whose reliableProgramLabel matches the string are added to the group. Examples:

- `bau ranap rnc_ranap` - add all boards that have a RPU whose reliableProgramLabel match the word `rnc_ranap` into the board group *ranap*

The `bam/brm` command is for adding/removing boards corresponding the PlugInUnit/Spm/PiuDevice MOs of an existing MO group. When specifying the "s" option ("bams") the children Spm/PiuDevice of the PlugInUnit will be included in the board group. Examples:

- `st plug dis` - makes a MO group called *st\_group* containing the disabled PlugInUnit MOs
- `bam boardsdown st_group` - makes a board group containing the boards corresponding the disabled PlugInUnits found above

The `br` command is for removing a whole group or certain boards out of a group.

Negative filter (!) is supported in order to remove all boards except those matching the filter.

Examples:

- `br group1 1 3 gpb` - boards 000100, 000300 and all boards of type matching **gpb** are removed from *group1*
- `br group2` - *group2* is removed

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `br group1 !gpb` - all boards are removed from the group *group1* except those of type matching **gpb**

The `brw` command is similar to the `br` command except that the boards are identified by their `SwAllocation` and/or `RncModule`.  
Examples:

- `brw group3 dc` - all boards whose `SwAllocation` matches **dc** will be removed from *group3*

The `bp` command is for printing existing groups or the contents of a particular group. Examples:

- `bp` - all existing groups are shown, eg: *group1* and *all*
- `bp all` - the contents of the group *all* is shown.

The `be` command is for extracting a number of boards from a group. To be used in conjunction with "mon" to handle board groups that contain more than 50 boards (the current limit on target monitor). Examples:

- `be10 partial_mod mod`
- `be20 partial_dc dc`
- `partial_mod_dc partial_mod partial_dc`
- `mon partial_mod_dc`

Once the group is created, the `lh` command is used to run an OSE command on all boards of the group. See help of the "lh" command in next chapter for more info.

It is also possible to run MO commands on board groups. In this case, the MO command will execute against the MOs connected to these boards, ie the **PlugInUnit** or `Spm` MOs.

Examples:

- `acc mod1 restart` - restart the MP found in board group "mod1"
- `st mp` view state of all MP `PlugInUnits`
- `acc cc1 restart` restart the `SPMs` found in board group "cc1"

Note: in RNC, the `bo` command also creates a number of default MO groups `ccXdev`, `dcXdev`, `pdrXdev`, where `X` is the module number.

So, in order to lock/unlock some devices, use the MO group instead of the board group since the board group connects to the `SPMs` which don't have an administrative state.

For example, `bl cc1dev`.

More info in `h syntax`.

#### 4.3.14 `lh[z] <boardGroup>|<moGroup> <OSE-command>|run <commandfile> [ | <unix-cmds>]`

Run COLI commands on all boards of a board group or MO group.

The first argument of the "lh" command is the board group or MO group.

- Board groups are made by default after running the commands `lh`, `bo`, or `bp`. Type "bp" to see the list of board groups and "bp <boardgroup>" to see what boards are inside a group. The user can also define their own board groups with the command "ba". Type "h ba" for info.
- MO groups are made with the commands "ma", "st", or "hget", type "h ma" and "h syntax" for info. If the MO group contains MOs of type `PlugInUnit`/`Spm`/`PiuDevice` then it can be used with the "lh" command.

The second argument is the COLI command or list of COLI commands. If several COLI commands will be run on the boards they can either be separated by semicolons or run from a commandfile stored on the workstation.

Option:

- `z`: transfers the printout in gzipped format. Can save time on very large printouts. Only applies when the number of commands to send is greater than `fast_lh_threshold`. See more information further down.

Examples:

- `lh group1 te log read`
- `lh group1 te log read | grep ERROR:`
- `lh all vii` OR: `all vii`
- `lh all te log read | grep ERROR: OR:all err` (special shortcut)
- `lh z all ps -w`



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

It is possible to send multiple commands to each board of the board group by separating them with semicolons. Examples:

- `lh mp te log read ; llog -l ; te log clear ; llog -c`
- `lh dc te e trace1 SP_HIST ; te log read`

It is also possible to send a command file to each board of the group. Example:

- `lh spb run sp_traces.txt`
- `lhz all ps -w; rld -a`

When many commands are to be sent, the `lh` function will put them into a command file, transfer that file to the node (using (s)ftp) and run that file from within the node, using the `shell -f` command. This will save a lot of time instead of having to send each command one by one to the node.

There is a user variable called `fast_lh_threshold` which decides the number of commands above which a command file will be transferred to the node. See Section 2.5 and `moshell` file for more info about user variables.

#### 4.3.15 mon/monu/mond/monf/mon?/mon- [<board(s)|<boardGroup(s)>] [</path/to/logfile.pcap>]

Start/stop/check target monitor session in TCP or UDP mode.

The "mon" command issues a set of OSE shell commands ("secmode" and "tm") in order to open TCP or UDP ports from the boards on the node to the client. The boards address and/or board groups are given as argument to specify which boards the client will connect to.

Options:

- **mon:** start the monitor in TCP mode. In TCP mode, only one session can be connected to a specific board and the "mon" command must be run before each time a new monitor client will be started. Note: If the target monitor is already running on the node in UDP or DISK mode, then the existing mode is kept and the respective monitor client command will be shown.
- **monu:** start the monitor in UDP mode with router/viewer or router/capture. By default router/viewer will be used, unless the path to .pcap logfile has been specified, in which case router/capture will be used instead, and the output will be saved in a pcap logfile instead of displayed on screen. In UDP mode, unlimited number of sessions can be connected to the same board(s). The "monu" command does not have to be run again if a handle is already open to the board(s) that will be monitored.
- **mond:** start the monitor in UDP mode with dispatcher/monitor. Currently assumes the dispatcher is already running and starts only the monitor. Currently only supported with dispatcher/monitor installed in clearcase.
- **monf:** start the monitor in DISK mode. The output will be saved locally on the hard disk of the node, the path will be shown in the monf printout.
- **mon-:** close all monitoring handles on the node.
- **mon?:** print the target monitor status (TCP or UDP) and list of monitored boards.

Arguments:

- **board(s) /boardgroup(s):** the list of board or board groups to monitor. See examples below.
- **path to pcap logfile:** only applicable with option "u" ("monu"). The output will be saved to .pcap file instead of displayed on screen. The filename must always have the extension ".pcap"

Examples:

- `mon 000800 000900` : open a monitoring session in TCP mode to the boards 000800 and 000900
- `monu mod tu` : open a monitoring session in UDP mode with router/viewer to the boards belonging to board groups "mod" and "tu"
- `monu 000100 $logdir/board01.pcap` : open a monitoring session in UDP mode with router/capture to the board 000100 and save to a file called board01.pcap
- `mond mod tu` : open a monitoring session in UDP mode with dispatcher/monitor to the boards belonging to board groups "mod" and "tu"
- `monf mp et` : open a monitoring session in DISK mode for all boards in the groups "mp" and "et"
- `mon-` : close all monitoring sessions
- `mon?` : print monitor sessions

The command to start the monitor client is printed on the screen and also it is stored in the scripting variable `$moncommand`.

It is usually better to run the monitor client in a separate window than the moshell window but if this is not possible then it is also possible to run it from the moshell prompt, either in foreground or in background, eg:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `l $moncommand`
- `l $moncommand > $logdir/mylogfile &`

The second method is especially useful when running moshell scripts. Then it is possible to put a `wait` statement while monitor is storing the traces in the logfile, then kill the monitor process using the command `l kill -9 $background_pid`, since the PID of the background process has been automatically stored in the variable `$background_pid`.

More information about the target monitor: 6/15518-CRX10201/1

#### 4.3.16 `sql+/sql-/sql? [<heap>]`

Start/stop/check the SQL client on the node (CXC1325608).

This command checks if the SQL client LM is already loaded or started. If not, it finds the latest version stored on disk, loads it and starts it. The argument can be used to specify a different heap size than the one that is pre-compiled in the LM. The heap size given will be in MB, should be an integer between 1 and 50. If no heap is given, the LM is loaded without specifying any heap value which means that it will use the default heap size that is specified in the LM.

Warning: This command should be used for trouble-shooting purposes only. SQL commands must be entered with great care since they can cause a crash on the node when printing SQL tables that are very large.

#### 4.3.17 `pgu[c][f][r] [-p <board1,board2,...>] /path/to/newLM [<cvcomment>]`

Program Upgrade. For lab use only, eg, to load black LMs.

##### Arguments

- the first argument specifies the location of the loadmodule on the workstation.
- the second argument (optional) is a text string that will be given as comment when making the PGU CV.

##### Options

- `-p <boardlist>`: Restrict the program upgrade to the specified boards

##### Options

- `c`: causes `pgu` to skip the CV and restart part. This is useful when many LMs need to be upgraded, avoids having heaps of CVs and restart.
- `f`: causes `pgu` to skip the confirmation questions.
- `r`: uses the same method as in system upgrade. A temp CV is made and old programs are removed after restart. Useful for core programs such as `basic_OS`.

##### Examples

- `pgu /home/userid/CXC132456_R1A02` - will update an LM that has a Program MO
- `pgu /home/userid/CXC132789_R1B03.jar` - will update an LM that is in the JVM classpath
- `pgu /home/eric/blackLMs/CXP9010472%1-R4C98` - will update a RBS DSP container LM
- `pgu /vobs/mmgw/r5/mgwr5_tc_mesc/build/mesc.ppc@@/main/1lv21_corr/58` - the CXC number of this LM will be found automatically by `pgu` through the `bswhat` command
- `pgu /home/userid/CXC12345678%2_R99A01 Black LM for HL12345` - will upgrade the program CXC12345678
- `pgu -p 001200,001300 /home/userid/CXC1320787_P90A01 Black system manager` - Will upgrade the program CXC1320787 only on boards 001200 and 001300, and include a CV comment.

The programs that are connected to an LM with the same product number will be identified and shown to the user. If the user confirms to go ahead, a loadmodule connected to the new LM will be created (if there isn't already one on the node) and the programs using the old LM will be deleted and recreated towards the new LM. Then a cv is made and the user is prompted to restart the node so that the change will take effect.

##### Options

- `c`: causes `pgu` to skip the CV and restart part. This is useful when many LMs need to be upgraded, avoids having heaps of CVs and restart.
- `f`: causes `pgu` to skip the confirmation questions.
- `r`: uses the same method as in system upgrade. A temp CV is made and old programs are removed after restart. Useful for core programs such as `basic_OS`.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

This procedure is "cleaner" than hacking the sql tables since it uses the MO interface.

This command is implemented as an moshell script, the script can be seen in **moshell/commonjars/scripts/pgu.mos**.

This shows that the user can define any new commands they need by adding an alias in the .moshellrc which will point to a script file. The parameters to the command will be sent to the script file via the positional parameters \$1, \$2, \$3, etc. The variable \$0 will be set to the whole line contents. See Section 4.1.32 for info.

#### 4.3.18 procload/proctemp [<unix-cmds>]

Print processor load or temperature.

The procload command is a wrapper for the COLI command `capi` (`capi tot` and `capi core all`) The proctemp command is a wrapper for the COLI command `boardtemp all` or `temprt` It can be run on individual boards or board groups.

Examples:

- `procload`: Check processor load on the current board
- `lhsh 001900 procload`: Check processor load on another board
- `lh mp procload`: Check processor load on all boards of a board group
- `lh all procload`: Check processor load on all boards of the node
- `lh allp proctemp`: Check temperature on all boards of the node

The output of `procload` shows the total processor load of each processor in the board, followed by the processor load of the individual cores of each processor.

- Example output on a GPB75: `72.3 (76.8 68.1)` The first number is the total load of the Main Processor, the next two numbers in brackets are the individual load of the two cores.
- Example output on a EPB1:  
`12.2 ( 6.2 11.9 7.1 17.0 5.5 28.0 9.4) 25.5 ( 3.3 29.3 36.6 24.7 26.9 28.1 29.6)` The first number is the total load of the primary Processor, then the individual load of the cores of that processor, then the total load of the secondary processor, then the individual load of the cores of that processor. Note: on EPB1, the load of the second core (ip bare metal) cannot be shown (it is usually 100)

The output of `proctemp` shows the temperature displayed by each sensor of the board as well as the alarm limit for each sensor. The alarm limit for each sensor is displayed in brackets next to each sensor's temperature. All temperature values are in Celcius.

#### 4.3.19 proglis/progkill [-e] [<string>] [<unix-cmds>]

List or restart programs on boards or board groups.

These two commands are wrappers for the COLI commands `pglist/pm_pginfo` and `pgkill/pm_kill`.

The proglis command lists the OSE programs running on MP, BP, or SP processor. The program handle, state, program number and name are shown. With option `-e`, the heap and pool size are also shown. It is also possible to add a filter after the `-e` option in order to show other program settings apart from the heap and pool. Eg: `proglis -e stack` shows all program settings matching the word `stack`. A list of all available program settings is shown at the end of the printout (when option `-e` is used).

The progkill command is used for restarting an individual program on a MP. If this command is used on a BP or SP then the whole processor restarts. A string which matches the program number or program name must be given as argument to the command. Eg: `progkill jvm`, or `progkill 0784`

Examples:

- `proglis`: List programs on central MP:
- `lhsh 001400 proglis | grep 0787`: List programs on board 001400, whose product number match "0787"
- `lh all proglis -e`: List programs on all MP/BP/SP processors together with heap and pool info
- `lh ommp proglis -e stack_size$`: Show the stack size for all programs running in the O&M MPs
- `lh mp proglis | grep system`: List programs on all boards of the boardgroup "mp", whose name match "system"
- `lhsh 001400 progkill 2417`: Restart the program whose product number matches "2417" on board 001400
- `lh mp progkill aal2ap`: Restart the programs whose name matches "aal2ap" on all boards of the boardgroup "mp"

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.20 `fte <te-command> [<trace-groups>|all] [<string>] [!<unix-cmds>]`

Filtered trace and error command.

This command is a wrapper for the COLI commands `te` and works in two ways:

- `fte s [all]` : to print trace status.  
Without the option `all`, only the trace objects and processes who have additional trace conditions are shown (supercedes `cabe` command).  
With option `all`, all traces objects and processes are shown, even those whose trace conditions are default. To filter this printout, pipe it to `grep`.
- `fte <subcommand> [<trace-groups>|all] [<string>]`  
Any `te` subcommand can be specified, e.g. `e` (enable), `save`, `config`, `default`, `preset`, etc.  
The string is matched against all processes and trace objects in that board and a list of `te` commands are run against every matching process.  
The pattern matching follows regular expression syntax and is non-case sensitive.

Examples:

- `lh all fte s` : List all non-default trace conditions in all boards of the node
- `lh mp fte s all | grep -i ose` : List all trace conditions in boards that belong to the board-group `mp` and pipe the output to `grep` lines matching the word `ose`
- `lhsh 000800 fte e all dsp.*meas` : Enable all traces on processes and trace objects that match the string `dsp.*meas` on board 000800
- `lh mod fte config -run bus_send bus_receive rrc|nbap|ranap` : Enable and save `bus_send` and `bus_receive` on trace objects and processes matching the string `nbap|rrc|ranap` on all boards of board group `mod`

#### 4.3.21 `goxb[ib] [-p <advpw>] <commands> [!<unix-cmds>]`

Run XB commands as advanced user (CMXB/SCXB)

Running commands as basic user is already supported with regular COLI commands, type `h coli` for info.

The `goxb` command is primarily intended to run XB commands as advanced user, by giving the advanced password from the `-p` option.

However if the password has not been specified the `goxb` command will use the basic mode.

One or more XB commands can be run, each command to be separated by semicolons.

Syntax:

- `lhsh <board> goxb[ib] [-p <advpw>] <command>[;<command>;...] [!<unix-cmds>] or:`
- `lh <bdgroup> goxb[ib] [-p <advpw>] <command>[;<command>;...] [!<unix-cmds>]`

Options:

- `i` : for running ISS/IMISH commands, eg: `goxbi`
- `b` : for running BCM commands, eg: `goxbb`
- `-p <password>` : to specify the password of the advanced user on XB board. If the password is not given then basic mode is used.

Examples:

- `lhsh 000300 goxb -p secret listsw current`
- `lh xb goxb -p secret listsw current; find /bin -ls | grep CX`
- `lhsh 002600 goxbi -p secret show interfaces status; show mac-address-table`
- `lh cmxb goxbi -p secret iss ; show interfaces status; show mac-address-table`
- `lh cmxb goxbi -p secret sh version`
- `lhsh 002600 goxbb -p secret show counters`

Note: the `goxb` command uses a script written in expect and assumes the path to be `/usr/bin/expect`. If expect is installed somewhere else, the path can be specified in the moshell uservariable called "expect".

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.22 `ftree[f][d][1] [-lnh>][<directory>] [| <unix-cmds>]`

Recursive listing of a directory on the file system of the node or the workstation.

This command first checks if the directory exists on the workstation and if yes, its contents is printed from the workstation. Otherwise the command logs into the OSE shell of the node, checks if the directory exists on the node and prints its contents.

The directory can be entered either as a relative path or an absolute path.

To list a directory on a different board, enter the linkhandler of the board followed by the absolute path.

If no directory is entered then the current working directory will be listed.

The option `d` is for printing the subdirectories only.

The option `f` is for printing files only. Mainly useful on `/c/pmd` for increased speed and sorting by timestamp.

The option `1` is for non-recursive listing of a folder, same as `ls -l`.

It is possible to pipe the output to a unix command for filtering/sorting purposes.

Examples:

1. `ftree /home/myuserid/moshell` - list all files and directories from the local folder `/home/myuserid/moshell` on the workstation.
2. `ftree /c/loadmodules_norepl` - list all files and directories from the folder `/c/loadmodules_norepl` on the node.
3. `ftree /c/loadmodules_norepl | grep xml` - all files/directories matching `xml` are listed
4. `ftree 001900/f` - all files and directories on the `/f` drive on board 001900 are listed
5. `ftreef /c/pmd` - list all PMD files, sorted by date and time
6. `ftreed /var` - list all the subdirectories under `/var` in the PRBS node

It is possible to run `ftree` on several boards by using the `lh` command.

Example:

```
ba bp 0-9999 # put all boards in a group called "bp"
br bp gpb    # remove all gpb boards from the "bp" group
lh bp ftree /f # recursive listing of the /f drive on all bp boards
```

#### 4.3.23 `ftget[c]/ftput[c]/ftdel[a] [<options>] <source>[/*] [<destination>]`

Transfer files or directories to/from the node, using `ftp` or `sftp`.

##### Syntax:

- `ftput[c] [<options>] localfile/localdir[/*] [remotefile/remotedir]`
- `ftget[c] [<options>] remotefile/remotedir[/*] [localfile/localdir]`
- `ftdel[a] [<options>] remotefile/remotedir`

Where "local" refers to the workstation and "remote" refers to the CPP node.

It is possible to transfer a whole directory to/from the node by specifying a source directory instead of a source file.

A subdirectory with the same name as the source directory will then be created under the destination folder, unless a asterisk is specified at the end of the source folder, eg:

```
ftget /d/usr/*
```

To fetch the contents of a folder located on a local volume of the node, the path must be preceded by the board address following by exclamation mark.

Eg: `ftget 000900!/d/systemfiles`

Note that in this case, only the files directly located under the folder will be collected and not the contents of the subfolders.

The `ftdel` command removes an individual file or a set of files/directories inside a directory. With "a" option, also the directory itself will be removed.

The `c` option in `ftget/ftput` stands for conditional and means that if the file(s) already exists on the workstation/node, they will not be overwritten.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The `a` option in `ftdel` means that also the remote directory itself will be deleted. Otherwise only its contents will be deleted.

The options can be placed anywhere on the command line but the source must be given before the destination.

If the destination is omitted then the current working directory is chosen.

#### **Filtering option:**

The `-f` option allows to specify a regular expression to only transfer the files matching that expression. See examples below.

Exclamation mark `!` can be used as negative filter, meaning that any files that do NOT match the filter will be transferred, eg.

```
ftget -f !tmp /c/usr
```

#### **Time options:**

The `-s` and `-e` options are used for specifying an absolute timespan: `-s` gives the starting date and `-e` gives the ending date.

The format is `yyyymmdd[.hhmm]`, for instance 20071230, or 20071230.0800.

The `-m` and `-p` options are used for specifying a timespan relative to today's date: `-m` gives how long time backward and `-p` gives how long time forward. The format is in days, hours, or minutes, eg. 10d (10 days), 2h (2 hours), 30m (30 minutes).

#### **Examples:**

- `ftget /c/logfiles` - fetch the folder `/c/logfiles` and all of its contents and store it in the current working directory on the workstation
- `ftget /c/logfiles /home/eric` - fetch the folder `/c/logfiles` and all of its contents and store it in the folder `/home/eric` on the workstation
- `ftputc /home/eric/rnc10/configuration/cv/cv-10 /d/configuration/cv` - put the folder `cv-10` from the workstation to the node, files that already exist on the node will not be overwritten (conditional transfer)
- `ftget -f A.*xml.gz -m 3h /c/pm_data /home/eric/rnc10` - download files from the folder `/c/pm_data` whose path matches the string `A.*xml.gz` and whose timestamp is from the last 3 hours.
- `ftget -f !(.xml.gz|.bin)$ /c /home/eric/backup` - download all files from the folder `/c` except those whose path match `.xml.gz` or `.bin`
- `ftget /c/loadmodules_norepl /home/eric/rnc10 -f (xml|jar)$` - fetch all files matching who have the extension `xml` or `jar` from the folder `/c/loadmodules_norepl` and any of its subfolders.
- `ftdel /c/pm_data -f .tmp$ -m 900 -p 870` - delete files with the extension `.tmp` in the folder `/c/pm_data`, with timestamp older than 30 days.
- `ftput /home/eric/backup/* /d/loadmodules` - transfer all the files and folders under `/home/eric/backup` to `/d/loadmodules` without making the folder called "backup"
- `ftget 000900!/d/systemfiles /home/eric/rnc10` - transfer the contents of the local volume `/d/systemfiles` from board 000900 to `/home/eric/rnc10`

#### **Notes:**

- By running `ftput/ftget` from **mobatch**, it is possible to get/put files to/from many nodes in parallel.
- Only active mode is supported for unsecure ftp. If active mode is not allowed by the firewall, then use *secure\_ftp* instead (see Section 2.6 for info about *secure\_shell/secure\_ftp*).
- By default, the original file timestamps and permissions are NOT preserved after transfer. In order to preserve them, it is necessary to set `uv ftp_preserveprops` to 1 (get), 2 (put), or 3 (get and put).

#### **4.3.24 htget <remote file> [<local file/local dir>]**

Download files using http or https.

This command allows to download a file to the workstation using http or https. Whether http or https will be used is decided from the user variable `secure_http` (0=http, 1=https).

The *remote file* can be specified with or without ipaddress.

If no ipaddress is specified then the file is fetched from the current node. If an ipaddress is specified the file can be fetched from a different server. It is also possible to specify the tcp port after the ip address eg `<ipaddress>:<port>`

The prefix `http://` or `https://` is optional.

Examples:

- `htget /cello/oe/xml/rnc_node_mim.xml` - file will be transferred to `./rnc_node_mim.xml`
- `htget 10.1.128.17/cello/oe/xml/rnc_node_mim.xml ~/rnc_mom.xml` - file will be transferred to

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
/rnc_mom.xml
```

- `htget http://10.1.128.17/cello/oe/xml/rnc_node_mim.xml /home/eratoto-` file will be transfered to `/home/eratoto/rnc_node_mim.xml`
- `htget http://10.68.110.44:8080/em/index.html`
- `htget https://10.68.110.44:443/em/index.html`

#### 4.3.25 edit <remote file>

Edit a file on the node.

Moshell will download the file, spawn your editor allowing you to edit it and then upload it once you have finished. If the file does not exist on the node it will be created.

The editor is specified in the `editor` user variable. By default, `editor` is set to the bash environment variable `$EDITOR`. If `$EDITOR` is not set, then `vi` will be used. To use another editor, add the following line in the `~/.moshellrc` file:  
`editor=/path/to/your/favorite/editor`

#### 4.3.26 fclean[f|ff|a|d|e] [<Inh>][<directory>] [-f <filename-filter>]

Removal of obsolete loadmodules OR recursive removal of a directory on the node.

1. Remove a directory (equivalent to "rm -Rf" in unix). Syntax:

- `fclean <directory>` to remove all contents in the directory but not the directory itself
- `fclean <directory> -f <filename-filter>` to remove part of the directory, ie, only files whose name matches the filter
- `fcleana <directory>` to remove all contents in the directory as well as the directory itself
- `fcleand <directory>` to remove all empty folders in the directory. Useful to clean up the `/c/pmd` and `/c/loadmodules_norepl` which often contain a number of empty directories.
- `fcleane <directory>` to empty all files in the directory (files are replaced with an empty file). This is mainly intended to clear the logfiles in `/c/logfiles`. For STP use only !

Note: The "d" and "e" options cannot be used together !

2. Examples (all files in that directory will get removed (after confirmation from the user):

- `fclean /c/pmd`
- `fclean /c/pmd -f 0x00`
- `fclean /p001200/pm`
- `fclean 001900/f`

3. All empty folders in those directories will get removed (after confirmation from the user).

- `fcleand /c/loadmodules_norepl`
- `fcleand /c/pmd`

4. All files in that directory are removed and replaced with empty files.

- `fcleane /c/logfiles`

5. Deleting loadmodules that are not in use by the system. Examples:

- `fclean`
- `fcleanf`
- `fcleanff`

This means any loadmodules that are present on the disk but are not defined in the current cv (*ARMAMENT* file and database) nor are listed in any of the existing upgrade packages on the node.

The directories which are "cleaned up" are:

- on the central MP:
  1. `/c/loadmodules`
  2. `/c/loadmodules_norepl`



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

3. `/c/java`
  4. `/c/dsp_load_files`
  5. `/c/fpga_load_files`,
- on all MPs: `/d/loadmodules`
  - on all BPs:
    1. `/f/loadmodules`
    2. `/f/satloadmodules`
    3. `/f/dsp_load_files`
    4. `/f/fpga_load_files`
    5. `/f/dp_loadmodules`

It is possible to specify a list of loadmodules that are not to be cleaned up in the variable `keepLmList` in the **moshell** file. When the command has completed, a command file containing all remove commands is generated and the user is prompted with the choice to run it at once or at a later time.

Options (`f` and `ff`):

- `f` : to clean the `/f` drives. Only files that are in use by the current cv will be left on the `/f` drives.
- `ff` : to remove all loadmodules from the `/f` drives, including those that are in use by the current cv. This will force the device boards to refetch their software from the `/c/loadmodules` directory.

Note: the `fclean` command will also clean up the `/f` drives but will leave all loadmodules that are specified in other upgrade packages as well as the current one. So the cleanup is not as drastic with **fclean** as with **fcleanf** or **fcleanff**.

#### 4.3.27 `hi [<commandFilter>], !<commandNr>`

Print history of moshell commands entered during the current session.

By using the filter, it will only show those command matching that pattern. Example:

```
moshell> hi
1 lt e1
2 st all dis
3 get 4 oper
```

To rerun e.g. command number 2, do:

```
!2
```

#### 4.3.28 `time[t] <command>|<logfile>`

Measure time taken by an moshell command or by each command in a moshell command file.

Arguments:

- `<command>` : the command is executed and its duration, in seconds, is shown on the last line of the printout. The duration is also saved in a moshell scripting variable called `$duration`. If the "t" option was specified ("timet" command), the duration is also saved in a hashtable called `$durationtable`, where the index is the command executed.
- `<logfile>` : the logfile is parsed by the time function and the duration of each command contained in this log is displayed as output. The logfile can be in gzipped format.

Example:

- `time get all` - Measure the time taken by the "get all" command
- `timet lh all te log read` - Measure the time taken by the "lh all te log read" command and save it in a hashtable
- `time /path/to/NODE_dcg_m.log.gz` - Show the duration of each command contained in the logfile of the "dcg" command:

#### 4.3.29 `lmid[c][h]/upid[om] <pattern>|refresh`

Print translation of loadmodule/upgradepackage product number or T&E error codes.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Examples:

- `lmid 2517` - to lookup the name of any LM whose product number matches 2517
- `upid 2014` - to lookup the release and cpp version of the UP whose product number matches 2014
- `lmid aal2` - to lookup the product number(s) of any LM(s) whose name matches aal2

In addition, the LM name is printed beside the product number when using certain OSE shell commands:

- `ls`, `pglist` and `ps` for uservariable `print_lmid=1`
- same as above, plus `te log read` for uservariable `print_lmid=2`

For instance, when printing the contents of a directory such as `/d/loadmodules` or `/c/loadmodules_norepl`, the LM name will appear next to the product number. This functionality can be disabled by setting the user variable `print_lmid` to 0.

See Section 2.5 and Section 2.6 for more info about user variables.

If some names are missing from the printout, just run `lmid refresh` or `bo` and it will update the moshell LM reference file with all missing LM names. The refresh also happens automatically if no LM name is found for the pattern given.

The "c" option in `lmid` is to print the error codes list (aal2/mtp3/sccp/utrancell) which is used to decode error codes from the "te log read".

The "h" option in `lmid` is to print the HW translation table.

The "m" option in `upid` is to print the CXP to MOM version table.

The "o" option in `upid` is to print an overview, mapping of major SW releases only.

#### 4.3.30 p/w/pw/b

Change moshell prompt and/or window title.

Examples:

- `p <newPrompt>` - to change the prompt. Can also be changed before startup with the uservariable `prompt`, eg `moshell -v prompt=xxx <node>`. The prompt can contain a carriage return, use the `\n` sign, eg `moshell -v prompt='moshell\nNode'`, or `p moshell\nNode`
- `w <newWindowTitle>` - to change the window title
- `pw <newText>` - to change the prompt and window title
- `b` - to make the prompt bold or unbold. When the prompt is in bold, command lines that are longer than the screen width do not wrap correctly.

#### 4.3.31 prox[+-]

Toggle display of proxy identities in printout of `get <mo> <attribute>` command.

To print the proxy identities, type `prox` and the proxys will be printed for the remaining of the session. By typing `prox` one more time, the proxys will not be printed anymore.

Options:

- `+` : activate the feature, proxy identities will be printed at the beginning of the line (`get_format=1`)
- `-` : return the feature to its original value, proxy identities will not be printed (`get_format=0`)

The default behaviour is to not show the proxy identities (`get_format=0`) To change the default behaviour, it is possible to use the uservariable `get_format`, eg, adding the line `get_format=1` in the file `/.moshellrc`.

#### 4.3.32 col

Toggle display of colors.

By default, some lines will appear in different color in the `te log read` and `cabrd` printouts.

To disable the coloring, type `col` once; to reenale it, type `col` again.

This setting can be saved permanently in the uservariable `show_colors`.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.33 ul

Toggle display of *userLabel* in `st/1st` and `pget/lpget` printouts.

By default, the *userLabel* is not shown.

Type `ul` to display this information for the remaining of the session. Type `ul` again to hide it.

This setting can be saved permanently in the uservariable `show_userlabel`.

#### 4.3.34 conf[bld][+-]

Toggle confirmation on various MO commands.

- `confb - bl/deb.set/acc` commands
- `confl - lt/lc` commands
- `confbl - both` type of commands
- `confd - del/rdel` commands

By default, these commands require confirmation (y/n). Type one of these commands to disable confirmation. Type the command again and confirmation is re-enabled.

These settings can be saved permanently in the uservariables `bldebset_confirmation`, `lt_confirmation`, `del_confirmation`.

Options:

- `+` : disable confirmation, instead of toggling it (sets the uservariable to 0)
- `-` : return confirmation to its previous value.

#### 4.3.35 gs[+]/gsg[+-]

Toggle display of old/new attribute value in `setbldeb` commands.

Purpose:

- `gs` - old value is displayed before setting the attribute. I.e, a `get` is performed before the `set`.
- `gsg` - old value is displayed before setting the attribute, then new value is displayed after setting the attribute. I.e, a `get` is performed before and after the `set`.

Notes:

- The set will not take place if the new value is the same as the old value. The result **-No Change-** is printed.
- If the set is accepted (no exception given) but the final value is still the same as the old value, then the result **>> Fail** is printed.
- These toggles don't affect setting attributes whose value take up several lines such as *Array of MoRef* and *Struct*.

Options:

- `+` : activate the feature, instead of toggling it.
- `-` : return the feature to its previous value.

#### 4.3.36 ip2d <ip-address>

Convert an IP address to a decimal number.

This can be used for instance when editing an entry in the `ip fro` table in the `sql` database, where ip addresses are stored in decimal format.

Example:

- `ip2d 10.1.2.3` —> would print: 10.1.2.3: -4127129085

Similar to the `get` command, you can also store the output of this command into a variable. Example:

- `ip2d 10.1.2.3 > $ip_db` —> stores -4127129085 into variable `$ip_db`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.37 d2ip/h2ip <number>

Convert a decimal or hexadecimal number to an IP address.

This can be used for instance when decoding T&E traces or COLI printouts where ip addresses are written in decimal or hexadecimal format.

Examples:

```
$ lhsh 001400 drh_trbr_data iphost
ipHostFroid admState opState availStatus piuId smn apn ern atmPortId ipAddress linkHandlerName msgBoard
7 1 1 0 7 0 7 0 29 0xa641002 000700 hostReady applSciRunning

$ lhsh 000200 te log read
[06:52:40.228] Ipet_app3dr_proc(IPET_APP3DR_SH_OBJ) app3dr_sh.c:5692 REC SIG:signo:190071 sender :100dc Rec: IPET_IPPSI_UDPSESSIONSETUP_IND. Ipet_remoteIpAddress = 174329861
```

- `h2ip 0xa641002` → will print: `0xa641002: 10.100.16.2`
- `d2ip 174329861` → will print: `174329861: 10.100.16.5`

Similar to the `get` command, you can also store the output of this command into a variable. Example:

- `d2ip 174329861 > $ip` → stores 10.100.16.5 into variable `$ip`
- `h2ip 0xa641002 > $ip` → stores 10.100.16.2 into variable `$ip`

#### 4.3.38 h2d/d2h <number>

Convert an integer to hexadecimal or viceversa.

Example:

- `d2h 10` would return `0xA`
- `h2d a` would return `10`
- `h2d 0xa` would also return `10`

Similar to the `get` command, you can also store the output of this command into a variable. Example:

- `d2h 10 > $var` - stores `0xA` into variable `$var`
- `h2d a > $var` - stores `10` into variable `$var`

#### 4.3.39 h2b/b2h <number>

Convert a binary to hexadecimal or viceversa.

Example:

- `b2h 101011` would return `2B`
- `h2b a` would return `1010`
- `h2d 0xa` would also return `1010`

Similar to the `get` command, you can also store the output of this command into a variable

Example:

- `b2h 10100011 > $var` stores the value "A3" into variable `$var`
- `h2b a > $var` stores 1010 into variable `$var`

#### 4.3.40 encpw <password>

Create an encoded password for use in Moshell

Encoded passwords will be prefixed by "ENC?". Make sure to keep this prefix when using the password

Encoded passwords can be used in `moshellrc`, `ipdatabase` files, scripts, on the command line and as input to MO creation

Example:

- `encpw SecretSLSPassword`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.3.41 mos2ro <moshell.zip>

Generate an installation package for moshell read-only version.

Example:

- `mos2ro /home/userid/moshell10.0v.zip` - generate a file called `moshellreadonly10.0v.zip` which can be used to install a read-only version of moshell

The following operations are blocked in the read-only version: MO and Scanner create/delete/set/action. Also certain commands such as `mon`, `fro`, `sql`, `pgu`.

#### 4.3.42 wait <delay>|<newtime>

Wait for a specific duration in hrs, mins, secs, rops, or until specified time.

Default duration is in seconds. Use "m" for minutes, "h" for hours and "r" for ROPs. Default ROP duration is 15 minutes but can be changed with the `uv rop_period` (eg: to change rop duration to 5 minutes, type `uv rop_period=5`)

Examples:

- `wait 2` - wait 2 seconds
- `wait 90` - wait 90 seconds
- `wait 3m` - wait 3 minutes
- `wait 4h` - wait 4 hours
- `wait 5r` - wait 5 ROP periods
- `wait 5r-60` - wait 5 ROP periods minus 60 seconds (the prompt will return 60 seconds before the end of the five ROPs)
- `wait 16:30` - wait until 16:30
- `wait 9:45:30` - wait until 9:45:30
- `wait 20130728.094530` - wait until 2013-07-28 at 09:45:30
- `wait 2013-07-28.09:45:30` - same as above

If a ROP period is already started, it will wait until it finishes, then wait additionally the number of ROP periods specified.

To wait only until the current ROP is finished, use the command "`wait 0r`"

If the new time is before the current time (e.g. if current time is 9:30 and new time is 8:00), then the waiting will continue to the next day at that time.

Note: the new time shall be given in same time zone as the workstation where moshell is running.

To abort a wait statement, just type `CTRL-C`

#### 4.3.43 return

To exit from a command file without exiting from MoShell (scripting).

Typically, the `return` command would be executed upon validation of a specific condition in the command file.

Note that the `return` cannot be put in a `for` loop, only in a `if/else` that is not contained in a `for` loop.

In order to use it in `for` loop, the `break` command has to be used in order to get out of the `for` loop, first.

In this example, we run a `restart` followed by running a `trun` script, 60 times.

If the `trun` script fails then we set the `$return` variable to 1 and break from the loop (note: the `$command_result` is a default variable which gets automatically set after running `trun`, see Section 6)

After the loop, there is a check to see if `$return` is 1, if yes, then "test failed" is printed, the commandfile is aborted and we are returned to the moshell prompt.

If `$return` was not set to 1, then it means we have completed all 60 iterations successfully and the final part of the script will be executed.

```
for 60
$return = 0
facc 0 manualrestart 0 0 0
trun $moshelldir/cmdfiles/rnc_commands.mos
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
if $command_result = 1
$return = 1
break
fi
done
if $return = 1
l echo "TEST FAILED"
return
fi
l echo "TEST OK"
```

Another example of using the return command can be found in [Section 6.5](#)

#### 4.3.44 print

To print a line or variable (scripting).

Handy when doing scripting to see what values will be substituted to a variable.

Examples:

```
print $var
print $table[$var]
```

#### 4.3.45 alias/unalias <alias> <command>

Print or define command aliases.

Example:

- `alias ter te log read`

Type `alias` on its own to list all defined aliases.

Use command "unalias" to undefine aliases. Example:

- `unalias ter`

Aliases can be stored permanently in the `~/.moshellrc` file, using the same syntax as above.

Note: moshell commands take precedence over aliases. This means that if an alias has the same name as an moshell command, the moshell command will be executed instead of the alias.

#### 4.3.46 If[c] <file>

Load a moshellrc file or a offline COLI file.

Examples:

- `lf /home/eraaldr/tools/moshell/jarxml/moshellrc` - Load a moshellrc file
- `lfc /home/moshell/moshell_logfiles/logs_moshell/dcg/RNC12/140101_1453/RNC12_dcg_m.log.gz` - Load a offline COLI file (only applicable in offline mode)

Note: The `If` command can be run from the moshell prompt or called from within a moshellrc file in order to source additional moshellrc files.

#### 4.3.47 bg[g]/bgs/bgw [<commands>|<id>|all] [<maxtime>]

Run some moshell commands in background or check status of background jobs.

Syntax:

- `bg[g] <commands>` : to execute some moshell commands in background. The "g" option is to gzip the log upon completion.
- `bgs` : check the status of all background jobs.
- `bgw [<id>|all] [<maxtime>]` : wait for one or all jobs to complete. Specifying `maxtime` (in seconds) will lead to stop waiting after a job has been running for longer than that amount of time.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The following variables and arrays keep track of the list of background jobs and their properties:

- `$bg_id` : the job id
- `$bg_pid[id]` : the PID of the moshell process corresponding to each background job
- `$bg_status[id]` : the status of each background job, RUNNING or FINISHED
- `$bg_log[id]` : the path of the logfile containing the printout of each background job
- `$bg_runtime[id]` : the current duration in seconds of each job

Examples:

- `bg dcgm` - Run some moshell commands in background
- `bgg cabx;al;str` - Run some moshell commands in background and gzip the logfile upon completion
- `bg l+ $logdir/$ipaddress_kget.log ; lt all ; kget ; l-` - Run some moshell commands in background using a user defined logfile
- `bgs` - Check status of background jobs
- `bgw 2` - Wait for job number 2 to complete
- `bgw` - Wait for all jobs to complete
- `bgw all 1800` - Wait for all jobs to complete but stop waiting if the running time of a job exceeds 1800 seconds (30 mins)

#### 4.3.48 - `smd[slcr] [-m <days>] [-s <size>] [-f <filter>] [-o a|s|n] [-u <user>|all] [-d <directory>] [-n <max>]`

Server Maintenance - disk usage

Options:

- `l`: list files
- `r`: remove files
- `c`: compress files (using gzip)

Switches:

- `-d <directory>`: the directory to process. Default= the `moshell_logfiles` directory.
- `-u <user>|all` : the user who owns the file. Default: if running as root, all users - otherwise, the current user.
- `-n <max>` : max number of files to list (eg: `-n 30`). Default=20.
- `-m <days>` : minimum age of the files in days (eg: `-m 10` ==> all files which were modified 10 days ago or more). Default=1.
- `-s <size>` : minimum file size in B/K/M/T/G (eg: `-s 100B` ==> all files of size at least 100 Bytes). Default=0.
- `-f <name>` : file name filter (eg: `-f A.*.xml.gz` ==> all files whose name file path `A.*.xml.gz`).
- `-o [a|s|n]` : printout order (eg: `-o a` ==> sort files/processes by age, `-o s` ==> sort by size, `-o n` ==> sort by name). Default=s

Examples - diagnostics with `smd`:

- `smds -n 15` : show disk usage summary of the log folder, max 15 largest files/directories displayed (default: 20)
- `smds -d /home/user/moshell` : show disk usage of the folder `/home/user/moshell` (default: the folder `/home/user/moshell_logfiles`)
- `smd -d /var/opt/ericsson` : show disk usage of the folder `/var/opt/ericsson` (default in amos: the folder `/var/opt/ericsson/amos/moshell_logfiles`)
- `smdl -m 30 -s 1M -f /logs_mobatch/*.log$ -o a` : show files of size at least 1M, aged at least 30 days, files located in `logs_mobatch` folder and file name ends with `.log`, sort printout by file age.
- `smdl -d /home/user/moshell_logfiles/logs_mobatch -m 10 -s 1M -o s` : show files in `/home/user/moshell_logfiles/logs_mobatch`, aged at least 10 days, size at least 1M, sort printout by file size

Examples - cleanup with `smd`:

- `smdr -m 30 -f /A.*.xml.gz$` : remove files in `moshell_logfiles`, aged at least 30 days, with file name matching `A.*.xml.gz`
- `smdr -m 30 -n _ropfiles.zip$` : remove files in `moshell_logfiles`, aged at least 30 days, with file name



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

matching\_ropfiles.zip

- `smdc -m 30 -o n`: compress logfiles aged at least 30 days, show file list sorted by filename (note: files already in compressed format will not be affected)
- `smdr -m 7 -f logs_moshell/(tempfiles|cache)/`: remove all the moshell tempfiles older than 7 days

Examples - cleanup with cronjobs:

Example of some cronjobs to remove ropfiles older than 30 days and compress logfiles older than 30 days (contents of "crontab -e" is shown). In this example the jobs are performed every saturday at 1:00 am.

```
# minute (0-59),
# | hour (0-23),
# | | day of the month (1-31),
# | | | month of the year (1-12),
# | | | | day of the week (0-6 with 0=Sunday).
# | | | | | command(s)
# | | | | |
00 01 * * * 6 find /home/user/moshell_logfiles -type f -name 'A*.l.xml.gz' -mtime +30 -exec rm -f {} \;
00 01 * * * 6 find /home/user/moshell_logfiles -type f -name '*_ropfiles.zip' -mtime +30 -exec rm -f {} \;
00 01 * * * 6 find /home/user/moshell_logfiles -type f -name '*.log' -mtime +30 -exec gzip -f {} \;
```

#### 4.3.49 q/by/exit/quit [<exitcode>]

Exit moshell.

Any of the following commands can be used to exit moshell: `q`, `by`, `exit`, `quit`

The exit command supports specifying an exit code, eg: `exit 1`

## 4.4 PM commands

### 4.4.1 pmom[acdpo]/lmom[c] [<moclass>] [<counter>] [<data-type>] [<flags>] [<description>]

Print description of PM counters (pmom) or log attributes (lmom, CDMA only).

Options:

- `a`: shows what regular attributes can be included in scanners.
- `c`: show all the MO classes specified in the filter as well as their children/grandchildren/etc classes.
- `d`: gives a shorter printout, without the description part.
- `p`: show only the definitions relating to platform MOs (CPP)
- `o`: show only the definitions relating to application MOs

The `type` field refers to the data type of the counter value, e.g. an integer (long), or a sequence of integers.

The `flags` field refers to the properties of the counter, eg:

- `deprecated`: means that the counter is obsolete and will never be stepped.
- `notInMOM`: means that the counter is implemented in RNC SW but not specified in the MOM. Should only happen on pre-GA SW.
- `notImplemented`: means that the counter is specified in the MOM but not implemented in RNC SW. Should only happen on pre-GA SW.
- `ropReset`: indicates that the counter value is reset to 0 before each ROP period.
- `noReset`: indicates that the counter value is not reset to 0 at the ROP period and will only be reset at node restart or when the value reaches 2<sup>31</sup>
- `PEG, GAUGE, PDF, etc`: this is the counter type, whose description can be found in CPI.
- a number in square brackets: shown on PDF counters to indicate the number of elements in the array.

Examples:

1. `pmom atmp` - List all PM counters for the **AtmPort** MO
2. `pmom atmp cell` - View description of all **AtmPort** counters that match the word "cell"
3. `pmom atmp .` - View description of all **AtmPort** counters
4. `pmomd . reject` - List all counters matching the word "reject"
5. `pmomd . . . . reject` - List all counters whose description contain the word "reject"

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

6. `pmomd . . sequence:long` - List all counters of type `sequence:long`
7. `pmomd . . . peg` - List all counters of type PEG
8. `pmomd . . . noreset` - List all counters which do not reset at the ROP period boundary
9. `lmom cdmachan all` - View info about **CdmaChannel** log attributes

#### 4.4.2 `kmom[d] [<area>] [<kpiname>] [<MOclass>] [<formula>] [<kpidescription>]`

Print description and formulas of KPIs.

Options:

- `d` : gives a shorter printout, without the description and formula part.

Examples:

- `kmomd access` - List all accessibility KPIs
- `kmomd . sp` - List all KPIs whose name matches the word "sp"
- `kmomd . . utrancell` - List all KPIs reported on MO class `UtranCell`
- `kmomd . . . pmTotNoRrcConnectReqCsSucc` - List all KPIs containing the counter `pmTotNoRrcConnectReqCsSucc` in the formula
- `kmomd . . . . multi` - List all KPIs containing the word "multi" in the description text
- `kmom . sp_a` - Show the description and formula for the KPI `Sp_A`

#### 4.4.3 `pget/lpget [<moGroup>|<moFilter>|<proxy(s)>|all] [<attribute-filter>|all] [<value-filter>]`

Read PM attribute(s) from MO(s).

Does not work on RNC MOs (`UtranCell`, `lubLink`, etc).

#### 4.4.4 `spget/lspget [<moGroup>|<moFilter>|<proxy(s)>|all] [<attribute-filter>|all] [<value-filter>]`

Read PM attribute(s) one by one ("slow pget").

Slow but useful in case the standard "pget" command is not working due to some attribute returning an exception.

Note: Does not work on RNC MOs (`lubLink`, `UtranCell`, etc).

#### 4.4.5 `hpget[c][m]/lhpget[c][m] [<moGroup>|<moFilter>|<proxy(s)>] [<attribute-filter>] [<value1-filter>] [<value2-filter>] [<value3-filter>] etc...`

Read PM attribute(s) from MO(s), print horizontally one line per MO (instead of one line per attribute).

Options:

- `c`: display the output in CSV format for easier export to excel (for instance).
- `m`: print all MOs in a single table instead of separate tables per MO class

Example:

- `hpget vcltp` print the counter values for `vclTp` MOs (`pmreceivedcells`, `pmtransmittedcells`)
- `hpget vcltp . ^0$ !^0$` print all `vcltps` that have 0 receivedCells and more than 0 transmittedCells

#### 4.4.6 `pdiff/lpdif [<moGroup>|<moFilter>|<proxy(s)>|all] [<attribute-filter>|all] [<value-filter>]`

Print incrementation of PM attributes.

Runs two consecutive `pget` commands separated by a time interval equal to the value of the uservariable `pm_wait` (default 25 seconds). Displays the value by which the counter(s) incremented in the interval `pm_wait`. The list of MOs displayed are stored in an MO group called `pdiff_group`. Example1: check all `VclTp` MOs whose `transmittedcells` have incremented but whose `receivedcells` have not incremented (could point towards a transmission problem)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
>> pdiff vcltp= transmit !^0
>> pdiff pdiff_group receive ^0
>> acc pdiff_group eteloopback (we can now perform a loopback test on those VclTp to check if they re
```

Example2: check all Os155SpiTtp MOs whose errored seconds have incremented.

```
>> pdiff os155 es !^0
```

#### 4.4.7 hpdiff[m]/lhpdiff[m] [<moGroup>|<moFilter>|<proxy(s)>|all] [<attribute-filter>|all] [<value1-filter>] [<value2-filter>] [<value3-filter>]

Print incrementation of PM attributes, horizontally one line per MO (instead of one line per attribute)

This is the same functionality as "pdiff" except that the counters are printed side by side, one line per MO. For more information, refer to the help of the "pdiff" command.

Options:

- **m** : print all MOs in a single table instead of separate tables per MO class

Example:

- `hpdiff vcltp transmit|receiv pmtransmittedcells` —> print the counter values for vclTp MOs (pmreceivedcells, pmtransmittedcells)
- `hpdiff vcltp transmit|receiv ^0$ !^0$` —> print all vcltps that have 0 receivedCells and more than 0 transmittedCells

#### 4.4.8 pmx[hfdnscwlb3zei] [<moFilter>|<mogroup>] [<counter-filter>|<kpi(s)>] [-l <zipfile>|<directory>] [-w <webdirectory>] [-m <minushours>] [-p <plushours>] [-s <startdate>.<starttime>]] [-e <enddate>.<endtime>]] [-a|-d|-h] [-o <outputFormat>] [-tz <hrs>] [-f <formulafile>] [-j <precision>] [! <unix-cmds>]

Display counter values, extracted from the statistics ROP files.

This command processes the ROP files by using the utilities **pmExtract/pmXtab/pmDiff**.

#### Arguments

- The first argument (**moFilter/mogroup**) is to specify the MO instances whose counters shall be printed: can be identified by an MO group or by a regular expression matching the LDN of the MOs. It is also possible to use the exclamation mark (!) to print all MOs except those matching the filter.
- The second argument can be either:
  - a regular expression matching the name(s) of the counter(s) that shall be printed, eg `pmrrc.*success`
  - a list of KPIs separated by pipe or commas, eg `MpLoad,CcLoad,DcLoad`. The KPI formulas will be read from the default formula file in `moshell/commonjars/pm` or from the formula file specified in the `-f` option.

If the first and second arguments are omitted, the XML ROP files are parsed and the result is stored on file instead of being displayed to screen. The result files will be stored in the same location on the workstation as where the XML ROP files are stored, ie: `~/moshell_logfiles/logs_moshell/pmfiles/nodeipaddress`. This location can be changed with the user variable `pm_logdir`.

#### Options

- **e** : to display KPI values instead of counter values. The KPI name argument is a non case sensitive regular expression. The KPI formulas are taken from the default formula file or from the formula file specified with the `-f` option.
- **h** : to display the counters side-by-side (**h** as in *horizontal*). Otherwise, there is one line for each MO instance and counter.
- **f** : to prevent `pmx` from rechecking if there are any newer xml files (**f** as in *fast*).
- **d** : needed for parsing PEG counters which are not reset at the end of each ROP period. In `pmom` command, these counters are marked with the tag `PEG,noReset`. The command `pmomd . . . peg,noreset` will show all such counters.
- **n** : to aggregate all counters on MO level.
- **k** : to keep the counters marked with the suspected faulty tag `<sf>TRUE</sf>`, without this option they are discarded.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- **s** : to flag the counters with negative or empty value, or those marked with the suspected faulty tag `<sf>TRUE</sf>`. The counters are then displayed with a label showing `SuspectedFaulty`, `NegativeCounter`, or, `EmptyCounter`.
- **c** : to display the number of counter instances in the ROP file. Should be identical to the number of active counter instances shown in `pgetsn`.
- **3** : to print 3GSIM instantaneous counters from `/c/3gsim/stats`
- **i** : to print ETIPG counters from `CELLO_IPTRAN_DEBUG_LOG`
- **w** : to generate a web report in table and graph format. The `-w` switch can be used to specify the location of the webpage. Currently works best with Firefox, avoid IE.
- **z** : to print the timestamps in the timezone of the workstation where moshell is executing
- **l** : to generate an excel graph. Can be combined with **b** option (`pmxlb`) for treating buckets in PDF counters as separate counters. To generate a graph comparing two different periods, specify the periods with the options: `-s1 <startperiod1> -e1 <endperiod1> -s2 <startperiod2> -e2 <endperiod2>`. See examples in the examples section below.

### Switches

- `-m, -p, -s, -e` : for specifying a time span, see `h pmr` (Section 4.4.9) for more information. By default, the time span will only cover the latest ROP file (equivalent to `-m 0.25`).
- `-a, -d, -h` : for specifying if any time aggregation should be used. If the switch is not included, then no time aggregation will occur. Otherwise, time aggregation can be done on hour basis `-h`, day basis `-d` or on the whole timespan `-a`.
- `-l <file>|<dir>` : for specifying a directory containing ROP files. In this case, `pmx` will parse the ROP files stored in that directory instead of from the node. Can also be used in offline mode.
- `-o <fmt>` : for specifying the output format. By default, the output is text but it is possible to specify: `csv`, `html`, or `htmltab`
- `-w <dir>` : for specifying the path where the web report will be stored. Only applicable with option `w` (`pmxw`). When `-w` is not given, a default location will be chosen for the web report.
- `-tz <hrs>` : for printing the timestamps a specific timezone. Eg: `-tz +10` will show the timestamps in UTC+10 timezone.
- `-j <precision>` : to specify the number of digits after the decimal point (eg: `-j 2 => 2 digits after the decimal point`). Only applicable with "pmxe"

It is possible to pipe the command into a unix command (eg: **grep**, **sort**, etc).

Examples:

- `pmx utrancell downtime` - all counters matching "downtime" on all MOs matching "utrancell" will be displayed for the last ROP period
- `pmxz utrancell downtime` - same as above but timestamps will be displayed in the workstation local time instead of utc time
- `pmx utrancell downtime -tz -5` - same as above but timestamps will be displayed in the timezone UTC-5
- `pmxh utrancell downtime` - same as above but the counters will be displayed side-by-side
- `pmxh utrancell downtime -m 3 -a` same as above but the last 3 hours (12 ROP periods) will be read and aggregated into one value
- `pmxhf utrancell downtime -m 3 -a | sort -k 2` print same as above without rechecking for new ROP files ("f" option), then sort on the second field ("| sort -k 2")
- `pmx -m 2` - fetch and parse all ROP files from the last two hours, result to be stored on disk
- `pmxf -s 19000101` - parse all ROP files that currently exist in the `pmlog` directory in the workstation
- `pmxs -m 0.25` - show the counters marked as suspected faulty for the last 15 minute ROP period.
- `pmxn !utrancell=iub-10 pmTotNoRrcConnectReq$` - show the counter `pmTotNoRrcConnectReq` aggregated on all cells except cell `iub-10`.
- `pmxw utrancell=iub-10-1 pmTotNoRrcConnectReq$ -m 24` - show the evolution in graphical format (web) over the past 24 hours of the `pmTotNoRrcConnectReq` counter on the MO `utrancell=iub-10-1`
- `pmxl utrancell=iub-10-1 pmTotNoRrcConnectReq$ -m 24` - show the evolution in graphical format (excel) over the past 24 hours of the `pmTotNoRrcConnectReq` counter on the MO `utrancell=iub-10-1`
- `pmxlb utrancell=iub-10-1 pmTotNoRrcConnectReq$ -s1 20140101.0900 -e1 20140101.1000 -s2 20140101.1100 -e2 20140101.1200`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- Compare the evolution of the counter pmTotNoRrcConnectReq in the period 20140101 from 9:00 to 10:00 , with the period from 11:00 to 12:00 , and use bucket option.

- `pmxi slot=24 reject` - print the ETIPG counters from slot 24 whose name match the word reject
- `pmxel utrancell drop -m 24` - print the utrancell KPIs whose name match the word "drop" for the last 24 hours and output in excel format
- `pmxen utrancell drop` - print the utrancell KPI whose name match the word "drop" and aggregate at node level for the last 15 minutes

Note: only counters that are included in an **active** performance scanner will be displayed.

Performance scanners can be read with the `pgets` command and created with the `pcr` command or from OSS.

In the case of RAN nodes (RNC, RXI, RBS, eNodeB), the OSSRC does not allow external tools (such as moshell) to create/delete/modify a PM scanner. Any attempt to do this will result in OSSRC reverting the change. This is a feature of OSS (write TR/CRs on OSS, not to MoShell!) and applies only to RAN nodes, not MGW.

Note: Handling of duplicated counter instances

Counter instances that are defined in several PM scanners will be reported by CPP as duplicated entries in the ROP files.

This can lead to unexpected results, such as counters with doubled values.

Usually the `pmr`/`pmx` commands can successfully ignore the duplicated counter entries but only when all the processed ROP files have the same type of counter duplication.

Unexpected values will occur when some of the processed ROP files have a duplication while other dont.

Therefore it is highly recommended to make sure that none of the counters are defined in several scanners at the same time.

The commands `pgets` and `pmxs` will show if there are any duplicated counter instances in the PM scanners.

**4.4.9 `pmr[agfkwop3z] [-g <mofilter>|<mogroup>] [-z <mogroup>] [-r <report(s)>] [-l <zipfile>|<directory>] [-w <webdirectory>] [-i <iubCellModule-file>] [-f <formulafile>] [-c <configfile>] [-m <minushours>] [-p <plushours>] [-s <startdate>[.<starttime>]] [-e <enddate>[.<endtime>]] [-o <outputFormat>] [-t <thresholdfile>] [-tz <hrs>] [ <unix-cmds>]`**

Produce PM KPI reports, based on counter values in statistics ROP files and formulas in CPI documentation.

#### Report choice

- if the command `pmrg` is used then all available reports are printed.
- if the option `-r` is used then it is possible to specify which reports to print. Eg: `pmr -r 1,3,5-10,12` → reports 1,3,5,6,7,8,9,10,12 are printed
- if the option `f` is used then the ropfiles are simply fetched and stored in a zipfile. The zipfile is called `<node>_ropfile.zip` and its location is `<logdir>/pmfiles/<nodeaddress>/<date_time>` . A different location can be specified by running: `pmrf <directory>`. The zipfile can then be used for offline processing with the `-l` option.
- otherwise, a menu is printed, prompting the user to enter a report number, or `x` to exit the `pmr` menu.

#### Options

- `a` : for fetching all available ROP files from the node.
- `f` : for fetching and saving the ROP files to a zipfile, which will be called `<node>_ropfiles.zip`. No reports are printed. In case of CPP nodes, a file containing MO data needed for certain reports is included.
- `ff`: same as `f` but the MO data file is not included which makes the collection quicker. This option is used when collecting a DCG since MO data will already be available in the MO dump.
- `g` : for printing all available reports, the menu will not be shown.
- `k` : for keeping the counters marked with the suspected faulty tag `<sf>TRUE</sf>`, without this option they are discarded.
- `w` : for generating a web report in table and graph format. The `-w` switch can be used to specify the location of the webpage. Currently works best with Firefox, avoid IE.
- `o` : for printing official KPI reports, using the KPI names of the CPI. Currently available for RNC only.
- `p` : for generating PNG files, using GnuPlot.
- `3` : for printing 3GSIM instantaneous counters from `/c/3gsim/stats`
- `z` : for printing the timestamps in the timezone of the workstation where moshell is executing

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

### MO filtering

- The option `-z <mogroup>` can be used to save the MOs printed by a report into a MO group.  
Example: to save the MOs printed by the RNC report `Worst 20 cells for RRC`, run  
`pmr -r <report> -z <mogroup>`.
- The option `-g` can be used to calculate KPIs for a smaller group of MOs instead of all. The processing time will be quicker and it is then possible to filter results on smaller groups of resources.  
Example:
  - `pmr -g tdmterm` will show KPIs based on TdmTermGrp counters only
  - `pmr -g cellmod8` will show KPIs based on counters of cells belonging to module 8 only (where cellmod8 is a MO group created with the commands `ma iubmod8 iublinc module =8$` and `ma cellmod8 iubmod8 reserved`)
- The option `-g` can also be used to exclude certain MOs from the reports, by using the exclamation mark (!) as a negative filter.  
Example: to exclude the MOs printed in the report `Worst 20 cells for RRC`, run  
`pmr -r <report> -z worstcellsgroup`, then `pmr -g !worstcellsgroup`.

### Threshold file

To check that the KPIs are within certain criteria, a threshold file can be specified with the option `-t`.

Example of threshold file and syntax explanation can be found at `moshell/examples/pm_files/thresholds.txt`.

Any KPI that are outside the the criteria specified in the threshold file will be marked in red or purple, depending on the severity.

### Generate Web report

Example:

- `pmrwg` : all reports will be saved to a web page in a default chosen location which is printed at the end of the `pmr` printout
- `pmrw -r 1-10,14 -w /home/user/public_html/RNC14` : reports 1-10 and 14 are saved to a web page located under `/home/user/public_html/RNC14`

Each `pmr` report in the web page shows up as a table and in some cases a graph is also generated. Currently works best with Firefox, avoid IE.

### Generate PNG files

The option `"p"` can be used to generate PNG files, using GnuPlot.

(\*\* Note : you may have to do a new re-install of cygwin to get gnuplot working)

The GnuPlot package can be installed from cygwin "select packages" window: Graphics > gnuplot

Example:

- `pmrp -m 24`

For RNC, it is recommended to run the plot on selected cell by using MO group. Example:

- `ma myrbs iublinc=1234 reservedby`
- `pmrp -m 24 -g myrbs`

### Suspected faulty counters

If some ROP files contain counters marked with the "suspected faulty" tag (`<sf>TRUE</sf>`) then they are silently discarded unless the user has given the option `k` (e.g. `pmrk`) in which case the suspected faulty counter values will be kept. To identify which counter values are marked with the suspected faulty tag, the command `pmxs` can be used.

### Online/Offline

#### 1. Online:

To use `pmr` online (ie when connected to a node) just skip the `-l` option.

The function will check if there are any new XML files on the node, download any new XML files to the workstation as long as their timestamp is within the timespan specified by the `-m/-p/-s/-e` options, get some MO configuration data and print the menu.

#### 2. Offline:

To run `pmr` offline, specify the location of the ROP files with the `-l` option. The location can be a folder or a zipfile produced by the `pmrf` command. Refer to the "Offline mode" chapter for more info.

### Time options

If the command `pmra` is used then all ROP files are fetched. Otherwise, the ROP files of the last hour are fetched.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The `-m` and `-p` options are used for specifying the timespan that the reports should cover, in hours or parts of hours. The timespan is relative to the node's current date/time if online, and relative to the most recent file timestamp if offline.

Example:

- `pmr -m 5` - will show statistics for the last 5 hours.
- `pmr -m 5 -p 3` - will show statistics starting 5 hours ago and 3 hours from then on (ie: up to 2 hours ago).
- `pmr -m 0.25` - will show statistics for the last 15 minutes
- `pmr -m 0.5` - will show statistics for the last 30 minutes
- `pmrf -m 18 /path/to/ropfiles` - fetch the ropfiles of the last 18 hours and put them in a zipfile called `<node>_ropfiles.zip` in the folder `/path/to/ropfiles`

By default the timespan is for the last 1 hour.

The `-s` and `-e` options are used for specifying an absolute timespan. The timestamp should be in the format `yyyymmdd[.hhmm]`

Example:

- `pmr -s 20050705 -e 20050710`
- `pmr -s 20050705.1000`
- `pmr -s 20050705.1000 -e 20050705.1915`

The `-tz <hrs>` switch is for printing the timestamps a specific timezone. Eg: `-tz +10` will show the timestamps in UTC+10 timezone.

### Piping

It is possible to pipe each report into some unix commands. The pipe is done at the "Your Choice" prompt, after the menu listing all the reports. Example:

- `4 | sort -k3 -n` - to view report 4 and sort it numerically on the third column
- `5 | grep 205` to view report 5 and grep for lines matching "205"
- `3 | sort -k4 -n -r` to view report 3 and reverse sort it numerically on the fourth column

If more reports are deemed necessary (eg reports for other kind of nodes) please contact <mailto:finn.magnusson@ericsson.com> to get these included.

### Output format

The output format can be specified with the `-o` option.

By default, the output is text but it is also possible to specify the following formats:

- `csv` Comma separated
- `html` HTML
- `htmltab` HTML tables only (use for embedding into a web page)

Note: HTML reports are best generated with the option "w", e.g. "pmrw". See description further up.

### lubCellModule file

This file is only needed when offline, when printing reports aggregated on lub/Module/AtmPort/Subrack level.

If ROP files have been collected with "pmrf" then the file is automatically included with the ROP files and will be used when parsing the ROP files offline. Otherwise, the path of the file can also be specified with the option "-i".

This file contains the following printouts:

```
get ^aal5tpvcctp= vcltpid
get ^unisaalt= aal5tpvcctp
get ^nbapcommon= unisaaltpref
get ^iublink= rncmoduleref
get ^utrancell= iublinkref
get device= spmreference
get ^ipaccesshostet= ipInterfaceMoRef
get ^sctp= rpuid
pr ^((ipinterface|pluginunit|atmport|vpltp|vcltp|ethernetswitchport|ethernetswitchmoduleport|internaleth
```

Note: Handling of duplicated counter instances

Counter instances that are defined in several PM scanners will be reported by CPP as duplicated entries in the ROP files.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

This can lead to unexpected results, such as counters with doubled values.

Usually the pmr/pmx commands can successfully ignore the duplicated counter entries but only when all the processed ROP files have the same type of counter duplication.

Unexpected values will occur when some of the processed ROP files have a duplication while other don't.

Therefore it is highly recommended to make sure that none of the counters are defined in several scanners at the same time.

The commands pgets and pmxs will show if there are any duplicated counter instances in the PM scanners.

#### 4.4.10 **pmef[fd][cgur] [<pm\_logdir>] [-b <boardgroup>] [-f] [-m <minushours>] [-p <plushours>] [-s <startdate>[.<starttime>]] [-e <enddate>[.<endtime>]]**

Fetch/decode event ROP files (RNC/RBS only).

The -m/-p/-s/-e options allow to specify the time period, see below for more information about time options.

The **f** option is for fetching the ROP files, the **d** option is for decoding them.

The **c**, **g**, **u**, **r** options are for specifying which type of event files shall be fetched/decoded. If none of these options are specified then all four types of event files are fetched/decoded:

- **c** for CTR/CellTrace
- **u** for UETR/UeTrace
- **g** for GPEH
- **r** for RNC\_Exception

The **-b <boardgroup>** option allows to fetch GPEH files on certain modules only, eg: `pmefdg -b mod1`.

The **-f** option is for decoding with the decoder option `--force`.

It is possible to specify the path where the event ROP files are stored:

- with the user variable `pm_logdir` (in `~/.moshellrc` or via `uv` command, see Section 4.3.1). If `pm_logdir` is undefined then the storage location will be the normal moshell log directory (`~/moshell_logfiles/logs_moshell/pmfiles`)
- from the command line, by specifying a directory as first argument to the command.

Examples:

- `pmefc` - fetch event ROP files for CTR only
- `pmefdug` - fetch and decode event ROP files for UETR and GPEH only
- `pmef` (or `pmefcgu`) - fetch event ROP files for all three event measurement types
- `pmef` (or `pmefcgu`) - decode existing event ROP files for all three event measurement types (no fetching)
- `pmef /home/eric/eventrops` - fetch ROP files to **/home/eric/eventrops**
- `pmef /home/eric/eventrops -m 1` - decode the ROP files that were stored in **/home/eric/eventrops** in the last hour

The `ls -l` printout of the node's event ROP files directory is saved in the file "node\_ls.log".

Time options:

The **-m** and **-p** options are used for specifying the timespan that the reports should cover, in hours or parts of hours. The timespan is relative to the node's current date/time if online, and relative to the most recent file timestamp if offline.

Example:

- `pmefgd -m 5` - fetch and decode gpeh rop files for the last 5 hours.
- `pmefd -m 5 -p 3` - fetch and decode all event rop files, starting 5 hours ago and 3 hours from then on (ie: up to 2 hours ago).
- `pmedu -m 0.25` - decode uetr rop files for the last 15 minutes
- `pmef -m 0.5` - fetch all event rop files for the last 30 minutes

By default the timespan is for the last 0.25 hour (15 minutes).

The **-s** and **-e** options are used for specifying an absolute timespan. The timestamp should be in the format `yyyymmdd[.hhmm]`

Example:

- `pmef -s 20050705 -e 20050710` - decode all event rop files between the dates 20050705 and 20050710
- `pmefc -s 20050705.1000` - fetch all ctr rop files from the 20050705 at 10:00 to now

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `pmefdu -s 20050705.1000 -e 20050705.1915` - fetch and decode uetr rop files between the 20050705 at 10:00 to the 20050705 at 19:15

Note: for GPEH, make sure the attribute `RncFunction::gpehDataLevel` is set to 1 (ALL\_DATA). Otherwise the GPEH ROP files will only contain header data.

#### 4.4.11 `pst` [`<scan-filter>`][`<scan-proxy>`] [`<scan-state>`]

List all PM scanners and their state. Examples:

- `pst`
- `pst . act` - list all active scanners
- `pst gpeh susp` - list all gpeh scanners that are suspended

#### 4.4.12 `pgets[m][n][r]` [`<scan-filter>`][`<scan-proxy>`] [`<contents-filter>`]

Print the counters defined in the scanner(s).

Without argument, all scanners are printed.

For a STATS scanner, we see the following:

- a header showing the proxy number, the scanner name, the state, the granularity period in seconds and the total number of counters activated.
- the list and number of counters grouped by MoClass
- with the option `n` we see the number of counter instances for each scanner (only applicable when the contents-filter is empty).
- for each counter, some optional tags may be shown:
  - deprecated: means that the counter is obsolete and will never be stepped.
  - a number in square brackets: shown on PDF counters to indicate the number of elements in the array.
  - notInMOM: means that the counter is implemented in RNC SW but not specified in the MOM. Should only happen on pre-GA SW.

For an EVENT scanner (GPEH/UETR/CTR), we see the following:

- a header showing the proxy number, the scanner name, the state, and the total number of events activated in the filter
- a list of filters such as IMSI, ACCESS\_CELL, CELL, etc
- the list and number of event filters grouped by event category

It is possible to use a *contents-filter*. If used, then only the scanners containing a string matching the filter (not case sensitive) will be displayed.

Options:

- `m` : groups the printout by MO and counter instead of by scanner. It shows for each counter in how many scanners they are defined and which ones.
- `n` : shows the number of counter instances defined in each scanner. The number of counter instances for each scanner are kept in a cache which stays for the duration of the moshell session. To refresh the cache, run `pgetsn` with the option `r`, ie `pgetsnr`.

Examples:

- `pgets stats` - list all statistics scanners
- `pgets stats receive` - list all stats scanners whose contents matches `receive`
- `pgets stats atmport.*receive` - list all stats scanners whose contents matches `atmport.*receive`
- `pgets . cell=12345` - list all scanners whose contents match the string `cell=12345`
- `pgetsn stats` - show the number of counter instances defined in each statistics scanner
- `pgetsm stats load` - for all counters matching the word `load`, show if they are defined in a scanner and if yes, in how many scanners and which ones
- `pgetsm stats utrancell` - same as above but for all counters belonging to `utrancell`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `pgetsm stats 2` - show all counters which are defined in two scanners

#### 4.4.13 `pcr[pcfpda]/lpcr[pcfpda] <scannerName> <moclass-filter>|<moinstance-filter>|<mo-group>|<counter-file> [<counter-filter>] [<granularity>]`

Create a statistics scanner.

Note: EVENT scanners are already created by default and shall be set with the "pset" command.

The granularity field is optional.

- On CPP nodes, it can be set to 300, 900, or 3600 seconds.
- On COM nodes, it can be set to 10, 30, 60, 300, 900, 1800, 3600, 43200, or 86400 seconds.

By default it will be set to 900 (15 minutes). In a RCS node, it may be necessary to run the following command before being able to select other granularity values: `/misc/authlevel disabled;/pms/rp ecim`

Options:

- `c`: for activating counters on all MO classes matching the filter as well as all their children/grandchildren, etc.
- `f`: for adding counters even if they are already included in another scanner.
- `p`: for creating a PREDEF scanner. By default a USERDEF scanner is created.
- `d[a]`: for debug. The syntax of "pcrd" is: `pcrd/pcrda <counter-file>` where the format of the counter-file shall be as per below. The purpose of "pcrd" is to test a counter file and identify any pm counters that are not supported by the node SW. With option "a" (pcrda), it is also possible to test which regular attributes can be included in a scanner.

A negative filter (!) can be used on the moclass or counter filter in order to exclude certain MOs or counters. See more info about regular expressions used in the filters in Section 3.

Note: for certain MO classes, the configuration attributes can be included in the scanner. The user variable `include_nonpm` must be set to 1.

Examples:

- `pcr atmport_vcltp atmport|vcltp` - A new user defined scanner is created for all MOs of class `AtmPort` and `VclTp`
- `pcr atmport_vcltp atmport=ms-6-1 received 300` - Only the MO instances whose RDN match "atmport=ms-6" are included. Only counters matching "received" will be activated. The granularity period will 5 minutes (300 seconds).
- `lpcr vc_scanner atmport=ms-6-1,vptlp=1,.*vcltp=100 ->` only the MO instances whose LDN match "atmport=ms-6-1,vptlp=1,.\*vcltp=100" are included.
- `pcr atmport atmportgroup` - All counters for MO group `atmportgroup` are added to a new scanner called `atmport`.
- `pcr atmport_utranrel ~/mycounters.txt` - All counters stored in the file `/mycounters.txt` are added to the scanner `atmport_utranrel`. See format of the counter file below.
- `pcrc all_transport transportnetwork` - All counters for all MO classes lying under the `TransportNetwork` MO are added to the scanner `all_transport`
- `pcrc all_transport_notMtp transportnetwork!mtp3` - All counters for all MO classes lying under the `TransportNetwork` except `Mtp3` MOs are added to the scanner `all_transport_notMtp`
- `pcrc all_rnc_without_utranrel rncfunction!utranrelation|utranrel !rej` - All counters for all MOs lying under `RncFunction` except `UtranRelation` and `UtranCell` MOs and not counters matching "rej"
- `pcrp myscanner atmport` - all counters in MO classes matching "atmport" are included in the scanner `PREDEF.myscanner.STATS`
- `pcrp PRIMARY ~/primarycounters.txt` - all counters stored in the file `/primarycounters.txt` are included in the scanner `PREDEF.PRIMARY.STATS`

Notes:

- If certain counters are already defined in another scanner of that node, they will be automatically excluded from the new scanner in order not to have duplicate lines in the XML file. This can be bypassed by using the `f` option, eg:  
`pcrf atmport_vcltp atmport|vcltp` (not recommended)
- By default, the counters that are marked as `deprecated` or `notInMOM` are not included by the `pcr` command. This behaviour can be changed by setting the uservariable `exclude_deprecated` to 0.

Format of the counter file: The spaces at the beginning of the line are not necessary, they are just shown for readability purposes. To comment out some lines, just precede the line with a hash sign (#). Any words after the first word are ignored.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Example:

```
AtmPort
  pmReceivedAtmCells
  pmSecondsWithUnexp
UtranCell
  pmCellDowntimeAuto
  pmCellDowntimeMan
  pmChSwitchDch128Fach
  pmChSwitchDch384Fach
  pmChSwitchDch64Fach
```

It is also possible to take the printout from pmom as is. This will work as well and saves having to do any editing.

```
#####
MO Class          Pm Counters
#####
UtranCell          296
                   pmCellDowntimeAuto
                   pmCellDowntimeMan
                   pmChSwitchDch128Fach
                   pmChSwitchDch384Fach
                   pmChSwitchDch64Fach
                   pmChSwitchFachDch
                   pmChSwitchFachIdle
```

It is also possible to use the following format for the counter file (for creating counters on MO instances where each MO instance will use a different set of counters).

Example:

```
ManagedElement=1,RncFunction=1,UtranCell=30124
pmRes1
pmRes2
ManagedElement=1,RncFunction=1,UtranCell=30125
pmRes3
ManagedElement=1,RncFunction=1,UtranCell=30126
pmRes2
pmRes4
```

It is also possible to take the printout of pgets as is. This is useful to clone an existing scanner, eg with a different rop period.  
Example:

```
l+
pgets <oldscanner>
l-
pbl <oldscanner>
pcrf <newscanner> $logfile . 300
```

#### 4.4.14 pbl <scan-filter>|<scan-proxy>

Suspend a scanner. This means that counters defined in this scanner will not be recorded in XML files each granularity period.

#### 4.4.15 pdeb <scan-filter>|<scan-proxy>

Resume a scanner.

#### 4.4.16 pdel <scan-filter>|<scan-proxy>

Delete a scanner.

Note: only statistics scanners can be deleted or created. Event scanners (GPEH/UETR/CTR) are fixed and can only be set.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

#### 4.4.17 emom [uetr|gpeh|ctr|all] [<event-filter>]

Display list of events available for each kind of event-based scanner (RNC/RBS only).

The event reference files are stored in **moshell/commonjars/eventfiles** and are SW dependent.

MoShell automatically chooses the appropriate version of the event files so the user does not have to worry about this.

Examples:

- `emom u ranap` - display all ranap related events for uetr
- `emom all audit` - display all events containing the word "audit" for all kind of event based scanners

#### 4.4.18 pset[d]

Set the contents of an event-based scanner (RNC/RBS/LTE only).

Following syntaxes apply:

- set filters for UETR/UTRACE:

```
pset[d] [-s/-f <ip>:<port>] <scan-filter>|<scan-proxy> <event-filter>|<event-file>|all
<imsi>
```

- set filters for GPEH/CELLTRACE:

```
pset[d] [-s/-f <ip>:<port>] <scan-filter>|<scan-proxy> <event-filter>|<event-file>|all
[<moGroup>|<moFilter>|all] [<ue-fraction>] [<filter>=<value>]
```

- set filters for CTR:

```
pset[d] [-s/-f <ip>:<port>] <scan-filter>|<scan-proxy> <event-filter>|<event-file>|all
<moGroup>|<moFilter> [<trigger-event>]
```

Use the `emom` command to view the list of events that apply for each measurement type.

Event scanners UETR/CTR are applicable for RNC only; UETRACE/CELLTRACE are applicable to LTE eNB; GPEH scanners are applicable to RNC and WRBS only.

The following fields are optional, if no value is entered (or `all` in the case of GPEH/CELLTRACE *utrancell-filter*), no filter will be sent to the node who will then use a default setting:

- *trigger-event* default=RRC\_RRC\_CONNECTION\_SETUP
- *ue-fraction* default=200
- *utrancell-filter in GPEH/CELLTRACE* default= all cells
- *filter* additional filters, for instance LTE filters `ASN1` and `GUMMEI`

A negative filter (!) can be used on the *event-filter* in order to exclude certain events. See more info about regular expressions used in the filters in Section 3.3 (or command `h syntax`).

An event-file can be specified instead of an event-filter. The event-file should list all the events required, one per line, in a similar format to the event files found in **moshell/commonjars/pm**.

For LTE eNB, the options `-s/-f` can be used to specify the output mode of the scanner:

- `s`: output mode `STREAM`
- `f`: output mode `FILE_AND_STREAM`

If these options are used then the `ip` address and port for the stream must be given as argument to the option, example: `-f 10.12.45.38:3402`. If these options are not specified the the output of the scanner will be to `FILE` only.

Examples:

- `pset 10000 S1 cell=2A 300` - set a LTE CELLTRACE scanner for S1 events on cell=2A, UE fraction=300
- `pset 4 X2 all ASN1=true GUMMEI=0x214365ABCD00,0x214365ABCD01` - set a LTE CELLTRACE scanner for X2 events on all cells with filter ASN1 and GUMMEI filters
- `pset 20001 ranap cell=30456` - set a CTR scanner for ranap events on cell=30456 using the default triggering event
- `pset 30006 all` - set a GPEH scanner for all events using no cell filter and no ue-fraction filter. All cells will be selected and the default ue-fraction will be used (200).
- `pset 30006 all all 1000` - set a GPEH scanner for all events on all cells using the ue-fraction 1000
- `pset 30004 handover 304[0-6] 1000` - set a GPEH scanner for all events matching "handover" on cells 3040 to 3046 with ue-fraction=1000

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `pset 30004 nbap mycellgroup - set a GPEH scanner for all nbap events on all cells of the MO-group mycellgroup`
- `pset 10002 rrc|nbap!connection 123456789012345 - set a UETR scanner on imsi 123456789012345, for all events matching "rrc" or "ranap" but not "connection".`
- `pset 30005 .!rrc_paging_type_1 - set a GPEH scanner for all events except RRC_PAGING_TYPE_1`
- `pset 30005 ~/myevents.txt - set a GPEH scanner for all events listed in /myevents.txt, no filter to be used for the cells nor the ue-fraction.`
- `pset 10004 -s 10.34.75.12:5443 . all 1000 - set a LTE CELLTRACE scanner for all events, without a cell filter and with UE fraction=1000, output stream to ip 10.34.75.12 and port 5443`

Note: The `d` option ("debug") can only be used on its own. The syntax of `psetd` is: `psetd <scanner> <event-file>` where the format of the event-file shall be as per above. The purpose of `psetd` is to test an event file and identify any events that are not supported by the node SW. The functionality of `psetd` is implemented as an moshell script, the script can be seen in `moshell/commonjars/scripts/psetd.mos`. The `psetd` command is called via an alias listed in the moshell file. Type `alias` to see the list of aliases. This shows that the user can define any new commands they need by adding an alias in the `.moshellrc` which will point to a script file. The parameters to the command will be sent to the script file via the positional parameters `$1`, `$2`, `$3`, etc. The variable `$0` will be set to the whole line contents. See `h run` for info.

Note: For RBS GPEH, the *utrancell* and *ue-fraction* fields are not applicable. If the event-filter does not match any events, the RBS GPEH scanner will be set to empty.

## 5 Lazy

### 5.1 Software Upgrade

Create the UpgradePackage MO:

```
cr swmanagement=1,upgradepackage=<name>
```

At the prompt, enter *FTP Server Address* where the UP is stored and the path to the *UCF (Upgrade Control File)*. At the next prompts, enter *FTP Server UserID* and *Password*, or just `d` for default value (will be **anonymous**)

Perform the SW installation:

```
acc upgradepackage=<name>$ nonblockinginstall
```

Monitor installation progress:

```
polu
```

Perform the SW upgrade:

```
acc upgradepackage=<name>$ rebootnodeupgrade
```

Monitor upgrade progress (confirmUpgrade will be done automatically):

```
polu
```

Check that the new cv is using the new upgrade package

```
cvls
```

Note: if `polu` was not run after performing `rebootnodeupgrade`, the upgrade will have to be manually confirmed:

- `get upgradepackage=<name>$ state` → wait until state "awaitingconfirmation"
- `acc upgradepackage=<name>$ confirmupgrade`

Note: in case the upgrade was started via the OSSRC SMO application, the `polu` command should not be used to monitor the progress since it will automatically confirm the upgrade which will confuse SMO.

### 5.2 RNC lub operations

- `str` - view state of all lub/Cells/Channels/Nbap/Nodesynch in a table format (type `h str` for more info)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `bl/deb iublink=<iubname>$` - block/deblock an iublink
- `bl/deb cell=<cellname>` - block/deblock a cell
- `str -i <iubname>` - view states for sites related to a particular iub filter only

### 5.3 Common RNC lub Integration Problems

When trying to integrate a new RBS, some data mismatch might cause the *lub*, *Cell*, or *Channels* not to come up. Things to check are the values of:

- Transmission
- AAL2 Addresses
- VCI values
- localCellId

Take a print of all MOs related to the lub and check that the vci values match on both sides, check if any related MOs are down:

```
lk iublink=<iub>    #in RNC
lk iub=<iub>         #in RBS
```

Find out the Aal2Ap used by that lublink in RNC:

```
lk iublink=<iub>
```

Check that the AAL2 addresses match on all sides:

```
get aal2routingcase=<rbsroutingcase> (in RNC and RXI)
get aal2sp=1                          (in RBS and RXI)
```

Check that the AAL2 path id's match on both sides:

```
get aal2pathvcctp=<pathname> pathid (in RNC, RBS, and RXI if applicable)
```

Check that the aal2 continuitycheck match on both sides:

```
get aal2pathvcctp=<pathname> continu (in RNC, RBS, and RXI if applicable)
```

Check that localcellid match on both sides:

```
get cell=<cell> local (in RBS and RNC)
```

Perform a loopback test on all VCIs of that iub, to see if transmission is ok

```
lacc atmpport=<port>,vpltp=<vp>,vpctp=1,vcl eteloopback
```

Check RNC/RBS alarms

```
al
```

Check RNC devices are ok

```
std
```

Check Cell error code, cellbarred, actor info, etc.

```
ced
tgc/tgd (type "h tg" for info)
```

Check RNC/RBS general state, look for potential HW/SW faults

```
dcgm
```

Check all cross-connects are ok



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
stc (mainly in RXI/MGW)
```

Find out the module MP that is handling the site and check if there are some errors:

```
get iublink= module
lh modXX te log read
```

(to restart the module MP, you can use the command "acc modXX restart")

## 5.4 Common RNC lu/lur Integration Problems

Take a print of all MOs related to the lu and check that the VCI values match on both sides. Also check if any related MOs are down:

```
lk mtp3bsrs=<name>
```

Perform a loopback test on all VCs of that interface, to see if transmission is OK:

```
lacc atmport=<port>, vpltp=<vp>, eteloopback
```

Check that the pointcodes matches on both sides

```
get mtp code
```

Activate/deactivate a C7 link

```
lacc mtp3bsls=<name>, deactivate/activate
```

Block/deblock an Aal2Path, check that the *pathIds* and *a2ea* addresses match on both sides:

```
b1/deb aal2pathvcctp=<pathname>
get aal2pathvcctp=<pathname> pathid
get routingcase=<name>
```

## 6 Scripting

Moshell supports the use of variables and logical constructs. These can be used directly from the command line or within MoShell command files.

### 6.1 Preset Variables

The following variables are set immediately after MoShell startup:

- `$logdir` - path to the **moshell\_logfiles/logs\_moshell** directory
- `$moshelldir` - path to the moshell directory
- `$gawk` - path to gawk
- `$ipaddress` - IP address of the node that MoShell is connected to
- `$moshell_version` - the MoShell version
- `$tempdir` - path to the directory containing all temporary files for this moshell session. Gets deleted at the end of the session.
- `$uname` - output from the "uname -a" command, showing the operating system of the workstation running moshell.

The following variables are set after the MOM has been parsed:

- `$momversion` - the MOM version of the node (eg: RNC\_NODE\_MODEL\_E\_5\_3, MGW\_NODE\_MODEL\_R3\_9\_0)
- `$cellmomversion` - the CPP MOM version (3.3, 4.3, 5.1, etc) of the node
- `$momdocnumber` and `$momdocrevision` - the document number and revision of the MOM (eg: 15554-AXD10503/1 , rev: Z1)

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `$background_pid` - the process id of a process started into background from moshell command line, eg:  
`l $moncommand > $logfile &`

These variables can be handy to have when a script needs to know what SW revision is running in the node or what kind of node it is.

The following variables are set after running certain MO commands:

- `$cppversion` the CPP version of the node, according to the information in the current UpgradePackage.
- `$nr_of_mos` the number of MOs that were printed on screen by the last run of the `pr/st/get/prod/fro/set/del/acc` commands.
- `$nr_of_mos_ok` the number of MOs that were successfully operated upon by the last run of the `set/del/acc` commands.
- `$command_result` set after running the `cr/pcr/pset/trun` commands. Possible values: 0 for success, 1 for failure.
- `$returncode` set after running any moshell command. Possible values: 0 for success, 1 for failure.
- `$action_result` set after running the `acc` command. Contains the return value of the action.
- `$nr_of_alarms` the number of active alarms on the node. Set after the last run of the `al` command.
- `$nr_of_cvs` the number of CV:s that exist on the node, is set after the last run of the `cvls` command.
- `$nr_of_scanners` the number of scanners printed by the last run of the `pst/pgets/pdel/pbl/pdeb` commands.
- `$nr_of_counter_instances` the number of counter instances printed by the last run of the `pgetsn` command.
- `$moncommand` the command to start the monitor client after having run the "mon" command.

The following variables are set after running the commands `bp/bo`:

- `$coremp_pos` the positions of the core MPs, eg: 001000:001100
- `$coremp_type` the board types of the core MPs, eg: GPB53:GPB53

The following variables are set after running one of the `l+/u+/u-` commands:

- `$logfile` the logfile that is currently open. Set immediately after executing the `l+` command, stays set even after `l-` and will only be reset the next time a new logfile is open with `l+`
- `$undologfile` the logfile used by the undo command. Set immediately after executing the `u+/u+s` command, stays set even after `u-` and will only be reset the next time a new undo mode is started with `u+/u+s`.
- `$undocommandfile` the command file that can be used to undo the commands that were run between `u+/u+s` and `u-`. Set immediately after executing the `u-` command.
- `$undodelcommandfile` - the file containing the delete commands. Only applicable to simulated undo mode `u+s`

The following variable is set after having logged on to the node via telnet/SSH or FTP/SFTP.

- `$password`

The contents of the variable can not be printed, it will only show if it's empty or not. By setting this variable to empty (by doing: `$password =` ), this will force MoShell to check the password again. Useful in case the password has changed on the node during the MoShell session.

The `$nr_of_vars` variable is set after running the `pv` command.

This variable indicates the number of scripting variables that were printed in the last `pv` printout. By using `pv` together with a filtering pattern (eg: `pv $stable`), it is possible to find out the number of variables that had matched the pattern, for instance the number of elements in a hashtable.

The `$nr_of_lines` variable is set after using the functions "readfile" or "testfile".

After using the function "testfile", this variable is set to 0 if the file does not exist and to 1 if the file exists.

After using the function "readfile", this variable is set to 0 if the file does not exist and to the number of liens in the file if the file exists. The difference between testfile and readfile is that testfile won't actually read the file, it will just check if the file exists whereas readfile will test the file, then read it.

Example1:

```
$lineContent = testfile(/path/to/myfile)
if $nr_of_lines = 0
  l echo "File not found!"
  return
fi
```

Example2:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
$lineContent = readfile(/path/to/myfile)
if $nr_of_lines = 0
  l echo "File not found!"
  return
fi
for $lineNumber = 1 to $nr_of_lines
  if $lineContent[$lineNumber] ~ thispattern
    print We found it! The line is $lineNumber.
    return
  fi
done
```

Example (list all scripting variables currently set via `pv`):

- `$logdir = /home/eanzmagn/moshell_logfiles/logs_moshell`
- `$momversion = RNC_NODE_MODEL_E_5_3`
- `$moshellldir = /home/eanzmagn/tools/moshell`
- `$cellomomversion = 4.3.1`
- `$gawk = /home/eanzmagn/tools/moshell/gawk`
- `$ipdaddress = 10.1.128.17`
- `$moshell_version = 6.1`
- `$nr_of_mos = 1`
- `$password = *****`

## 6.2 Variable assignment

A variable value can be assigned in six ways, see below.

A variable can also be unassigned, using the `unset` command.

By using the command `unset small`, all variables are unset, except:

- the "system" variables (`$gawk`, `$ipaddress`, `$password`, `$moshell_version`, `$moshellldir`, `$logdir`, `$momversion`, `$cellomomversion`)
- the "global" variable(s) (ie: assigned with the `:=` sign, instead of `=`).

By using the command `unset all`, all variables are unset except the "system" variables.

NOTE: It is always good practice to unset a variable as soon as it is not needed anymore since having too many variables defined slows down the processing of the command line. It is also good to do `unset all` at the beginning and end of a command file (and before doing the `return` command) in order to avoid interference from un-needed variables. See the script examples in **moshell/commonjars/scripts**

To print all currently assigned variables, use the command `printvar`. To just print one variable, type:

`pv <pattern>` (where the pattern matches the variable(s) to print)

```
print "$variable"
```

The variable value can be assigned in six ways:

1. From the command line.

The variable to be assigned is on the left side of the equal sign and the value is on the right side. Each element must be separated by spaces. Example:

```
$i = 3
$node = RNC
$password =
```

By running `password =` this sets the password to an empty value and will force MoShell to ask for the password again.

2. At moshell startup, using the `-v` option. In this case, the `"$"` sign should be omitted. (otherwise it gets interpreted by the Unix shell)

Example:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
moshell -v upmo=CXP9011008_R1A03,ftpserv=10.1.0.16,secure_shell=1,secure_ftp=1 rnc34}
```

In this case, we can see that scripting variables (\$upmo and \$ftpserv) and user variables (secure\_shell and secure\_ftp) have been mixed in the same statement. This is OK because any variable that is not recognised as a user variable will be treated as a scripting variable.

### 3. From the following commands: get, fro ip2d/d2ip, h2d/d2h, uv

The mo-filter and attribute-filter must be specified, then comes the redirection sign (>), then the variable name. If several attributes are printed, only the last attribute value gets assigned into the variable. Examples:

- get 0 productName > \$nodeType (result: \$nodeType = RBS3202)
- get ethernetlink ipaddress > \$ipaddress (result: \$ipaddress = 10.1.128.17)
- get configurationversion currentupgrade > \$currentUp (result: \$currentUp = UpgradePackage=CXP9011123\_R12F)
- get \$currentUp administrativedata > \$swRev (result: \$swRev = Struct{5} >>> 1.productNumber = CXP9011123 >>> 2.productRevision = R12F etc.)
- lfro subrack=ms,slot=1,pluginunit=1\$ ^r > \$froid (result: \$froid = 0)
- d2ip -4127129085 > \$ip\_addr (result: \$ip\_addr = 10.1.2.3)
- h2d 0xa > \$res (result: \$res = 10)
- uv ^credential > \$credential (result: /home/ericsson/moshell/commonjars/host.p12)

### 4. Using an arithmetic operation

The following numeric operations are supported:

- + addition
- - subtraction
- \* multiplication
- / division
- % **modulo** (returns the remainder of an integer division operation)

Examples:

- \$i = 1 (result: \$i=1)
- \$i = \$i + 1 (result: \$i=2)
- \$j = \$i \* 3 (result: \$j=6)
- \$k = \$i \* \$j (result: \$k=12)
- \$l = \$i / \$j (result: \$l=0.333)
- \$m = \$k % 5 (result: \$m=2)

Note: Only one operation per line is allowed. A space must exist between each element of the operation. There cannot be more than two members in the operation (ie: \$i = \$j + \$k ==> OK. But \$i = \$j + \$k + \$l ==> NOTOK)

### 5. Using the output from a Unix command:

The Unix command must be surrounded by back-quotes (`). Variables can be used within the Unix command. Examples:

#### • Example

```
$date = `date +%y%m%d-%H%M`
(result: $date = 040930-1745)
get ethernetlink ipaddress > $ipaddress
(result: $ipaddress = 10.1.128.17)
$logfile = $ipaddress_$date.log
(result: $logfile = 10.1.128.17_040930-1745.log)
l+m $logfile
(open a logfile, don't display anything on the screen)
te log read
l-
(close logfile)
$errors = `grep -c ERROR $logfile`
(result: $errors = the number of ERRORS found in the logfile)
l rm $logfile
(remove the logfile)
```

- Example: Making a cv that has the same name as the current startable cv but the last digit is incremented by 1

```
lt configurationversion
get configuration startable > $startable
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
(result: $startable = RBS3045_P2.1.5_CU3_A_01)
$cvname = '$gawk -v cvname=$startable 'BEGIN {
    print gensub(/.\/, "", 1, cvname) sprintf("%02s", substr(cvname, length(cvname)-1)+1) }'`
cvms $cvname
(result: $cvname = RBS3045_P2.1.5_CU3_A_02)
```

## 6. Using result from a unix command

If the built-in function `system` is used instead of the back-quotes ( ``` ) then the exit code of the unix command is saved to the variable (instead of its output).

Example:

```
$result = system(cp $file1 $file2)
(result: 0 if the copy was successful, or some other number if the copy failed)
```

## 7. Using String manipulation

The following string operations are supported: *concatenation* and *substitution / replacement*.

The concatenation is performed by juxtaposing the strings. Syntax for concatenation: `$var = string1string2string3` (the strings are concatenated without space in between) or `$var = string1 string2 string3` (the strings are concatenated with spaces in between)

Syntax for concatenation:

- a) `$var = string1string2string3` (the strings are concatenated without space in between)
- b) `$var = string1 string2 string3` (the strings are concatenated with spaces in between)

The substitution/replacement is performed using the `-s` switch to specify the string to substitute and the `-r` switch to specify the string it should be replaced with. If the `-r` switch is not used, then the string will be replaced by nothing. If the `-g` switch is specified, then all instances of the string to substituted, otherwise, only the first instance.

Syntax for substitution/replacement:

- a) `$var = origString -s strToSubstitute [-r strToReplaceItWith [-g]]`

Regular expressions can be used in the string manipulations. Examples:

```
$var = abc_defabc ghi
$var1 = $var -s abc
```

Result: `$var1 = _defabc ghi`, only first instance of **abc** was replaced

```
$var2 = $var -s \x020
```

Result: `$var2 = abc_defabcghi`, the space sign was removed

```
$var3 = $var -s abc -g
```

Result: `$var3 = _def ghi`, all instances of **abc** were replaced

```
$var4 = $var -s abc -r xyz
```

(result: `$var4 = xyz_defabc ghi`, first instance of **abc** was replaced)

```
$var5 = $var -s abc -r xyz -g
```

(result: `$var5 = xyz_defxyz ghi`, all instances of **abc** were replaced with **xyz**)

```
$var6 = $var -s a.*c -r xyz
```

(result: `$var6 = xyz ghi`, the regular expression **a.\*c** was replaced with **xyz**)

```
$var7 = $varABC$var6
```

(result: `$var7 = abc_defabc ghiABCxyz ghi`, the three strings **\$var**, **ABC** and **\$var6** have been concatenated)

```
$var8 = $var ABC $var6
```

(result: `$var8 = abc_defabc ghi ABCxyz ghi`, there are spaces in between the three strings)

Note: if more advanced string manipulation is needed, it is always possible to use an external program such as `gawk` to do the string manipulation. See the example above about using Unix programs.

## 8. Using output from a predefined function

Currently, the following functions exist:

- `fdn(proxy)` input is the proxy id, output is the FDN
- `ldn(proxy)` input is the proxy id, output is the LDN
- `rdn(proxy)` input is the proxy id, output is the RDN
- `motype(proxy)` input is the proxy id, output is the MO type
- `proxy(string)` input is the LDN or FDN (NOT RDN!), output is the proxy id

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `readinput (string)` input is a prompt that should appear on the screen, so that the user can input an answer which will then be assigned to the variable.
- `readinputsilent (string)` same as `readinput ()` but without echoing the user input. useful for entering passwords.
- `readfile (file)` input is a filename. Each line of the file is assigned into an element of the hashtable into which we have assigned the result of the function. If the file is not found, the variable `$nr_of_lines` is set to 0, otherwise it is set to the number of lines in the file. Note, this should not be used on large files as it will slow down things very much.
- `testfile (file)` input is a filename. If the file is not found, the variable `$nr_of_lines` is set to 0, otherwise it is set to 1.
- `split (string)` The string is split into the array specified on the left side of the equal sign (see example below). The separator used to split the string can be specified in the variable "`$split_separator`". By default it is a space. If the `$split_separator` has been changed and needs to be reset to the default value, just run the command "`unset $split_separator`". The number of elements in the array is stored in the variable `$split_last`
- `mod2nr (string)` Convert a RncModule name into a module number. Eg: `mod2nr(MS-6-1)` returns 1061 , `mod2nr(ES-1-24-0)` returns 241 .
- `check_ip_contact (ipaddress, port)` Check connectivity to a given ip address and tcp port. Returns 0 if ok, or 1 if not ok.

**Example 1:**

```
lt iublink
ma iub iub
for $mo in iub
    $mordn = rdn($mo)
    if $mordn ~ 1023
        lcc $mordn
        lbl $mordn,
    fi
done
```

**Example 2:**

```
$var = readinput(Please confirm [y/n]: )
if $var !~ ^y
    return
fi
```

**Example 3:**

```
$table = readfile(/path/to/myfile)
for $lineNumber = 1 to $nr_of_lines
    print $table[$lineNumber]
    $word = split($table[$lineNumber])
    if $word[1] ~ ^#
        $nr_of_comments = $nr_of_comments + 1
    fi
    unset $word
    unset $table[$lineNumber]
done
```

(Note: by unsetting the entry we've just read - provided we don't need it anymore - will make things faster)

**Example 4:**

Logging in to an unknown node with moshell `-n` option (no ip connectivity check) and checking the type of node

The loop is intended to keep checking if the node happens to be down at the time of the check.

```
for 10
    $var80 = check_ip_contact ($ipaddress, 80)
    $var9830 = check_ip_contact ($ipaddress, 9830)
    $var4912 = check_ip_contact ($ipaddress, 4192)
    if $var4192 = ok
        uv comcli=21
        uv username=expert
        $password = expert
        break
    else if $var9830 = ok
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
        uv comcli=11
        uv username=root
        $password = root
        break
    else if $var80 = ok
        uv comcli=0
        break
    else
        wait 30
        $i = $i + 1
    fi
done
```

Example: The purpose of the little moshell script below is to make a customized CV name like: date\_nodeType\_swRev

```
cvls
$date = `date +%y%m%d`      (result: $date = 040930)
get 0 productName > $nodeType (result: $nodeType = RBS3202)
$nodeType = $nodeType -s RBS (result: $nodeType = 3202)
get configurationversion currentupgrade > $currentUp
(result: $currentUp = UpgradePackage=CXP901913%2_R12N)
get $currentUp administrativedata > $swRev
(result: $swRev = Struct{5} >>> 1.productNumber = CXP901913/2 >>>
2.productRevision = R12N >>> 3.productName = CXP901913%2_R12N ....)
$swRev=`gawk -v currentsw="$swRev" 'BEGIN{ swrev=gensub(/\r|\n/, "", "g", currentsw);
print gensub(/^. *Revision = | >>> 3.product.*$/, "", "g", swrev) }'`
(result: $swRev = R12N)
cvms $date_$nodeType_$swRev (result: cvms 040930_3202_R12N )
```

### 6.3 Hashtables (arrays)

The index and the value of the hashtable can be a variable, a constant, or a mix of both.

All variable assignment methods described in Section 6.2 apply for assigning values into hashtables as well.

To print a hashtable, do: `pv <table>`

Examples:

Assigning constants into a hashtable

```
>> $table[1] = hello
>> $table[2] = hej
>> $table[hoho] = 5
>> pv tab (result printout:)
$table[hoho] = 5
$table[1] = hello
$table[2] = hej
```

Assigning variables into a hashtable:

```
>> $mo = AtmPort=MS-6-1
>> $proxy = proxy($mo)
>> $proxylist[$mo] = $proxy
>> $mo = AtmPort=MS-6-2
>> $proxy = proxy($mo)
>> $proxylist[$mo] = $proxy
>> pv proxylist (result printout:)
$proxylist[AtmPort=MS-6-1] = 103
$proxylist[AtmPort=MS-6-2] = 112
```

More examples on how to use hashtables are described in Section 6.5.

### 6.4 If/Else constructs

The `if` statement must be followed by a *condition*. The comparison operator of the condition must be surrounded by spaces. Zero or more `else if` statements can be used after the `if` statement. Zero or one `else` statements can be after the `if` or `else if` statements.



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

The end of the if/else structure must be specified with a `fi` statement. Each statement must be on its own line and can be followed by one or more commands. Several conditions can be combined, using the logical AND (`&&`), or the logical OR (`||`). Any number of AND/OR can be put on a line but NOT BOTH on the same line.

Grouping conditions with brackets is NOT supported.

The `return` command can be used to exit from the command file in case a certain condition is met. Type `h return` for more information on how to use this command.

Syntax examples:

1. 

```
if <condition>
    command1
    command2
fi
```
2. 

```
if <condition1> || <condition2>
    command1
    command2
else
    command3
fi
```
3. 

```
if <condition> && <condition2> && <condition3>
    command1
else if <condition4>
    command2
else
    command3
fi
```

A condition can use the following comparison operators:

- `=` equals
- `~` matches (as in pattern match)
- `!=` is not equal to
- `!~` does not match
- `>` greater than
- `<` less than
- `>=` greater than or equal to
- `<=` less than or equal to

The words around the operator can be either a variable or a single word but NOT a string containing spaces or a concatenation of a variable and string. Following conditions are syntactically correct:

```
if $var1 = $var2
if mystring ~ $var
if 10 > 3
if $i < 2
```

Following conditions are NOT syntactically correct and will return unexpected results:

```
if mystring_$var1 ~ $var2
if mystring is this ~ your string
```

A condition can also just contain one variable, in which case it will check if the variable exists. The words around the operator can be either a variable or a single word but NOT a string containing spaces or a concatenation of a variable and string. Following conditions are syntactically correct:

```
if $var1 = $var2
if mystring ~ $var
if 10 > 3
if $i < 2
```

Following conditions are NOT syntactically correct and will return unexpected results:

```
if mystring_$var1 ~ $var2
if mystring is this ~ your string
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

A condition can also just contain one variable, in which case it will check if the variable exists.

Example to check if a variable \$var exists. If \$ exists (i.e. has any value set) then it will do something..

```
if $var
    ...do something
fi
```

Example to check for node type and see attenuation accordingly:

```
get 0 productname > $nodeType

if $nodeType ~ 3202 || $nodeType ~ 3104
    get feeder attenuation
    set feeder attenuation 4
else if $nodeType ~ 3101
    get feeder attenuation
    set feeder attenuation 16
else
    get feeder attenuation
fi
```

## 6.5 For constructs

The parameter to the `for` construct can be:

1. `ever` - to repeat the loop an infinite number of times
2. `<numberOfIterations>` - to repeat the loop a specific number of times
3. `$mo in <moGroup>` - to run the body of the loop on each MO of the specified moGroup. MO groups are created using `ma/lma`. See `h ma` (Section 4.1.6) for more info.
4. `$board in <boardGoup>` - to run the body of the loop on each board of the specified board group. Board groups are created using `ba/ba`. See `h bo` (Section 4.3.13) for more info.
5. `$var in $table` for each iteration of the loop, \$var will cycle through the index values of the hashtable \$table
6. `$var = $start to $stop` \$var is assigned every integer value between \$start and \$stop. \$start and \$stop can be variables or constants but must be an integer. If \$start is smaller than \$stop then the order will be ascending, otherwise it will be descending.

The end of the `for` structure must be specified with a `done` statement.

The `wait` command can be used in the body of the loop to specify a delay to wait in between each iteration. The delay can be in seconds, minutes, hours, or even ROP periods. (Type `h wait`, Section 4.3.42 for info.)

Do not use the `sleep` command as this will result in hanging if the loop is aborted.

The loop can be aborted any time by typing `ctrl-z`, then `touch <stopfile>`, then `fg`. The `<stopfile>` path is shown in the window title bar. Type `h ctrl-z` for more info about aborting.

The `break` command can be used within the loop to exit from the loop.

Syntax examples:

1. 

```
for ever
    command1
    command2
done
```
2. 

```
for <numberOfTimes>
    command1
    wait <numberOfSeconds>
done
```
3. 

```
for $mo in <moGroup>
    get $mo <attribute> > $variable
    $variable1 = ....
    set $mo <attribute> $variable1
    etc...
done
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```

4. for $board in <boardGroup>
    bl $board
    facc $board restart 0 1
    wait 10
    deb $board
    lhsh $board vii
done

5. for $proxy in $proxytable
    bl $proxy
    st $proxy
    deb $proxy
    st $proxy
    get $proxy operational > $opstate
    if $opstate != 1
        break
    fi
done

6. for $var = $maxproxy to $minproxy
    del $var
done
for $var = 1 to 6
    te e trace$min process
done
    
```

#### Practical examples:

1. Checking the progress of a UP installation, every 10 seconds. Break from the loop if the result is  
1 (INSTALL\_COMPLETED), and continue with upgrade action. Abort the command file if the result is  
6 (INSTALL\_NOT\_COMPLETED)

```

lt upgrade
acc upgradepackage=xxx nonblockinginstall
for ever
    $return = 0
    wait 10
    get upgradepackage=xxx state > $upstate
    if $upstate ~ ^1
        break
    else if $upstate ~ ^6
        $return = 1
        break
    fi
done
if $return = 1
    return
fi
acc upgradepackage=xxx upgrade
    
```

2. Run a testcase 50 times

```

for 50
    run testcase_3.1.1.cmd
    wait 2m
done
    
```

3. Increase the primaryCpichPower by 0.1 dBm on each UtranCell

```

lt ^utrancell
ma cell ^utrancell
for $mo in cell
    get $mo primarycpichpower $pich
    $pich = $pich + 1
    set $mo primarycpichpower $pich
done
    
```

4. restart all boards in a board group

```

ba spb spb
for $board in spb
    
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
facc $board restart 0 1
done
```

5. Save the fRO values of all programs into a table and then restart every program

```
lma programs_on_slot_19 subrack=ms,slot=19,.*program
for $prog in programs_on_slot_19
  $i = $i + 1
  fro $prog ^res > $frolist[$i]
done
for $fro in $frolist
  restartObj pgm $fro
done
```

6. Restart some boards in a specific order

```
for $var = 20 to 14
  $board = 00$var00
  facc $board restart 0 1
done
```

## 6.6 User-defined functions

Users can define their own functions, using the `func/endfunc` construct.

If the function is called with arguments, these are assigned to the variables \$1, \$2, \$3, etc The variable \$0 is set to the whole line contents.

Example:

1. Define the function (the function definitions can be run in a different command file)

Here we are defining a function which checks the state of the mirrored disks and returns once the disks are in sync

```
func check_disk_state
  #if $1 is undefined or different to an integer value
  #then we set it to 10 seconds
  if $1 ~ ^[0-9]+$
    $wait_interval = $1
  else
    $wait_interval = 10
  fi
  for ever
    wait $wait_interval
    l+om $tempdir/diskstate
    lh coremp mirror s
    l-
    $res = `grep -c "Peer Disk Status: *Valid" $tempdir/diskstate
    if $res > 0
      break
    fi
  done
endfunc

func waitforuser
  $date = `date "+%Y-%m-%d %H:%M:%S"`
  for ever
    $reply = readinput(Waiting from [$date]. Type "y" when ready: )
    if $reply ~ ^[yY]
      break
    fi
  done
  $date = `date "+%Y-%m-%d %H:%M:%S"`
  print "Finished waiting at [$date]"
endfunc
endfunc
```

2. Call the function Here we have made a small script which makes use of our user-defined function.

First we are running a file containing all the definitions for our user-defined functions.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

(Note that the functions can also be defined within the same script, but by keeping all functions in a separate file means that several command files can use the same functions)

We have called the function `check_disk_state` with an argument "5" which in this case will be used as the "\$wait\_interval" parameter in the function

```
run ~/myfunctions_define.mos
for ever
    check_disk_state 5
    waitforuser
    facc 0 manualrestart 0 0 0
    pol
done
```

## 6.7 Nesting for and if statements

It is possible to nest one or more if/else statement within a loop statement and vice-versa. But it is currently not possible to nest an if/else statement within an if/else statement and a loop statement within another loop statement.

The current workaround is to put the for/if constructs into functions. See Section 6.6 for more info on functions.

Example:

The following script starts the install, then checks the state of the install every 10 seconds. Once the upgradepackage is installed, it starts the upgrade. Then it checks the state again and once the upgrade is in state **awaitingconfirm**, it confirms the upgrade.

```
$UP = upgradepackage=CXP9011123_R12F
acc $UP nonblockinginstall
for ever
    wait 10
    get $UP state > $state
    if $state ~ ^1
        break
    fi
done
get $UP state > $state
if $state ~ ^1
    acc $UP upgrade
fi
wait 120
for ever
    wait 10
    get $UP state > $state
    if $state ~ ^3
        break
    fi
done
if $state ~ ^3
    acc $UP confirmupgrade
fi
```

Some more examples:

1. Example to check the `mirror stat` status of the node (i.e. to check whether the passive FTC MP is ready to take over or not)

```
for ever
    board_status -d 00 10 -c "mirror stat" | tee tmpfile.tmp
    board_status -d 00 11 -c "mirror stat" | tee -a tmpfile.tmp
    $tmp = `grep -c "Peer Disk Status: Valid" tmpfile.tmp`
    if $tmp > 0
        break
    else
        wait 60
    fi
done
```

2. Example to check if an upgrade is complete (i.e. the upgradepackage is in state 3)

```
wait 300 #give it some time to run first
for ever
    pol 1 1
    get upgradepackage=mypkg state > $state
```

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

```
if $state ~ ^1
    break #upgrade failed
else if $state ~ ^3
    break #upgrade complete
fi
wait 60
done
```

## 6.8 Example scripts

Example scripts can be found under **moshell/commonjars/scripts** and **moshell/examples/scripting/**

Note two types of comments can be used in scripts:

- *visible comments* start with the "#" sign. These comments are printed on the screen while the script is executing.
- *invisible comments* start with the "/" sign. These comments are not printed on the screen.

## 7 Utilities

The moshell package includes a number of utilities that complement the moshell functionality. They are stored in the same directory as moshell but are executed as separate, stand-alone scripts.

Detailed information about each utility can be found by executing the utility without any arguments. (Except for the PM tools where the help is shown by using the **-help** option, eg: `pmExtract -help`).

### 7.1 Network Management Tools

The network management utilities are for performing operations on many nodes in parallel. The list of nodes on which to perform the operations shall be stored in a file called the **sitefile**. The ip-addresses/DNS-names and passwords of all nodes of the network must be stored in a reference file called the **ipdatabase**.

The ipdatabase uses the following syntax:

```
<nodeName> <nodeIpAddress> <nodePassword>
```

OR:

```
<nodeName> <nodeDNSAddress> <nodePassword>
```

(It is recommended to use the DNS-address instead of the IP-address).

It is also possible to specify user variables and/or scripting variables in the fourth column of the ipdatabase.

An example of a **sitefile** and an **ipdatabase** can be found in the directories **moshell/examples/mobatch\_files**.

The following Network Management tools are currently included in Moshell:

1. **mobatch** to run moshell commands or command files towards many nodes in parallel.
2. **restartcollector** - to collect restart data (manual and spontaneous restarts), system downtime and upgrade data from many nodes in parallel. Data is presented in a report together with statistics and TR mapping and can be imported in other tools (such as OpenOffice Spreadsheet or Excel or Perl::CSV).
3. **swstat** - It has the following 2 functions:
  - a) to collect SW level and CV from many nodes in parallel.
  - b) to delete UP's from many nodes in parallel.
4. **swup** - to SW upgrade many nodes in parallel.
5. **cvms** - to create CV's on many nodes in parallel.

### 7.2 Parameter Auditing Tools

The auditing tools are for offline postprocessing of Node logs (*MO Dumps*) and reference files in order to check parameter validity or health-checks.

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

Note! Most of these tools are now obsolete, please use the moshell built-in command `diff` instead. Type `h diff` for info.

The following tools are included with Moshell:

1. **rncaudit** - to compare rnc moshell dumps against a number of reference files (CDR, winnow baseline, Uerc data, etc.). Certain consistency checks can be performed (scrambling code collisions, cell relations consistency). `Rncaudit -b` option can be used on other nodes as well such as MGW, RXI.
2. **rbssaudit** to compare rbs moshell dumps against a number of reference files (CDR, winnow baseline).
3. **netcheck** - Has the following two functions:
  - a) to parse moshell health-check logs and check for obsolete HW, HW faults, errors, etc.
  - b) to compare pre-check and post-check (eg health check logs before and after an upgrade or a testcase)
4. **mocmd** - Has the following to functions:
  - a) to convert a file from winnow baseline format to moshell command file
  - b) to convert a change\_all script to moshell command file

### 7.3 PM Tools

The PM tools are for offline postprocessing of Performance Management XML files. They consist of a number of command-line tools, designed to be linked together in a pipeline (ie the output of one tool feeds into the next). The following PM tools are included in Moshell:

1. **pmExtract** - to extract counter values from the XML files. The counter values to be extracted are selected based on the managed object and counter name. The counters are output to the standard output, one line per counter value.
2. **pmXtab** - to aggregate counters according to time, Managed Object hierarchy or arbitrary relations
3. **pmDiff** - to output the difference between counter values in successive XML files. This is useful for counters that are not automatically reset after each period (eg. CPP counters).
4. **pmList** - to list all counters defined in a XML ROP file.

For more information, run the utility with the `-help` option (eg `pmExtract -help`) or have a look in the **pmTools** documentation. This document (which may not be the latest can also be found in **moshell/examples/pm\_files/pmTools.doc**)

### 7.4 Miscellaneous Tools

Some other tools included with Moshell are:

1. **pstool** - to display process usage or kill a process tree.
2. **momdoc** - to convert a MOM from xml format to html format
3. **swcomp** - to compare SW between two nodes containing similar HW.

Note: There are other files in the moshell directory but they are moshell components and not meant to be executed as stand-alone scripts.

## 8 Server Maintenance

When running moshell on a server in a multi-user environment, there needs to be regular maintenance in order to clean up the disk and any hanging processes.

### 8.1 Hanging Processes

A known bug of moshell is that it doesn't always shut down all of its spawned processes upon exiting which leads to CPU overload and run out of RAM memory. This problem should now be fixed thanks to the use of various timeouts but if this does not help, then it is recommended to regularly check the rogue processes using the unix command `top`.

Once the `top` command is running, you can type the following commands in the `top` screen:

- `n` followed by the number of processes to display (e.g `n 40`) -> to show more than the default number of 15 processes
- `o` to change the order of the sorting. E.g:



|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- o `time` (to see the processes that have been running for the longest time)
  - o `size` (to see the processes that are using up the most memory)
  - o `cpu` (to see the processes that are using up the most cpu. this is the default).
- `k` followed by the process to kill. E.g `k 2742`

## 8.2 Disk full

For disk usage diagnostics and cleanup, please refer to the help of the `smd` command.

## 8.3 Run out of memory

If you get the following error when trying to start moshell:

```
gawk: fatal: cannot create child process for `/tmp/readlineXXXX_hhmmss'
(fork: Not enough space)".
```

It means that you do not have enough memory (i.e. RAM + swap space) on the machine.

Try running the command `top` on the Solaris box (it might not exist on the box though).

If you can run it, you'll see a line like this:

```
Memory: 512M real, 107M free, 333M swap in use, 2.0G swap free
```

A fundamental rule of Operating System management is that your swap space should also be > 2x the memory, so in this box we have 512Mb of RAM so we should have at least 1Gb of RAM (it started with just 256Mb of RAM - and we had this problem after opening a few sessions of MoShell). As a rule - if you are running MoShell on a Solaris box you should give it at least 512Mb of RAM and ideally 2Gb of swap space.

Luckily - there is an easy way to add new swap space - this is to make a new "swap file" on the disk (then you don't need to repartition everything). You can do this in Solaris by following these steps:

```
1. mkfile -v 2000m /usr/swapfile
```

This will make a 2Gb file **/usr/swapfile** to be used as our extra swap space. But it's not enabled as swap space yet.. To add it as swap space.

```
2. swap -a /usr/swapfile
```

This adds it in as swap space. But this is not permanent, next time you reboot the machine it'll disappear. You can make it permanent by adding the following line to (the end of) **/etc/vfstab**

```
3. /usr/swapfile - - swap - no -
```

See `man vfstab` for more details on **/etc/vfstab**

## 9 Offline Mode and Multi Mode

### 9.1 Offline Mode

The offline mode allows to run an moshell session against a set of logfiles of a node. When running moshell commands in offline mode, all information will be read from the logfiles, no communication taking place with the node.

The offline mode can be useful in several cases. For instance:

- when the node is not accessible directly, e.g. the 3rd line support need to examine the configuration and do not have remote access to the customer network.
- to save a snapshot of a node before an upgrade or configuration change, to use as reference information for comparison purposes.

The following logfiles are used in offline mode:

- the MO dump of the node. The MO dump consists of a printout showing all the attribute values of all the MOs of the node. It is taken with the command `dcgk`, `dcge`, `dcgm`, or `lt all;kget`. It can be in text format, gzipped format, or zip format. This file is used when running MO commands in offline mode (e.g. `get`, `set`, `st`, `inv`, `str`, etc). If MO dumps need to be taken from many nodes, then `mobatch` can be used instead, using the command: `mobatch <sitefile> 'lt all;kget'`. To run moshell against an MO dump, execute the following command from the unix prompt: `moshell /path/to/modump`

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- the CPP logs of the node. These files are located under `/c/logfiles` and can be fetched with the moshell command `lgf`, `dcgf`, or `dcgm`. These files are used when running the `lg` command in offline mode. E.g.  
`lgaevm -l /path/to/node_logfiles.zip`
- the ROP files of the node. These files are located in the `pm_data` folder of the node and can be fetched with the moshell command `pmrf`, `dcgf`, or `dcgm`. These files are used when running the commands `pmr` or `pmx` in offline mode. E.g.  
`pmx atmpport -l /path/to/node_ropfiles.zip`
- the CV or db.dat of the node, taken with `dcgm`, `cvget` or `ftget` command. This allows to use the `dbc` and `fro` commands in offline mode.
- the COLI printouts of the node, eg files called `<nodename>_dcg_<option>.log.gz` and taken with `dcg` command. This allows to use the COLI commands in offline mode.

It is possible to start moshell in offline mode by typing any of the following:

- moshell without arguments: MO commands will then not be available, only `lg/pmr/pmx` and a few others (`lmid`, `ip2d`, `h2d`, etc)
- moshell `<modump>`: MO commands will then be available and if the folder containing the MO-dump also contains the `ropfiles.zip/logfiles.zip` then it is not necessary to use the option `"-l"` when running `lg/pmr/pmx`.
- moshell `<offline.zip>`: the file `<nodename>_offline.zip` is generated by `dcgm` and contains the MO dump, CV, ropfiles, and logfiles. If the logfile `<nodename>_dcg_<option>.log.gz` is placed in the same folder as the `offline.zip` then also the COLI commands will be available.
- moshell `<dcgzipfile>`: in this case the `<dcgzipfile>` is the zipped directory containing all the files produced by the `dcg` command, ie, `modump.zip/logfiles.zip/ropfiles.zip/dcg_<option>.log.gz`.

The list of commands that are supported in offline mode can then be printed with the `h` command.

Type `h <command>` for information on a specific command.

## 9.2 SQL Mode

The SQL mode allows to run an offline moshell session against the configuration database of a node (db.dat or zipped CV).

- The db.dat file can be fetched using the `ftget` command, eg: `ftget /d/configuration/cv/<cvname>/db.dat`
- The CV can be fetched using the `cvget` command, eg: `cvget <cvname>`

To start moshell in SQL mode, use option `-d`, eg: `moshell -d <cv.zip>` or `moshell -d <db.dat>`

Moshell then opens an SQL client session to the file and loads all the MO data into memory.

To prevent loading MO data (for faster startup time), use the option `-v nomo=1`. Only sql commands will then be available.

Note that the MO data when read from a CV/db.dat only contains the MO configuration attributes, not the MO state attributes. During startup, moshell also performs a consistency check on the various SQL tables of the database, to detect if there are any inconsistencies or corruptions.

Currently not all moshell commands are supported in SQL mode, type `h` at the moshell prompt to see the list of supported moshell commands.

It is also possible to run sql commands directly, e.g. `sql select * from tables.`

The Polyhedra database viewer which is used to load the CV/db.dat is currently delivered in two versions:

- version 6.0, supported on solaris and linux, allows read and write to the db.dat, might not work on newest CPP releases (CPP9 and above)
- version 8.3, supported on solaris, linux, and cygwin, read-only (cannot write to db.dat). Supports all CPP releases.

By default, moshell uses the newest version but it is possible to use the old version by starting moshell with option `-v polyversion=0`.

## 9.3 Multi Mode

The multi mode allows an moshell session to be connect to several nodes at the same time.

The command syntax for starting moshell in multi mode is: `moshell -m <sitelist>|<sitefile>`

The sitelist consists of a comma separated list containing all the node names or ip/dns addresses.

The sitefile is a text file containing the list of nodes names or ipaddresses, on node per line.

Example:

|                                                                     |         |                        |           |                              |
|---------------------------------------------------------------------|---------|------------------------|-----------|------------------------------|
| Prepared (also subject responsible, if other)<br>EAB Finn Magnusson |         | No.<br>1553-CXC1328930 |           |                              |
| Document responsible/Approved<br>EAB Finn Magnusson                 | Checked | Date<br>2015-04-19     | Rev.<br>R | File<br>moshellUserGuide.tex |

- `moshell -m rnc2,10.1.128.17,rbs34,rxi2.ericsson.se,mgw3`
- `moshell -m /path/to/sitefile`

If node names are used, they must be defined in the ipdatabase. For more information about ipdatabase and sitefile, see the help of the mobatch utility by typing "mobatch" from the unix prompt.

To print the list of commands which are supported in multi-mode, type `h` at the moshell prompt.

More information about a specific command can be obtained by typing `h <command>`.

The multi mode is primarily geared towards commands that use the corba services CS/FM/PM.

Moshell commands that access the node via telnet/ssh/ftp/sftp are currently not supported in multi mode.

When moshell is running in multi mode, a prefix is appended in front of certain objects in order to distinguish between different nodes and MOM versions:

- the RDN/LDN in MO commands are prefixed with the string "Me=<nodename>".
- the scanner names in PM commands are prefixed with the node name.
- the MO class in mom/pmom command are prefixed with the MOM version.

MOM handling in multimode:

- at startup, the MOM version of each node is checked, so it is supported to connect to nodes running different MOM versions.
- it is also possible to skip the MOM check by parsing a MOM file with the parsemom command. Then all nodes will use the same MOM, which may have unexpected effects.

#### Known limitations in multi mode (to be fixed in a later release)

- `u+`, `emom`, `pset`: currently only work when all nodes have the same MOM version.
- `ul`: conversion of `.mos` to `.mo` script does not work correctly yet. Avoid using the `ul` command in multimode.
- `pol`: options (`c/h/s/u`) not yet supported in multimode. Syntax is: `pol <node>`. E.g: `pol rnc2`
- `pcr`: not yet supported in multi-mode. The future syntax will be: `pcr <node> <scannername>` etc...
- `getmom`: not yet supported in multimode.
- `re`: the "i" option is not yet supported in multimode.
- `diff`: `syntax2` (parameter audit and dump comparison) is not supported yet. Only `diff` between individual MOs is currently supported (`syntax1`).
- `pgets`: the "n" and "m" options are not yet supported.
- `lko`: not yet supported in multimode.

## 10 Revision History

The revision history has been moved into a file called **README** in the moshell directory.