Command Line Interface Specification

nVent SCHROFF
Guardian Management Gateway

Revision 1.0

April 7, 2020

nVent
Schroff GmbH

schroff.nVent.com

Doc.-No. :63972-385

Table of contents

1 Introduction

Important Note

	**Before using the Command Line Interface, please read the User Manual, article number 63972-383. The User Manual contains basic and advanced information about the Guardian Management Gateway and is necessary for understanding the CLI commands.**

.

The Guardian Management Gateway provides a Command Line Interface (CLI). After the user logs in (normally via SSH or Telnet) the command-line interpreter becomes available for that user in the interactive mode. The prompt contains the user name enclosed in curly braces.

```
login as: admin
Using keyboard-interactive authentication.
Password:
Last login: Thu Mar 26 07:47:25 2020 from 10.19.28.126
SMRC Command Line Interpreter
Connection from 10.19.22.51 as admin
RESTRICTED SERVICE AGREEMENT
---------- ------- ---------
Unauthorized access prohibited; all access and activities not explicitly
authorized by the management are unauthorized. All activities are monitored
and logged.

There is no privacy on this system.
Unauthorized access and activities or any criminal activity will be
reported to the appropriate authorities.

locale=25, en_US
Current language: English
CLI{admin}>
```

2 Help command

2.1.1 Syntax

help [<command>]

2.1.2 Purpose

This command issues help information for the specified command (possibly with subcommands). If the command is omitted, general help information about SMRC / iPDU is shown.

2.1.3 Example

```
CLI{admin}> help control
Manages controls
Subcommand include
  list [<res id>] - shows the information about all available controls
  info <res id> <ctrl number> - shows the RDR information of the specified
control
  <res id> <ctrl number> - shows the current value and attributes of the
         specified control
  <res id> <ctrl number> (auto | manual <value>) - sets the current mode and
         value to the specified control
  name <res id> <ctrl number> <name> - sets the name for the specified control
  default_name <res id> <ctrl number> - resets the name for the specified
control
CLI{admin}> help
Command set:
  action
  alarm
  config
  config_write
  control
  debug
  device
  discover
  expression
  factory_reset
  filter
  firewall
  global
  group
  help
```

```
  inventory
  ipdu
  ldap
  loginrestrictions
  modbus
  named_action
  netconf
  quit
  reachability
  reboot
  resource
  restart
  restricted_service_agreement
  role
  role_firewall
  sel
  sensor
session
  srvconf
  sslcert
  system
  terminate
  upgrade
  user
  verify_expression
  verify_filter
  wlan
```

3 Resource commands

The following commands deal with resources (devices):

3.1 Resource List

3.1.1 Syntax

resource list

resource

3.1.2 Purpose

This command shows the information about all available resources in the list format. The resource ID and resource tag are shown.

3.1.3 Example

```
CLI{admin}> resource list
(0): "Managed Sensors", Capabilities: {S|RDR|INV|RES}
(1002): "1-wire Sensor 1431800011", Capabilities: {S|RDR|INV|RES}
(2002): "Schroff RackChiller In-Row / 8:1", Capabilities: {S|RDR|INV|CNT|RES}
(2007): "SHX30 / 3:5", Capabilities: {S|RDR|INV|CNT|RES}
(3000): "MCB", Capabilities: {S|RDR|INV|CNT|RES}
```

3.2 Resource Show

3.2.1 Syntax

resource (info|show) <resource ID>

resource <resource ID>

3.2.2 Purpose

This command shows information about the resource with the specific resource ID. The following information is shown:

- resource ID
- resource tag (name)
- capabilities
- severity
- entity path

3.2.3 Example

```
CLI{admin}> resource 3000
Resource Id: 3000
Entity: {0xe0,0}
```

```
Capabilities: SENSOR|RDR|INVENTORY_DATA|CONTROL|RESOURCE
Resource Severity: MAJOR
Resource Name (Tag): "MCB"
Instrument counts: sensors: 9, controls: 4, inventory: 1
CLI{admin}> resource 1024
Resource Id: 1024
Entity: {0xd1,0}{0xd2,3}{0xd3,30}{0xd4,7892473}
Capabilities: SENSOR|RDR|INVENTORY_DATA|RESOURCE
Resource Severity: CRITICAL
Resource Name (Tag): "2D-00001E786DF9"
Instrument counts: sensors: 5, controls: 2, inventory: 1
```

3.3 Set resource name

3.3.1 Syntax

resource <resourceID> <name>

3.3.2 Purpose

This command allows a user to set user-defined name for the specified resource. This name is persistently stored.

3.3.3 Example

```
CLI{admin}>resource name 4007 OutletModule6
```

3.4 Set resource severity

3.4.1 Syntax

resource severity <resourceID> <severity>

```
<severity>::= critical | major | minor | info | ok
```

3.4.2 Purpose

This command allows a user to set severity for the specified resource.

3.4.3 Example

```
CLI{admin}>resource severity 4007 critical
```

3.5 Get list of persistent resources

3.5.1 Syntax

resource persistent [list]

3.5.2 Purpose

This command prints the list of persistent resource ID assignments, one by one.

3.5.3 Example

```
CLI{admin}>resource persistent
1000: "2D-00001E786DE6-BB"
1001: "1431800012"
```

3.6 Delete resource from the list of persistent resources

3.6.1 Syntax

resource persistent delete <resourceID>

3.6.2 Purpose

This command deletes the specified resource from the list of persistent resources. Currently, it works only for the 1-wire subsystem (resources 1000 - 1999) and the specified resource must be absent.

3.6.3 Example

```
CLI{admin}>resource persistent delete 1001
```

3.7 Delete resource

3.7.1 Syntax

resource remove <resourceID>

3.7.2 Purpose

This command deletes the specified resource. Currently, it works only for Modbus devices (resources 2000 - 2999) connected via TCP.

3.7.3 Example

```
CLI{admin}>resource remove 2001
```

4 Sensor commands

The following commands deal with sensors:

4.1 Sensor List

4.1.1 Syntax

sensor list [<resource ID>]

sensor

4.1.2 Purpose

This command shows the information about sensors in the list format. If the resource ID is specified, the sensor list for that resource is shown. Otherwise the command shows the list of all available sensors.

4.1.3 Example

```
CLI{admin}> sensor
Sensor(1000/1) "Temperature 1" Type: Temperature
Sensor(1000/2) "Temperature 2" Type: Temperature
Sensor(1000/3) "Humidity" Type: Humidity
Sensor(1000/4) "Pin 0 State" Type: Other FRU
Sensor(1000/5) "Pin 1 State" Type: Other FRU
Sensor(3000/1) "MCB Temperature" Type: Temperature
Sensor(3000/2) "MCB 12V" Type: Voltage
Sensor(3000/3) "Reboot Reason" Type: Reboot Reason
Sensor(3000/4) "USB1 Power Fault" Type: Operational State
Sensor(3000/5) "USB2 Power Fault" Type: Operational State
Sensor(3000/6) "Ext+12V Power Fault" Type: Operational State
Sensor(3000/7) "I2C_1 Bus Fault" Type: Operational State
Sensor(3000/8) "I2C_2 Bus Fault" Type: Operational State
Sensor(3000/9) "LAN Physical Link" Type: LAN
```

…

4.2 Sensor Description

4.2.1 Syntax

sensor info <resource ID> <sensor number>

4.2.2 Purpose

This command shows the RDR information of the specified sensor. For a managed sensor this command additionally reports the attach state of the sensor and its entity path.

4.2.3 Example

```
CLI{admin}> sensor info 2007 1
Name:            Valve Position
Entity path:     {0xd0,53009}{0x1e,773}
Type:            10 Cooling Device
Event Category: 127 OEM defined events
Enable Control: Disabled
Event Control:  All enabled
Event support:  0x0000
  Assert:       0xFFFF
  Deassert:     0xFFFF
Threshold:       Not supported
Data format:     Supported
  Reading type: UINT64
  Accuracy:     0.000000
  Units:        ? (Unspecified)
    Base unit: 0; Modifier: 0; Modifier unit: 0
  Percentage:   Yes
  Range:        0x18 (MIN, MAX)
    MIN:        (UINT64) 0
    MAX:        (UINT64) 100
Poll period: 0
Assertion Delay Count: 0
Severities: {}
Resolution: 1.000000
3+4 ms

CLI{admin}> sensor info 0 1
Name:            MCB Temperature
Entity path:     {0x17,1}
Attach state:    Attached (2)
Attached to:     Resource: 3000 sensor: 1 entity path: {0xe0,0}
Type:            1 Temperature
Event Category: 1 Threshold events
Enable Control: Disabled
Event Control:  All enabled
Event support:  0x003F
  Assert:       0xFFFF
```

```
  Deassert:      0xFFFF
Threshold:       Supported (Linear)
  Read mask:     0xF6
  Write mask:    0x00
Data format:     Supported
  Reading type: FLOAT64
  Accuracy:      0.000000
  Units:         C (Degrees C)
    Base unit: 1; Modifier: 0; Modifier unit: 0
  Percentage:    No
  Range:         0x1E (MIN, MAX, NORMAL MIN, NORMAL MAX)
    MIN:         (FLOAT64) -5.000000
    MAX:         (FLOAT64) 100.000000
    NORMAL MIN: (FLOAT64) 0.000000
    NORMAL MAX: (FLOAT64) 90.000000
Poll period: 3000
Assertion Delay Count: 0
Severities: {MINOR, MAJOR, CRITICAL, MINOR, MAJOR, CRITICAL}
Resolution: 0.100000
```

4.3 Sensor Data

4.3.1 Syntax

sensor data <resource ID> <sensor number>

sensor data <sensor number>

4.3.2 Purpose

This command shows the reading of the specified sensor on the specified resource. If the resource ID is omitted, resource ID 0 is used (the resource that handles managed sensors).

4.3.3 Example

```
CLI{admin}> sensor data 3
Reading: supported; type: FLOAT64; value: 23.250000
State: 0x0000
CLI{admin}> sensor data 1002 0
Reading: supported; type: FLOAT64; value: 23.312000
State: 0x0000
CLI{admin}> sensor data 3000 3
Sensor(3000/3): "Reboot Reason" Type: Reboot Reason
State: Soft Reboot (0x0004)
```

4.4 Sensor Assertion and Deassertion Event Masks

4.4.1 Syntax

sensor event_masks <resource ID> <sensor number> [<action> <assertion mask> <deassertion mask>]

<action> ::= add | remove

4.4.2 Purpose

This command shows or changes the Assertion and Deassertion Event masks of the specified sensor on the specified resource, depending on the presence of the additional arguments. If the *<action>* parameter is *add*, the other two parameters are added as a hexadecimal mask to the Assertion and Deassertion Event masks, respectively. If the *<action>* parameter is *remove*, the other two parameters are removed as a hexadecimal mask from the Assertion and Deassertion Event masks, respectively.

4.4.3 Example

```
CLI{admin}> sensor event_masks 1001 4
  Assert:       0x0003
  Deassert:     0x0000
CLI{admin}> sensor event_masks 1001 4 add 0x0 0x2
CLI{admin}> sensor event_masks 1001 4
  Assert:       0x0003
  Deassert:     0x0002
```

4.5 Sensor Enable Control flag

4.5.1 Syntax

sensor enable_control <resource ID> <sensor number> [<enable>]

<enable> ::= enable | disable

4.5.2 Purpose

This command shows or sets the Enable Control flag of the specified sensor on the specified resource, depending on the presence of the additional argument.

4.5.3 Example

```
CLI{admin}> sensor enable_control 3000 1
Enable Control: Enabled
CLI{admin}> sensor enable_control 3000 1 disable
```

4.6 Sensor Event Control flag

4.6.1 Syntax

sensor event_control <resource ID> <sensor number> [<enable>]

<enable> ::= enable | disable

4.6.2 Purpose

This command shows or sets the Event Control flag of the specified sensor on the specified resource, depending on the presence of the additional argument.

4.6.3 Example

```
CLI{admin}> sensor event_control 3000 1
Event Control: Enabled
CLI{admin}> sensor event_control 3000 1 disable
CLI{admin}> sensor event_control 3000 1
Event Control: Disabled
```

4.7 Sensor Reading, State, Event Mask, Threshods and Severities

4.7.1 Syntax

sensor dts <resource ID> <sensor number>

4.7.2 Purpose

This command shows the reading, the state, the event mask, the thresholds and severities of the specified sensor on the specified resource.

4.7.3 Example

```
CLI{admin}> sensor dts 3000 1
Sensor (3000/1): "MCB Temperature":
Reading: type: FLOAT64; value: 32.0
State: 0x0000
         Lower Minor -- Not set
         Lower Major -- type: FLOAT64; value: 5.000000
      Lower Critical -- type: FLOAT64; value: 0.000000
         Upper Minor -- Not set
         Upper Major -- type: FLOAT64; value: 65.000000
      Upper Critical -- type: FLOAT64; value: 70.000000
 Positive Hysteresis -- Not set
 Negative Hysteresis -- Not set
Event mask: 003f
Severities: {MINOR, MAJOR, CRITICAL, MINOR, MAJOR, CRITICAL}
CLI{admin}> sensor dts 1001 4
Sensor (1001/4): "Digital Input 2":
State: OFF (0x0001)
Event mask: 0003
Severities: {INFORMATIONAL, INFORMATIONAL}
```

4.8 Sensor attributes

4.8.1 Syntax

sensor attr <sensor number> list

sensor attr <sensor number> name <name>

sensor attr <sensor number> description <description>

sensor attr <sensor number> type <type> [<subtype>]

sensor attr <sensor number> subtype <subtype>

sensor attr <sensor number> X <number>

sensor attr <sensor number> Y <number>

sensor attr <sensor number> Z (<number> | <string>)

4.8.2 Purpose

This command manages sensor attributes for managed sensors. The subcommand *list* shows attributes for the designated sensor. The subcommands *name*, *description*, *type*, *subtype* assign corresponding attributes (specified as opaque strings by the user) to the specified sensor.

4.8.3 Example

```
CLI{admin}> sensor attr 5 name "Speed Fan"
CLI{admin}> sensor attr 5 description "Main Fan Speed"
CLI{admin}> sensor attr 5 type "Fan Speed" "Vert"
CLI{admin}> sensor attr 5 list
Sensor 5: "Speed Fan"
Description: "Main Fan Speed"
User Type: "Fan Speed" Subtype: "Vert"
X: "" Y: "" Z: "" ((symbolic))
```

4.9 Sensor threshold management

4.9.1 Syntax

sensor threshold [get] <resource ID> <sensor number> [<threshold name>...]

sensor threshold set <resource ID> <sensor number> <threshold name> (<value> | disable)

```
<threshold name> ::= ucr | umj | umn | lcr | lmj | lmn | phy | nhy
```

4.9.2 Purpose

This command manages sensor thresholds.

The subcommand *get* shows current values of the thresholds and/or hysteresis for the specified sensor. The list of threshold names can be specified; in that case only the specified thresholds are printed. If the threshold name list is empty, all threshold and hysteresis values are printed.

The subcommand *set* sets the specified threshold or hysteresis to the provided numeric value. When the term *disable* is present, this command disables the specified threshold or hysteresis.

The threshold and hysteresis names have the following meaning:

ucr = Upper Critical Threshold

umj = Upper Major Threshold

umn = Upper Minor Threshold

lcr = Lower Critical Threshold

lmj = Lower Major Threshold

lmn = Lower Minor Threshold

phy = Positive Hysteresis

nhy = Negative Hysteresis

4.9.3 Example

```
CLI{admin}> sensor threshold  1002 0
         Lower Minor -- type: FLOAT64; value: -128.000000
         Lower Major -- type: FLOAT64; value: -128.000000
      Lower Critical -- type: FLOAT64; value: -128.000000
         Upper Minor -- type: FLOAT64; value: 128.000000
         Upper Major -- type: FLOAT64; value: 128.000000
      Upper Critical -- type: FLOAT64; value: 128.000000
 Positive Hysteresis -- type: INT64; value:   0
 Negative Hysteresis -- type: INT64; value:   0
CLI{admin}> sensor threshold  1002 0 lmj
         Lower Major -- type: FLOAT64; value: -128.000000
CLI{admin}> sensor threshold set  1002 0 lmj -127
CLI{admin}> sensor threshold set  1002 0 lmj disable
```

4.10 Set sensor name

4.10.1 Syntax

sensor name <resource ID> <sensor number> <name>

4.10.2 Purpose

This command allows a user to set a user-defined name for a specified sensor. This name is persistently stored.

4.10.3 Example

```
CLI{admin}> sensor list 1021
Sensor(1021/1) "Temperature" Type: Temperature
Sensor(1021/2) "Humidity" Type: Humidity
```

```
CLI{admin}> sensor name 1021 2 "My humidity"
CLI{admin}> sensor list 1021
Sensor(1021/1) "Temperature" Type: Temperature
Sensor(1021/2) "My humidity" Type: Humidity
```

4.11 Set default sensor name

4.11.1 Syntax

sensor default_name <resource ID> <sensor number>

4.11.2 Purpose

This command restores a default name for a specified sensor. This name is persistently stored.

4.11.3 Example

```
CLI{admin}> sensor list 1021
Sensor(1021/1) "Temperature" Type: Temperature
Sensor(1021/2) "My humidity" Type: Humidity
CLI{admin}> sensor default_name 1021 2
Name set to "Humidity"
CLI{admin}> sensor list 1021
Sensor(1021/1) "Temperature" Type: Temperature
Sensor(1021/2) "Humidity" Type: Humidity
```

4.12 Show or set Assertion Delay Count

4.12.1 Syntax

sensor assert_delay <resource ID> <sensor number> [<count>]

4.12.2 Purpose

This command shows or sets the Assertion Delay Count for a specified sensor, depending on the presence of an optional argument. If there is no optional argument, this command shows the Assertion Delay Count. If the optional argument <*count*> is present, this argument specifies the Assertion Delay Count to be set. This attribute is persistently stored.

4.12.3 Examples

```
CLI{admin}> sensor assert_delay 3000 1 20
CLI{admin}> sensor assert_delay 3000 1
Assertion Delay Count: 20
```

4.13 Managed Sensors commands

4.13.1 Syntax

sensor manage <resource ID> <sensor number> [auto|noauto]

sensor attach <managed sensor number> <resource ID> <sensor number>

sensor detach <managed sensor number>

4.13.2 Purpose

These commands establish and remove the association between physical and managed sensors. Managed sensors are aliases of physical sensors located on resource *0*. For managed sensors, additional functionality is provided; for example, values of managed sensors can be aggregated and logged and additional user-defined attributes can be specified for the managed sensors.

Association between managed sensors and physical sensors is persistent and preserved across system reboot.

The subcommand *manage* creates a managed sensor on resource *0* for the specified physical sensor and reports its number in the CLI output. A sensor can be managed in automatic (default) or non-automatic mode. For a new association, automatic mode should be specified; non-automatic mode only restores a previously existing association between the specified physical sensor and some managed sensor.

The subcommand *attach* forcefully attaches the specified physical sensor to the specified managed sensor. It is not possible to use a managed sensor number that is already in use.

The subcommand *detach* terminates the association with a physical sensor for the specified managed sensor.

4.13.3 Example

```
CLI{admin}> sensor manage 1006 1 auto
Sensor 1 on resource 1006 is now managed as sensor 8 on resource 0
CLI{admin}> sensor data 0 8
Sensor(0/8): "Temperature" Type: Temperature
Reading: type: FLOAT64; value: 28.000000
State: 0x0010
CLI{admin}> sensor detach 8
CLI{admin}> sensor data 0 8
Sensor(0/8):
client -- DoRpc on SensorReadingGet returned NOT_PRESENT (-1011)
CLI{admin}> sensor attach 8 1006 1
CLI{admin}> sensor data 0 8
Sensor(0/8): "Temperature" Type: Temperature
Reading: type: FLOAT64; value: 27.812000
State: 0x0010
```

4.14 Sensor log commands

4.14.1 Syntax

sensor log enable

sensor log disable

sensor log status

sensor log period [<value>]

sensor log [show] <sensor number>

4.14.2 Purpose

This command manages sensor log, which is used to collect aggregate values (minimum, maximum, average and dispersion) for managed sensors. The logged managed sensors are periodically sampled, aggregate values are evaluated over these samples and periodically stored into the log. The log for each sensor operates as a ring buffer of a certain size, and the newly accumulated values overwrite the oldest ones.

The subcommand *enable* turns on sensor logging, the subcommand *disable* turns it off, and the subcommand *status* shows whether logging is currently enabled or disabled.

The subcommand *period* allows to change the accumulation period for the log, or shows the current period if *<value>* is omitted.

The subcommand *show* shows the contents of the log for the specified sensor. All entries from the corresponding ring buffer are shown, in the order from the oldest to the newest, together with their timestamps. For each entry, the following information is shown, relative to the corresponding accumulation period: the timestamp when the entry was written, the sample count, the minimum, the maximum, the average, the dispersion and the sampled event state masks ORed together.

4.14.3 Example

```
CLI{admin}> sensor log enable

Operation completed successfully

CLI{admin}> sensor log status

Sensor log Enabled

CLI{admin}> sensor log period 200

Operation completed successfully

CLI{admin}> sensor log period

Sensor log accumulation period: 200 seconds

CLI{admin}> sensor log show 3

ID | Timestamp                | Read  |  ES   |   Average   |     Min     |
Max    |    Disp    | Acc ES

    |                          | count | count |             |             |
|            |

----|--------------------------|-------|-------|-------------|-------------|------
------|------------|-------

 05 | Fri May  5 12:13:00 2017 |   12  |   12  |  23.229000  |  23.187000  |
23.250000 |   0.029698 | 0x0000

 06 | Fri May  5 12:13:30 2017 |   12  |   12  |  23.239333  |  23.187000  |
23.312000 |   0.043010 | 0x0000

 07 | Fri May  5 12:14:00 2017 |   12  |   12  |  23.255167  |  23.250000  |
23.312000 |   0.017136 | 0x0000

 08 | Fri May  5 12:14:30 2017 |   12  |   12  |  23.270667  |  23.250000  |
23.312000 |   0.029227 | 0x0000

 09 | Fri May  5 12:15:00 2017 |   12  |   12  |  23.317333  |  23.250000  |
23.375000 |   0.030877 | 0x0000
```

```
 10 | Fri May  5 12:15:30 2017 |   12  |   12  |  23.317417 |  23.250000 |
23.375000 |   0.040045 | 0x0000
 11 | Fri May  5 12:16:00 2017 |   12  |   12  |  23.291417 |  23.250000 |
23.375000 |   0.038843 | 0x0000
 12 | Fri May  5 12:16:30 2017 |   12  |   12  |  23.260333 |  23.250000 |
23.312000 |   0.023106 | 0x0000
 13 | Fri May  5 12:17:00 2017 |   12  |   12  |  23.265417 |  23.187000 |
23.312000 |   0.037073 | 0x0000
 14 | Fri May  5 12:17:30 2017 |   12  |   12  |  23.296500 |  23.250000 |
23.312000 |   0.026847 | 0x0000
 15 | Fri May  5 12:18:00 2017 |   12  |   12  |  23.275833 |  23.250000 |
23.312000 |   0.030566 | 0x0000
 00 | Fri May  5 12:18:30 2017 |   12  |   12  |  23.260250 |  23.187000 |
23.312000 |   0.034448 | 0x0000
 01 | Fri May  5 12:19:00 2017 |   12  |   12  |  23.275833 |  23.250000 |
23.312000 |   0.030566 | 0x0000
 02 | Fri May  5 12:19:30 2017 |   12  |   12  |  23.301750 |  23.250000 |
23.375000 |   0.034448 | 0x0000
 03 | Fri May  5 12:20:00 2017 |   12  |   12  |  23.338333 |  23.250000 |
23.375000 |   0.040142 | 0x0000


Log size: 15 entries
First entry: 5; last entry: 3
```

4.15 Sensor polling

4.15.1 Syntax

sensor poll [get] <resource ID> <sensor number>

sensor poll set <resource ID> <sensor number> <poll period>

4.15.2 Purpose

This command manages the sensor polling. The subcommand *get* show the current values of the Poll Period, in milliseconds, and the Assertion Delay Count for the specified sensor. The subcommand *set* sets the Poll Period value, in milliseconds, for the specified sensor.

4.15.3 Example

```
CLI{admin}> sensor poll 3007 2
Poll period: 3000
Assertion Delay Count: 0
CLI{admin}> sensor poll set 3007 2 2000
CLI{admin}> sensor poll get 3007 2
Poll period: 2000
Assertion Delay Count: 0
```

4.16 Reset sensor configuration

4.16.1 Syntax

sensor reset <resource ID> <sensor number>

4.16.2 Purpose

This command resets the configuration of the specified physical sensor.

4.16.3 Example

```
CLI{admin}> sensor reset 2007 2
```

4.17 Set sensor severity

4.17.1 Syntax

sensor severity <resource ID> <sensor number> <state mask> <severity1> … <severityN>

```
<severity>::= crt | mjr | mnr | info | ok
```

4.17.2 Purpose

This command applies to discrete sensors (that report a current state rather than a numeric value). It assigns severities to specific event states. The parameter *<state mask>* is an unsigned two-byte integer and specifies the event states to be set. The number of additional parameters *<severity1>*, …, *<severityN>* must be equal to the number of *1* bits in *<state_mask>* parameter, N > 0.

4.17.3 Example

Resource ID = 1001, sensor number = 4, state mask = 3.

```
CLI{admin}> sensor severity 1001 4 3 info crt
```

4.18 User-defined sensors

4.18.1 List of all user-defined sensor types

4.18.1.1 Syntax

sensor usertype [list]

4.18.1.2 Purpose

This command prints out a list of all user-defined sensor types. The user types “Normally Closed” and “Normally Open” are pre-defined and always present.

4.18.1.3 Example

```
CLI{admin}> sensor usertype
User defined sensor types:
    320(0x140): Normally Closed
        State 0: Closed (OK)
        State 1: Open (CRITICAL)
```

```
    321(0x141): Normally Open
        State 0: Closed (CRITICAL)
        State 1: Open (OK)

    448(0x1C0): MyType
        State 0: A (INFORMATIONAL)
        State 1: B (INFORMATIONAL)

454(0x1C6): DoorLock
        State 0: DoorShut (OK)
        State 1: DoorOpen (OK)
        State 2: DoorBrocken (CRITICAL)
```

4.18.2 Get description of user-defined sensor type

4.18.2.1 Syntax

sensor usertype info [(<type_id> | <type_name>)]

4.18.2.2 Purpose

This command prints out information for a user defined sensor type specified by name or by type id. If no optional argument is specified, the output is same as *sensor usertype list*.

4.18.2.3 Example

```
CLI{admin}> sensor usertype info DoorLock
454(0x1C6): DoorLock
        State 0: Shut (INFORMATIONAL)
        State 1: Open (MAJOR)
```

4.18.3 Add user-defined sensor type

4.18.3.1 Syntax

sensor usertype add <type_name> [<state_1_name> <severity1> [<state_2_name> <severity2> ... [<state_n_name> <severity_n>]..]]

4.18.3.2 Purpose

This command creates a new user-defined sensor type with *<type_name>* name and several named states and corresponding severities (up to 16), sensor types with no states are allowed.

4.18.3.3 Example

```
CLI{admin}> sensor usertype add DoorLock Shut info Open mjr
```

```
New user type "DoorLock" has been added with code 454(0x1C6)
```

4.18.4 Delete user-defined sensor type

4.18.4.1 Syntax

sensor usertype delete (<type_id> | <type_name>)

4.18.4.2 Purpose

This command deletes an existing user-defined sensor type specified either by name or by type id.

4.18.4.3 Example

```
CLI{admin}> sensor usertype delete DoorLock
```

4.18.5 Modify user-defined sensor type

4.18.5.1 Syntax

sensor usertype update (<type_name>|<type_id>) [<state_1_name> <severity1> [<state_2_name> <severity2> ... [<state_n_name> <severity_n>]..]

```
<severity>::= crt | mjr | mnr | info | ok
```

4.18.5.2 Purpose

This command changes the list of states of an existing user-defined sensor type specified either by *<type_name>* or by *<type_id>*. The number of named states should not exceed 16, sensor types with no states are allowed.

4.18.5.3 Example

```
CLI{admin}> sensor usertype update DoorLock DoorShut ok DoorOpen ok DoorBrocken
crt
```

4.18.6 Assign sensor type

4.18.6.1 Syntax

sensor type <resource ID> <sensor number> (<type_name>|<type_id> <event category number> [<state mask> <severity1> <severity2> ... <severityN>])

4.18.6.2 Purpose

This command assigns the sensor type specified either by *<type_name>* or by *<type_id>* to sensor. When the sensor type specified either by *<type_name>* or by *<type_id>* is a user-defined sensor type, the parameter *<event category number>* should be *0x7E* (Sensor-specific events). The optional parameter *<state mask>* is an unsigned two-byte integer and specifies the event states to be set. The number of additional parameters *<severity1>*, …, *<severityN>* must be equal to the number of *1* bits in *<state_mask>* parameter, N > 0. These optional parameters may be used if the sensor is specified by *<type_id>*.

4.18.6.3 Example

Resource ID = 0, sensor number = 5, sensor type = “DoorLock” (user-defined).

```
CLI{admin}> sensor type 0 5 DoorLock
```

Resource ID = 3000, sensor number = 1, sensor type = 1 (Temperature), event category = 1 (Threshold events), state mask = 0x3.

```
CLI{admin}> sensor type 3000 1 1 1 0x3 mjr crt
```

5 Control commands

The following commands deal with controls:

5.1 Control List

5.1.1 Syntax

control list [<resource ID>]

control list

5.1.2 Purpose

This command shows the information about controls in the list format. If the resource ID is specified, the control list for that resource is shown. Otherwise the command shows the list of all available controls on all resources.

5.1.3 Example

```
CLI{admin}> control
Control(1002/5): "Pin 0 Control", Type: DIGITAL, Output: GENERIC
Control(1002/6): "Pin 1 Control", Type: DIGITAL, Output: GENERIC
Control(3000/1): "Buzzer", Type: DIGITAL, Output: DRY_CONTACT_CLOSURE
Control(3000/2): "USB1 Power Fault Reset", Type: DIGITAL (Write Only), Output:
ACTIVATE
Control(3000/3): "USB2 Power Fault Reset", Type: DIGITAL (Write Only), Output:
ACTIVATE
Control(3000/4): "Ext+12V Power Fault Reset", Type: DIGITAL (Write Only),
Output: ACTIVATE
…
```

5.2 Control Description

5.2.1 Syntax

control info <resource ID> <control number>

5.2.2 Purpose

This command shows the RDR information of the specified control.

5.2.3 Example

```
CLI{admin}> control info 3000 1
Name:           Buzzer
Entity path:    {0xe0,0}
Type:           0 Digital
Output type:    5 Audible
Default:
  State:        0 OFF
```

```
  Mode:          Manual (Read only)
CLI{admin}> control info 2007 4
Name:            Max Cooling ON/OFF
Entity path:     {0xd0,53009}{0x1e,773}
Type:            0 Digital
Output type:     2 Fan speed
Default:
  State:         0 OFF
```

5.3 Control Get Value

5.3.1 Syntax

control <resource ID> <control number>

5.3.2 Purpose

This command shows the current value and attributes of the specified control. Besides the value, the command shows control name, type and output type, current and default mode (auto or manual), the default value, the actual state or the actual value.

5.3.3 Example

```
  CLI{admin}> control 2000 2
  Control(2000/2)
    Name:          Set Temperature
    Type:          ANALOG
    Output:        OEM (266)
    Mode:          Manual
    Default Mode: Manual (Read Only)
    Default Data: 0 (min = 18, max = 40)
    Units:         C (Degrees C)
      Base unit: 1; Modifier: 0; Modifier unit: 0
    Percentage:    No
    Actual value: 24
```

5.4 Control Set Value

5.4.1 Syntax

control <resource ID> <control number> auto

control <resource ID> <control number> manual <value>

<value> ::= <digital value> | <discrete value> | <analog value>

<digital value> ::= on | off | pulseon | pulseoff

<discrete value> ::= <number>

<analog value> ::= <number>

5.4.2 Purpose

This command sets the current mode and value to the specified control. The mode can be auto or manual. If the mode is set to manual, the value should also be specified, that is assigned to the control. For a digital control, the value can be one of the predefined constants `on`, `off`, `pulseon` or `pulseoff`. For a discrete or analog control, a number is specified for the value.

5.4.3 Example

```
CLI{admin}> control  4000 50 manual on
Operation completed successfully
```

5.5 Set control name

5.5.1 Syntax

control name <resource ID> <control number> <name>

5.5.2 Purpose

This command allows a user to set a user-defined name for a specified control. This name is persistently stored.

5.5.3 Example

```
CLI{admin}> control list 1012
Control(1002/5): "Pin 0 Control", Type: DIGITAL, Output: GENERIC
Control(1002/6): "Pin 1 Control", Type: DIGITAL, Output: GENERIC
CLI{admin}> control name 1012 5 "Door Lock"
CLI{admin}> control list 1012
Control(1012/5): "Door Lock", Type: DIGITAL, Output: GENERIC
Control(1012/6): "Pin 1 Control", Type: DIGITAL, Output: GENERIC
```

5.6 Set default control name

5.6.1 Syntax

control default_name <resource ID> <control number>

5.6.2 Purpose

This command restores a default name for a specified control. This name is persistently stored.

5.6.3 Example

```
CLI{admin}> control list 1012
Control(1002/5): "Door Lock", Type: DIGITAL, Output: GENERIC
Control(1002/6): "Pin 1 Control", Type: DIGITAL, Output: GENERIC
CLI{admin}> control default_name 1012 5
Name set to "Pin 0 Control"
CLI{admin}> control list 1012
```

```
Control(1012/5): "Pin 0 Control", Type: DIGITAL, Output: GENERIC
Control(1012/6): "Pin 1 Control", Type: DIGITAL, Output: GENERIC
```

6 Inventory commands

The following commands show inventory data information that is associated with resources. Currently in the SMRC / iPDU only two resources contain inventory data:

- The main resource (resource 0) has the system inventory associated with it. It contains information about the physical SMRC / iPDU device and firmware (device human-readable name, product type, serial number, firmware version).

- The Inlet module resource. The corresponding inventory contains iPDU configuration information and calibration data for voltage, current and energy sensors.

Inventory data is treated by the CLI as read-only.

6.1 Inventory List

6.1.1 Syntax

inventory [list [<resource ID>]]

6.1.2 Purpose

This command list all inventory data objects present in the system (if *<resource ID>* is omitted) or on the specific resource.

6.1.3 Example

```
CLI{admin}> inventory
Inventory (0/0): "System Inventory"; READ_ONLY; 3 areas
Inventory (1002/0): "Inventory"; READ_ONLY; 2 areas
Inventory (3000/0): "Inventory"; READ_ONLY; 4 areas
```

6.2 Show inventory data

6.2.1 Syntax

inventory [(info|show)] <resource ID> [<inventory ID>]

6.2.2 Purpose

This command shows the contents (data) of the specific inventory, including all its areas and fields. nVent IPDU Configuration record, nVent Outlet Power Metering Calibration Parameters record, LCD Calibration Parameters record and 1-Wire Device Identification record are parsed. Default value of the parameter *<inventory ID>* is *0*.

6.2.3 Example

```
CLI{admin}> inventory show 0 0
Inventory: 0; Update Count: 0; READ_ONLY; 2 areas
    Area: 0; Type: BOARD_INFO; READ_ONLY; 6 fields
        Field: 0; Type: MFG_DATETIME; READ_ONLY; "2019.06.12 10:00:00"
        Field: 1; Type: MANUFACTURER; READ_ONLY; ""
        Field: 2; Type: PRODUCT_NAME; READ_ONLY; ""
        Field: 3; Type: SERIAL_NUMBER; READ_ONLY; ""
        Field: 4; Type: PART_NUMBER; READ_ONLY; ""
```

```
        Field: 5; Type: FILE_ID; READ_ONLY; ""
    Area: 1; Type: PRODUCT_INFO; READ_ONLY; 7 fields
        Field: 0; Type: MANUFACTURER; READ_ONLY; "nVent, SCHROFF"
        Field: 1; Type: PRODUCT_NAME; READ_ONLY; "Guardian Management Gateway"
        Field: 2; Type: PART_NUMBER; READ_ONLY; "123456789"
        Field: 3; Type: PRODUCT_VERSION; READ_ONLY; ""
        Field: 4; Type: SERIAL_NUMBER; READ_ONLY; "0000000000AB"
        Field: 5; Type: ASSET_TAG; READ_ONLY; ""
Field: 6; Type: FILE_ID; READ_ONLY; "6399860552.bin"
```

7 Event Filter and Action commands

Event filters are the mechanism that allows user to specify custom reaction to events. Each filter has a predicate expression, that evaluates on each event that happens, and if the expression evaluates to *TRUE* (non-zero), the event passes the filter. In that case, a list of actions associated with the event filter, is executed with the event passed as the parameter to the actions.

Each action has type, disposition and text parameters.

The supported action types include:

- evaluate an expression over the target event (with possible side effects)
- run a CLI command
- send an e-mail message to a specified recipient list
- log the event in the system log
- send an SNMP trap

The action disposition determines whether the specific action should be run depending on success or failure of the previous action.

The action parameters may comprise the expression to evaluate, the command to execute or the list of e-mail recipients to receive the message.

Periodic expressions are expressions that are registered to run periodically, every N seconds. Periodic expressions are similar in structure to event filters; they have an expression which is tested when the periodic expression is invoked, and if the expression yields *TRUE*, the associated actions are run. Actions can be attached to periodic expressions in the same way as to the event filters.

7.1 Filter List

7.1.1 Syntax

filter list [<name>]

filter show [<name>]

filter

7.1.2 Purpose

This command shows the information about existing event filters. If the *<name>* is specified, information about the event filter with that name is shown. Otherwise information about all known event filters is shown.

For each event filter, the following information is shown: the filter name, the predicate (expression that evaluates to *TRUE* for events that pass the filter) and the list of actions for the filter. If a named action list is assigned to the filter, the name of the list is printed.

For each action, the action type, disposition and parameters are shown.

7.1.3 Example

```
CLI{admin}> filter show
Filter "f2": " severity<=1"
Action list ("HighSeverity"):
```

```
    0: Always: Syslog: ""
    1: Always: MQTT message: ""
CLI{admin}> filter list
Filter "TestFilter": "resource==1001 && sensor_number==2 && assertion==1"
Action list:
    0: Always: Expression: "CONTROL(4002,300)=1"

Filter "NewFilter": "resource==1001 && sensor_number==2 && assertion==0"
Action list:
    0: If successful: Expression: "CONTROL(4002,300)=0
```

7.2 Filter Add

7.2.1 Syntax

filter add <name> <expression>

7.2.2 Purpose

This command creates a new event filter with the specified name and predicate expression. The filter name must be unique within filter and periodic expression name space.

The event filter is created with an empty action list. Actions can be added to the filter later via the *action add* command.

7.2.3 Example

```
CLI{admin}> filter add TestFilter 3
Operation completed successfully
```

7.3 Filter Delete

7.3.1 Syntax

filter delete <name>

filter remove <name>

7.3.2 Purpose

This command deletes the event filter with the specified name, including all associated actions.

7.3.3 Example

```
CLI{admin}> filter delete TestFilter
Operation completed successfully
```

7.4 Action List

7.4.1 Syntax

action list <filter name>

7.4.2 Purpose

This command shows the information about existing actions associated with the specified event filter (or periodic expression). If a named action list is assigned to the filter, the name of the list is printed.

For each action, the action type, disposition and parameters are shown.

7.4.3 Example

```
CLI{admin}> action list TestFilter
Action list ("TempAlarm"):
    0: Always: SNMP Trap: "192.168.1.107"
    1: If unsuccessful: Maximize Cooling: "2001,2002"
```

7.5 Action Add

7.5.1 Syntax

action add <filter name> <disposition> <action type> <parameters>

```
<disposition> ::= always | success | failure
<action type> ::= expression | command | sendmail | syslog | snmptrap |
cooling_on | cooling_off | max_cooling | mqtt
```

7.5.2 Purpose

This command creates a new action for the event filter (or periodic expression) with the specified name. The new action has the specified disposition, action type and parameters.

An action with the disposition `always` runs regardless of the previous action result. An action with the disposition `success` runs only if the previous action was successful, and the action with the disposition `failure` runs only if the previous action was unsuccessful.

The action parameters have the following meaning depending on the action type:

- For the action `expression`, this is the actual expression to evaluate, expressions are evaluated in units defined by the global settings.
- For the action `command`, this is the SMR CLI command
- For the action `sendmail`, the parameters comprise the list of recipients, the message subject and the message body, separated by *\n* separator (literally). The host name of the device and the event that initiates the action will be included in the message body.
- For the action `syslog`, the parameters are not needed and are ignored when specified
- For the action `snmptrap`, the parameters contain the trap destination address
- For the action `cooling_on`, the cooling algorithm is enabled for the specified resources.
- For the action `cooling_off`, the cooling algorithm is disabled for the specified resources.
- For the action `max_cooling`, the fans are set to the maximum level for the specified resources.
- For the action *mqtt*, MQTT message is published

On successful completion, the command shows the number of the new action in the list of actions for the event filter (or periodic expression). This number can be used later to selectively update or delete the action

7.5.3 Example

```
CLI{admin}> action add TestFilter always expression 5
Added as action 0
CLI{admin}> action add TestFilter always syslog
Added as action 1
CLI{admin}> action add TestFilter always snmptrap 192.168.1.107
Added as action 2
CLI{admin}> action add NewFilter failure max_cooling 2001,2002
Added as action 6
CLI{admin}> action add AnotherFilter success command 'sel clear'
Added as action 0
CLI{admin}> action add AnotherFilter always sendmail "urgent@mysite.com\nRequest
for maintenance\nLow Voltage"
Added as action 1
```

7.6 Action Update

7.6.1 Syntax

action update <filter name> <action number> <disposition> <action type> <parameters>

action set <filter name> <action number> <disposition> <action type> <parameters>

7.6.2 Purpose

This command updates the existing action (with the specified action number) in the list of actions for the event filter (or periodic expression) with the specified name.

The disposition, type and parameters of the specified action are replaced with the disposition, type and parameters specified on the command line.

7.6.3 Example

```
CLI{admin}> action update TestFilter 0 always expression 7
Action 0 has been updated
CLI{admin}> action set TestFilter 0 always command 'sel clear'
Action 0 has been updated
```

7.7 Action Delete

7.7.1 Syntax

action delete <filter name> <action number>

action remove <filter name> <action number>

7.7.2 Purpose

This command deletes the specified action (by the action number) in the list of actions for the event filter (or periodic expression) with the specified name.

7.7.3 Example

```
CLI{admin}> action delete TestFilter 0
Action 0 has been removed
```

7.8 Assign named action list to filter/periodic expression

7.8.1 Syntax

action assign <filter name> <action list name>

7.8.2 Purpose

This command assigns the named action list specified by its name to the event filter/periodic expression specified by its name.

7.8.3 Example

```
CLI{admin}> action assign TestFilter TempAlarm
```

7.9 Assign anonymous action list to filter/periodic expression

7.9.1 Syntax

action unassign <filter name>

7.9.2 Purpose

This command assigns an anonymous action list to the event filter/periodic expression specified by its name.

7.9.3 Example

```
CLI{admin}> action unassign TestFilter
```

7.10 Expression

7.10.1 Syntax

expression <expression>

7.10.2 Purpose

This command evaluates an arbitrary expression and prints the result.

7.10.3 Example

```
CLI{admin}> expression 7+4
Result: INT64: 11
CLI{admin}> expression '$abc'
Result: String: "123"
```

7.11 Periodic expression

7.11.1 Show

7.11.1.1 Syntax

periodic (show|list) [<name>]

7.11.1.2 Purpose

This command shows the information about existing periodic expressions. If the *<name>* is specified, information about the periodic expression with that name is shown. Otherwise information about all known periodic expressions is shown.

For each periodic expression, the following information is shown: the periodic expression name, the predicate (expression), period and the list of actions that are called if the predicate is *TRUE*.

For each action, the action type, disposition and parameters are shown.

7.11.1.3 Example

```
CLI{admin}> periodic list

Periodic expression "Simple": "resource==1001 && sensor_number==2 &&
assertion==0"; Period: 3000 sec

Action list:

    0: Always: SNMP Trap: "192.168.1.107"
```

7.11.2 Add

7.11.2.1 Syntax

periodic add <name> <expression> <period>

7.11.2.2 Purpose

This command creates a new periodic expression with the specified name and predicate expression. The periodic expression name must be unique within filter and periodic expression name space. The parameter *<period>* value is in seconds. Periodic expressions are evaluated in units defined by the global settings.

The periodic expression is created with an empty action list. Actions can be added to the periodic later via the *action add* command.

7.11.2.3 Example

```
CLI{admin}>periodic add Simple "resource==1001 && sensor_number==2 &&
assertion==0" 3000

Operation completed successfully
```

7.11.3 Delete

7.11.3.1 Syntax

periodic delete <name>

7.11.3.2 Purpose

This command deletes the periodic expression with the specified name, including all associated actions.

7.11.3.3 Example

```
CLI{admin}> periodic delete Simple
Operation completed successfully
```

7.12 Named action lists

7.12.1 Show

7.12.1.1 Syntax

named_action list [<name>]

7.12.1.2 Purpose

This command prints information about all the known named action lists. If the *<name>* is specified, information about the named action list with that name is shown.

7.12.1.3 Example

```
CLI{admin}> named_action list
Named action list "TempAlarm":
Action list:
    0: Always: SNMP Trap: "192.168.1.107"
    1: If unsuccessful: Maximize Cooling: "2001,2002"
Named action list "VoltageFailure":
Action list:
    0: Always: SNMP Trap: "192.168.1.108"
    1: If successful: Command: "sel clear"
```

7.12.2 Create named list

7.12.2.1 Syntax

named_action create <name>

7.12.2.2 Purpose

This command creates a new named action list. The newly created list is empty.

7.12.2.3 Example

```
CLI{admin}> named_action create VoltageFailure
```

7.12.3 Delete named list

7.12.3.1 Syntax

named_action remove_list <name>

named_action delete_list <name>

7.12.3.2 Purpose

This command deletes the named action list specified by the parameter *<name>*.

7.12.3.3 Example

```
CLI{admin}> named_action delete VoltageFailure
```

7.12.4 Add action

7.12.4.1 Syntax

named_action add <name> <disposition> <action type> <parameters>

```
<disposition> ::= always | success | failure
```

```
<action type> ::= expression | command | sendmail | syslog | snmptrap |
cooling_on | cooling_off | max_cooling | mqtt
```

7.12.4.2 Purpose

This command adds a new action for the named action list with the specified name. The new action has the specified disposition, action type and parameters. The parameters of the command are fully similar to the parameters of the *action add* command (section 7.5).

7.12.4.3 Example

```
CLI{admin}> named_action add TempAlarm always snmptrap 192.168.1.107
Added as action 0
```

7.12.5 Update action

7.12.5.1 Syntax

named_action update <name> <index> <disposition> <action type> <parameters>

named_action set <name> <index> <disposition> <action type> <parameters>

```
<disposition> ::= always | success | failure
```

```
<action type> ::= expression | command | sendmail | syslog | snmptrap |
cooling_on | cooling_off | max_cooling | mqtt
```

7.12.5.2 Purpose

This command updates the existing action (with the specified action number) in the named list of actions with the specified name.

The disposition, type and parameters of the specified action are replaced with the disposition, type and parameters specified on the command line. The parameters of the command are fully similar to the parameters of the *action update* command (section 7.6).

7.12.5.3 Example

```
CLI{admin}> named_action update TempAlarm 1 failure max_cooling 2001,2002
Action 1 has been updated
```

7.12.6 Delete action

7.12.6.1 Syntax

named_action delete <name> <index>

named_action remove <name> <index>

7.12.6.2 Purpose

This command deletes the existing action (with the specified action number) in the named list of actions with the specified name.

The parameters of the command are fully similar to the parameters of the *action delete* command (section 7.7).

7.12.6.3 Example

```
CLI{admin}> named_action delete TempAlarm 1
Action 1 has been removed
```

7.13 Verify expression for sensor

7.13.1 Syntax

verify_expression <resource ID> <sensor number> <expression>

7.13.2 Purpose

This command verifies whether an expression is applicable to a sensor.

7.13.3 Example

```
CLI{admin}>verify_expression 3000 1 "resource==3000 && assertion==1"
Applicable
CLI{admin}>verify_expression 3000 1 "resource==1000 && assertion==1"
Not applicable
```

7.14 Verify filter for sensor

7.14.1 Syntax

verify_filter <resource ID> <sensor number> <expression>

7.14.2 Purpose

This command verifies whether a filter is applicable to a sensor.

7.14.3 Example

```
CLI{admin}>verify_filter 3000 1 Testfilter
Applicable
```

8 Event Log commands

These commands show the information about the system event log (SEL) and contents of the system event log.

8.1 Show event log information

8.1.1 Syntax

sel info

8.1.2 Purpose

This command shows information about the event log. The information includes:

- the current and maximum number of entries in the event log
- whether the event log is enabled
- the current time reported by the event log
- the last update time
- the overflow flag

8.1.3 Example

```
CLI{admin}> sel info
EventLog: entries = 7, size = 500, enabled = 1
Update  Time: 2017-05-06 19:55:49
Current Time: 2017-05-06 20:25:13
Overflow Action: Overwrite
Overflow State: FALSE
CLI{admin}> sel info
EventLog: entries = 10000, size = 10000, enabled = 1
Update  Time: 2017-12-05 08:25:53
Current Time: 2017-12-05 08:25:53
Overflow Action: Overwrite
Overflow State: FALSE
```

8.2 Show event log entries

8.2.1 Syntax

sel

sel full [<entry number>]

sel [short] [<entry number>]

8.2.2 Purpose

This command shows the entries of the event log. If the entry number is specified, the entries are shown starting with the specified number, otherwise the whole event log is dumped. The output format for each entry can be full or short (full by default). In full format, the amount of information shown for each entry is substantially larger.

8.2.3 Example

```
CLI{admin}> sel full 9990
EventLog: entries = 10000, size = 10000, enabled = 1
Update  Time: 2017-12-21 12:26:03
Current Time: 2017-12-21 12:49:34
Overflow Action: Overwrite
Overflow State: TRUE

#9990 (next: #9991)
EntryId: 9990
Timestamp: 2017-12-21 12:25:48
  Event Type: SENSOR
  Event Resource ID: 1019
  Event Timestamp: 2017-12-21 12:25:47
  Event Severity: CRITICAL
  Event Sensor Num: 2 (2 hex)
  Event Sensor Type: Other FRU
  Event Sensor Category: Generic state events
  Event Sensor Assertion: TRUE
  Event Sensor State: Unspecified
  Optional Data: None

#9991 (next: #9992)
EntryId: 9991
Timestamp: 2017-12-21 12:25:48
  Event Type: SENSOR
  Event Resource ID: 1020
  Event Timestamp: 2017-12-21 12:25:47
  Event Severity: CRITICAL
  Event Sensor Num: 1 (1 hex)
  Event Sensor Type: Other FRU
  Event Sensor Category: Generic state events
  Event Sensor Assertion: TRUE
  Event Sensor State: Unspecified
```

```
  Optional Data: None

#9992 (next: #9993)
EntryId: 9992
Timestamp: 2017-12-21 12:25:48
  Event Type: SENSOR
  Event Resource ID: 1020
  Event Timestamp: 2017-12-21 12:25:47
  Event Severity: CRITICAL
  Event Sensor Num: 2 (2 hex)
  Event Sensor Type: Other FRU
  Event Sensor Category: Generic state events
  Event Sensor Assertion: TRUE
  Event Sensor State: Unspecified
  Optional Data: None

#9993 (next: #9994)
EntryId: 9993
Timestamp: 2017-12-21 12:25:57
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:25:57
  Event Severity: CRITICAL
  Event Sensor Num: 10 (a hex)
  Event Sensor Type: Voltage
  Event Sensor Category: Threshold events
  Event Sensor Assertion: FALSE
  Event Sensor State: Unspecified
  Optional Data: None

#9994 (next: #9995)
EntryId: 9994
Timestamp: 2017-12-21 12:26:00
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:00
  Event Severity: MAJOR
  Event Sensor Num: 10 (a hex)
```

```
  Event Sensor Type: Voltage
  Event Sensor Category: Threshold events
  Event Sensor Assertion: FALSE
  Event Sensor State: Unspecified
  Optional Data: None

#9995 (next: #9996)
EntryId: 9995
Timestamp: 2017-12-21 12:26:00
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:00
  Event Severity: MAJOR
  Event Sensor Num: 10 (a hex)
  Event Sensor Type: Voltage
  Event Sensor Category: Threshold events
  Event Sensor Assertion: TRUE
  Event Sensor State: IDLE
  Optional Data: None

#9996 (next: #9997)
EntryId: 9996
Timestamp: 2017-12-21 12:26:00
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:00
  Event Severity: CRITICAL
  Event Sensor Num: 10 (a hex)
  Event Sensor Type: Voltage
  Event Sensor Category: Threshold events
  Event Sensor Assertion: TRUE
  Event Sensor State: FAILURE DEASSERTED
  Optional Data: None

#9997 (next: #9998)
EntryId: 9997
Timestamp: 2017-12-21 12:26:00
  Event Type: SENSOR
```

```
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:00
  Event Severity: MAJOR
  Event Sensor Num: 11 (b hex)
  Event Sensor Type: Current
  Event Sensor Category: Threshold events
  Event Sensor Assertion: FALSE
  Event Sensor State: Unspecified
  Optional Data: None

#9998 (next: #9999)
EntryId: 9998
Timestamp: 2017-12-21 12:26:00
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:00
  Event Severity: CRITICAL
  Event Sensor Num: 11 (b hex)
  Event Sensor Type: Current
  Event Sensor Category: Threshold events
  Event Sensor Assertion: FALSE
  Event Sensor State: Unspecified
  Optional Data: None

#9999 (next: #-2)
EntryId: 9999
Timestamp: 2017-12-21 12:26:03
  Event Type: SENSOR
  Event Resource ID: 4000
  Event Timestamp: 2017-12-21 12:26:03
  Event Severity: MAJOR
  Event Sensor Num: 17 (11 hex)
  Event Sensor Type: Other Units-based Sensor
  Event Sensor Category: Threshold events
  Event Sensor Assertion: FALSE
  Event Sensor State: IDLE
  Optional Data: None
```

8.3 Clear the system event log

8.3.1 Syntax

sel clear

8.3.2 Purpose

This command clears the SEL.

8.3.3 Example

```
CLI{admin}> sel clear
```

9 Alarm Table commands

The following commands deal with the alarm table:

9.1 Show the list of alarms in the alarm table

9.1.1 Syntax

alarm

alarm list

9.1.2 Purpose

This command lists the contents of the alarm table, one alarm per line. The information about each alarm includes the alarm ID, timestamp, severity, whether it is acknowledged, alarm type and originating resource ID. For alarms caused by sensor events, the sensor number and event state is also shown.

9.1.3 Example

```
CLI{admin}> alarm list
(4): 2017-05-11 21:48:08; Severity: CRITICAL; Ack: No; Type: SENSOR; Resource:
4002; Sensor: 210; Event State: 2
```

9.2 Show a specific alarm in the alarm table

9.2.1 Syntax

alarm [show|info] <id>

9.2.2 Purpose

This command shows information about the specific alarm. The information includes the following:

- alarm ID
- timestamp
- alarm severity
- the 'acknowledged' flag
- the alarm condition type (sensor, resource, user, OEM)
- the originating resource ID
- the sensor number and event state (for sensor alarms)
- the manufacturer ID (for OEM alarms)
- the alarm data in the text format (for user and OEM alarms)

9.2.3 Example

```
CLI{admin}> alarm show 4
AlarmId: 4
Timestamp: 2017-05-11 21:48:08
```

```
Alarm Severity: CRITICAL
Acknowledged: No
Alarm Condition:
    Type: SENSOR
    Entity: {0xd9,0}{0xde,1}{0xdd,1}{0xdc,1}{0xdb,1}
    Resource ID: 4002
    Sensor: 210
    Event State: 2
```

9.3 Acknowledge an alarm

9.3.1 Syntax

alarm (ack|acknowledge) <id>

9.3.2 Purpose

This command acknowledges the specified alarm in the alarm table. This operation sets to *TRUE* the flag 'acknowledged' in the corresponding data structure, thus indicating that the user has recognized the presence of this specific alarm.

9.3.3 Example

```
CLI{admin}> alarm ack 4
```

9.4 Delete an alarm

9.4.1 Syntax

alarm delete <id>

9.4.2 Purpose

This command deletes the specified alarm from the alarm table.

9.4.3 Example

```
CLI{admin}> alarm delete 2
Operation completed successfully
```

10 Device Configuration commands

This set of commands deals with the SMRC / iPDU hardware configuration, primarily stored in the FRU Information in the device EEPROM. All configuration parameters represented here are read-only, except the device name and the system time that can be modified by the user.

10.1 Show device configuration

10.1.1 Syntax

device

10.1.2 Purpose

This command shows the following device configuration information:

- device name (can be changed by the user)
- device model
- manufacturer
- product name
- device hardware and firmware version
- device serial number
- current UTC offset of the system time
- the system time, including the time zone
- Asset Tag
- location of the device

10.1.3 Example

```
CLI{admin}> device
Device Name: GuardianManagementGateway
Device Model: 123456789
Manufacturer: nVent, SCHROFF
Product Name: Guardian Management Gateway
Hardware version:
Firmware version: 1.0.4 63998-20552-04 AWS Build date/time: Mar 14 2020 03:18:40
Serial Number: 0000000000AB
Current Time: 2020-03-29 19:43:27
Time Zone: CEST (Europe/Paris)
UTC Offset: +0200
Asset Tag: 0987654321
```

10.2 Show or set device name

10.2.1 Syntax

device name [<name>]

10.2.2 Purpose

This command shows or sets (if the *<name>* parameter is specified) the device name.

10.2.3 Example

```
CLI{admin}> device name SmartProduct
CLI{admin}> device name
Device Name: SmartProduct
```

10.3 Show device model

10.3.1 Syntax

device model

10.3.2 Purpose

This command shows the device model.

10.3.3 Example

```
CLI{admin}> device model
Device Model: AAAAA-BBBB
```

10.4 Show device version

10.4.1 Syntax

device version

10.4.2 Purpose

This command shows the device hardware and firmware versions and the firmware image version.

10.4.3 Example

```
CLI{admin}> device version
Hardware version: 0.1
Firmware version: 1.0.2 63998-20552 AWS Build date/time: Feb 2 2020 14:00:36
Firmware image version: kernel version: 0.20; rootfs version: 0.97; U-Boot
version: 0.17
```

10.5 Show device serial number

10.5.1 Syntax

device serial

10.5.2 Purpose

This command shows the device serial number.

10.5.3 Example

```
CLI{admin}> device serial
Serial Number: 0000000000AB
```

10.6 Show manufacturer

10.6.1 Syntax

device manufacturer

10.6.2 Purpose

This command shows the manufacturer of the device (the field MANUFACTURER of the PRODUCT INFO area)..

10.6.3 Example

```
CLI{admin}> device manufacturer
Manufacturer: nVent, SCHROFF
```

10.7 Show product name

10.7.1 Syntax

device product

10.7.2 Purpose

This command shows the product name of the device (the field PRODUCT NAME of the PRODUCT INFO area).

10.7.3 Example

```
CLI{admin}> device product
Guardian Management Gateway
```

10.8 Show or set System Date, Time and Time Zone

10.8.1 Syntax

device date [<date/time> [<timezone>]]

10.8.2 Purpose

This command shows or sets (if the *<date/time>* and *<timezone>* parameters are specified) the device system date, time and time zone.

The parameter *<date>* is in *yyyy-MM-dd HH:mm:ss* format. The parameter *<timezone>* is an abbreviation-based identifier. A timezone can be set only if the correspondent file is present in */usr/share/zoneinfo* directory.

10.8.3 Example

```
CLI{admin}> device date
```

```
Current Time: 2019-03-22 18:44:07
Time Zone: UTC (Universal)
CLI{admin}> device date 2020-02-10 18:13:02 EST
```

10.9 Show UTC offset

10.9.1 Syntax

device utcoffset

10.9.2 Purpose

This command shows the current UTC offset for the system time.

10.9.3 Example

```
CLI{admin}> device utcoffset
UTC Offset: +0300
```

10.10 Show or set asset tag

10.10.1 Syntax

device asset_tag [<asset tag>]

10.10.2 Purpose

This command shows or sets (if the *<asset tag>* parameter is specified) the device Asset Tag.

10.10.3 Example

```
CLI{admin}> device asset_tag "Device 95"
CLI{admin}> device asset_tag
Asset Tag: Device 95
```

10.11 Show or set device location

10.11.1 Syntax

device location [<location>]

10.11.2 Purpose

This command shows or sets (if the *<location>* parameter is specified) the device location.

10.11.3 Example

```
CLI{admin}> device location "Room 103"
CLI{admin}> device location
Location: Room 103
```

11 Network Configuration commands

The following commands are used for network configuration on the low level:

11.1 Show network configuration

11.1.1 Syntax

netconf info

netconf

11.1.2 Purpose

This command shows the network configuration information. It includes the following:

- for each available network interface, the interface number, the adapter name, the MAC address, the IPv4 address and the IPV6 addresses
- IPv4 DNS servers
- IPv6 DNS servers
- whether IPv4 DNS servers are preferred to IPv6 DNS servers with respect to DNS name resolution

11.1.3 Example

```
CLI{admin}> netconf info
Host name: imx6sxpdu94
Default domain: imx6sxpdu94
Domain search path: imx6sxpdu94
DNS IPv4: 80.240.102.105,8.8.8.8
DNS IPv6:
DNS preference: IPv4
1: "lo" (MAC:00:00:00:00:00:00, IPv6:::1/128, IPv4:127.0.0.1/8)
2: "eth0" (MAC:3C:FB:96:70:81:BA, IPv6:fe80::3efb:96ff:fe70:81ba/64,
IPv4:192.168.1.99/24)
3: "eth1" (MAC:D6:00:B3:BA:4B:F3, IPv6:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00,
IPv4:0.0.0.0/0)
4: "sit0" (MAC:00:00:00:00:00:00, IPv6:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00,
IPv4:0.0.0.0/0)
```

11.2 List network interfaces

11.2.1 Syntax

netconf list

netconf ifaces

11.2.2 Purpose

This command lists known network interfaces. For each network interface, the interface number, the adapter name, the MAC address, the IPv4 address and the IPV6 addresses are shown.

11.2.3 Example

```
CLI{admin}> netconf list

1: "lo" (MAC:00:00:00:00:00:00, IPv6:::1/128, IPv4:127.0.0.1/8)

2: "eth0" (MAC:3C:FB:96:70:81:BA, IPv6:fe80::3efb:96ff:fe70:81ba/64,
IPv4:192.168.1.99/24)

3: "eth1" (MAC:D6:00:B3:BA:4B:F3, IPv6:00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00,
IPv4:0.0.0.0/0)

4: "sit0" (MAC:00:00:00:00:00:00, IPv6:00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00,
IPv4:0.0.0.0/0)
```

11.3 Show hostname

11.3.1 Syntax

netconf hostname

11.3.2 Purpose

This command shows the hostname and the fully qualified domain name of the system.

11.3.3 Example

```
CLI{admin}> netconf hostname

Host name: imx6sxpdu93

Fully Qualified Domain Name: imx6sxpdu93.ppstest
```

11.4 Set hostname

11.4.1 Syntax

netconf hostname <host name>

11.4.2 Purpose

This command sets the hostname of the system

11.4.3 Example

```
CLI{admin}> netconf hostname  imx6sxpdu94

Operation completed successfully
```

11.5 Show DNS domain search path

11.5.1 Syntax

netconf search

netconf search_path

11.5.2 Purpose

This command shows the DNS domain search path of the system.

11.5.3 Example

```
CLI{admin}> netconf search
Domain search path: dns-device.com
```

11.6 Set DNS domain search path

11.6.1 Syntax

netconf search <path>

netconf search_path <path>

11.6.2 Purpose

This command sets the DNS domain search path of the system.

11.6.3 Example

```
CLI{admin}>netconf search dns-device.com,power-device.com
Operation completed successfully
```

11.7 Show DNS information

11.7.1 Syntax

netconf dns [<servers>]

11.7.2 Purpose

This command shows addresses of IPv4 and IPv6 DNS servers for the system. When the *<servers>* parameter is present, the command assigns the IPv4 DNS server addresses.

11.7.3 Example

```
CLI{admin}> netconf dns
Host name: imx6sxpdu93
Default domain: ppstest
Domain search path: ppstest
DNS IPv4: 192.168.1.95
DNS IPv6:
DNS preference: IPv4
CLI{admin}> netconf dns  192.168.1.253
Operation completed successfully
```

11.8 Show and set IPv4 DNS servers

11.8.1 Syntax

netconf dns4 [<servers>]

11.8.2 Purpose

This command shows, or assigns, the IPv4 DNS server addresses. The parameter *<servers>* should comprise comma-separated DNS server names or addresses.

11.8.3 Example

```
CLI{admin}> netconf dns4 192.168.1.253
Operation completed successfully
CLI{admin}> netconf dns4
DNS IPv4: 192.168.1.253
```

11.9 Show and set IPv6 DNS servers

11.9.1 Syntax

netconf dns6 [<servers>]

11.9.2 Purpose

This command shows, or assigns, the IPv6 DNS server addresses. The parameter *<servers>* should comprise comma-separated DNS server names or addresses.

11.9.3 Example

```
CLI{admin}> netconf dns6 fe80::218:49ff:fe01:8f78,fe80::218:49ff:fe01:8f77
Operation completed successfully
CLI{admin}> netconf dns6
DNS IPv6:       fe80::218:49ff:fe01:8f78,fe80::218:49ff:fe01:8f77
DNS preference: IPv6
CLI{admin}> netconf dns6
DNS IPv6:
DNS preference: IPv4
```

11.10 Set DNS IPv4/IPv6 preference

11.10.1 Syntax

netconf dns6pref <value>

```
<value> ::= true | false
```

11.10.2 Purpose

This command sets the flag that indicates whether IPv4 DNS servers are preferred to IPv6 DNS servers with respect to DNS name resolution. The value *true* indicates IPv6, the value *false* indicates IPv4.

11.10.3 Example

```
CLI{admin}> netconf dns6pref false
Operation completed successfully
CLI{admin}> netconf dns6pref
DNS preference: IPv4
```

11.11 Show network interface configuration

11.11.1 Syntax

netconf show <interface number>

11.11.2 Purpose

This command shows the detailed information about the specified network adapter. This information includes the following:

- interface number
- adapter name
- interface mode and speed
- adapter MAC address
- IPv4 address and netmask
- IPv4 default gateway address
- IPV6 addresses with prefixes assigned to the adapter
- IPv6 routing information

11.11.3 Example

```
CLI{admin}> netconf show 1
lo:
MAC Address:          00:00:00:00:00:00
IPv6 Address & Prefix: ::1/128
Routes:
1: ::1/128: ::
2: fe80::3efb:96ff:fe70:81ba/128: ::
3: ::/0: ::
DHCP:                 static
IP Address & Mask:    127.0.0.1/8
Gateway:              0.0.0.0
DNS:                  192.168.1.253
CLI{admin}> netconf show 2
eth0:
```

```
MAC Address:          3C:FB:96:70:81:BA
IPv6 Address & Prefix: fe80::3efb:96ff:fe70:81ba/64
Routes:
1: fe80::/64: ::
2: ff00::/8: ::
DHCP:                 static
IP Address & Mask:    192.168.1.99/24
Gateway:              192.168.1.253
DNS:                  127.0.0.1
Auto Negotiate:       True
Duplex:               1
Speed:                100
```

11.12 Show specific network interface configuration parameters

11.12.1 Syntax

netconf mac <interface number>

netconf route <interface number>

netconf mode <interface number>

netconf speed <interface number>

netconf autoneg <interface number>

netconf duplex <interface number>]

netconf ip <interface number>

netconf gw <interface number>

netconf ip6 <interface number>

netconf gw6 <interface number>

11.12.2 Purpose

This set of commands shows separately specific configuration items for the specified network adapter, as follows:

- `mac` - the MAC address
- `route` - the IPv6 routing information
- `mode` - interface mode and speed
- `speed` - interface speed
- `autoneg` - auto-negotiation mode
- `duplex` - duplex mode
- `ip` - IPv4 address and netmask

- `gw` - IPV4 gateway
- `ip6` - IPv6 addresses with prefixes
- `gw6` - IPv6 gateway address

11.12.3 Example

```
CLI{admin}> netconf mac 2
MAC Address:          3C:FB:96:70:81:BA
CLI{admin}> netconf mode 2
Auto Negotiate: False
Duplex:         0
Speed:          100
CLI{admin}> netconf route 2
Routes:
1: fe80::/64: ::
2: ff00::/8: ::
CLI{admin}> netconf  mode 2 120 1 true
Auto Negotiate: True
Duplex:         1
Speed:          120
CLI{admin}> netconf  ip 2
DHCP:               static
IP Address & Mask: 192.168.1.99/24
Gateway:            192.168.1.253
CLI{admin}> netconf  ip6 2
Addressing: Static
1: 4123:db8::3efb:96ff:fe77:88a1/64
2: fe80::3efb:96ff:fe77:88ad/64
CLI{admin}> netconf gw6 2
Default Gateway: 4123:db8::3efb:96ff:fe77:88a0
```

11.13 Set network interface configuration

11.13.1 Syntax

netconf speed <interface number> <speed> [<duplex> [<auto-neg>]]

netconf autoneg <interface number> <auto-neg>

netconf duplex <interface number> <duplex>

netconf ip <interface number> <ipv4 address param>

netconf gw <interface number> <ipv4 gateway>

netconf ip6 <interface number> <ipv6 addresses param>

netconf gw6 <interface number> <ipv6 gateway>

```
<speed> ::= <number>

<auto-neg> ::= true | false

<duplex> ::= 0 | 1

<ipv4 address param> ::= (auto | <ip address>/<number>) [<ipv4 gateway>]

<ipv4 gateway> ::= <ip address>

<ipv6 addresses param> ::= (auto | dhcp| <ipv6 address>/<number>[,<ipv6
address>/<number>]...) [<ipv6 gateway>]

<ipv6 gateway> ::= <ipv6 address>
```

11.13.2 Purpose

This command sets various configuration attributes of the specified network adapter. Multiple configuration attributes can be set with a single command. Syntactically the command may include several clauses each of which is responsible for setting one interface parameter. The order of the parameters doesn't matter.

The clauses correspond to network interface configuration parameters as follows:

- `speed` - interface speed
- `autoneg` - auto-negotiation mode
- `duplex` - duplex mode
- `ip` - IPv4 address and netmask
- `gw` - IPV4 gateway
- `ip6` - IPv6 addresses with prefixes
- `gw6` - IPv6 gateway address

If the IPv4 address is specified as `auto`, the DHCPv4 client functionality is turned on and the IPv4 address is received via DHCP. If the IPv6 address is specified as `auto`, the auto-configuration functionality is turned on and the IPv6 address is configured without the need for a server. If the IPv6 address is specified as `dhcp`, the DHCPv6 client functionality is turned on and the IPv6 address is received via DHCP. The parameter *<duplex>* has the value *0* for half-duplex mode and *1* for full-duplex mode.

11.13.3 Example

```
CLI{admin}> netconf  mode 2 120 1 true
Auto Negotiate: True
Duplex:         1
Speed:          120
CLI{admin}> netconf ip 2 192.168.1.94 192.168.1.253
DHCP:               static
IP Address & Mask: 192.168.1.94
Gateway:           192.168.1.253
```

```
CLI{admin}> netconf ip 2 auto
DHCP:               auto
CLI{admin}> netconf ip 2 192.168.1.94/24 192.168.1.253
DHCP:               static
IP Address & Mask: 192.168.1.94/24
Gateway:            192.168.1.253
CLI{admin}> netconf gw6 2
Default Gateway: fe80::7271:bcff:fe9a:851
CLI{admin}> netconf ip6 2
Addressing: Static
1: fe80::3efb:96ff:fe77:88ad/64
2: 4222:db8::3efb:96ff:fe77:88ac/64
CLI{admin}> netconf ip6 2  4111:db8::3efb:96ff:fe77:88a1/64
4111:db8::3efb:96ff:fe77:88a0
CLI{admin}> netconf ip6 2 auto
```

11.14 Show or change rejected DHCP v4 servers

11.14.1 Syntax

netconf dhcp_reject [<ip-address-list> | clear]

11.14.2 Purpose

This command shows or changes the list of rejected DHCP v4 servers, depending on the presence of additional arguments. If there is no optional argument, this command shows the list of rejected DHCP v4 servers. If the optional argument <*ip-address-list*> is present, this argument specifies the list of rejected DHCP v4 servers to be set. Components of the list are separated with commas (“,”). If the optional argument *clear* is present, this command clears the list of rejected DHCP v4 servers.

11.14.3 Example

```
CLI{admin}> netconf dhcp_reject 172.16.0.0/16,192.168.1.0/12
CLI{admin}> netconf dhcp_reject
Rejected DHCP Servers: 172.16.0.0/16,192.168.1.0/12
CLI{admin}> netconf dhcp_reject clear
```

11.15 Show or change rejected DHCP v6 servers

11.15.1 Syntax

netconf dhcpv6_reject [<ipv6-address-list> | clear]

11.15.2 Purpose

This command shows or changes the list of rejected DHCP v6 servers, depending on the presence of additional arguments. If there is no optional argument, this command shows the list of rejected DHCP v6 servers. If the optional argument <*ipv6-address-list*> is present, this argument specifies the list of rejected DHCP v6 servers to

be set. Components of the list are separated with commas (“,”). If the optional argument *clear* is present, this command clears the list of rejected DHCP v6 servers.

11.15.3 Example

```
CLI{admin}> netconf dhcpv6_reject fe80::7271:bcff:fe9a:851/64
CLI{admin}> netconf dhcpv6_reject
Rejected DHCPv6 Servers: fe80::7271:bcff:fe9a:851/64
CLI{admin}> netconf dhcpv6_reject clear
```

12 Network Service Configuration commands

The following commands are used for network services configuration:

12.1 Show services configuration

12.1.1 Syntax

srvconf info

srvconf

12.1.2 Purpose

This command shows the network services configuration information. It includes the following:

- for HTTP and HTTPS, the port numbers and whether usage of HTTPS is enforced
- for Telnet, whether it is enabled and if enabled, the port number
- for SSH, whether it is enabled and if enabled, the port number and authorization method
- for SMTP, own e-mail address, server address and the list of recipients by default
- for SNMP, enable flags, read/write community, sysName, sysContact, sysLocation attributes, the default trap destination and what protocol should be used for traps
- for NTP, whether it is enabled, and if enabled, the flag that indicates whether the server addresses come from DHCP, and primary and secondary server addresses

12.1.3 Example

```
CLI{admin}> srvconf info
HTTP port: 80
HTTPS port: 443
Enforce HTTPS: True
Telnet: enabled
   Telnet Port: 23
SSH: enabled
   SSH Port: 22
   Authorization Method: Password Or Public Key
SMTP:
   Own Address: smartrack@smartrack
   Server: smtp.test
   Default Recipients:
SNMP:
   state: enabled
   V1/V2: enabled
   Read Community: public
```

```
   Write Community: private
   Sys Name: Sample system name
   Sys Contact: syscontact@nVent.com
   Sys Location: New facility
   Trap Destination: 192.168.1.16
   V2 Traps: enabled
NTP: enabled
   NTP From DHCP: disabled
   Primary Server: 192.168.1.3
   Secondary Server:192.168.1.4
```

12.2 Show or change HTTP information

12.2.1 Syntax

srvconf http

srvconf http <port>

<port> ::= <number>

12.2.2 Purpose

This command shows or changes HTTP port, depending on the presence of the port argument.

12.2.3 Example

```
CLI{admin}> srvconf http
HTTP port: 80
CLI{admin}> srvconf http 8080
CLI{admin}> srvconf http
HTTP port: 8080
```

12.3 Show or change HTTPS information

12.3.1 Syntax

srvconf https

srvconf https <port>

<port> ::= <number>

12.3.2 Purpose

This command shows or changes HTTPS port, depending on the presence of the port argument.

12.3.3 Example

```
CLI{admin}> srvconf https
HTTPS port: 443
CLI{admin}> srvconf https 993
```

```
CLI{admin}> srvconf https
HTTPS port: 993
```

12.4 Encrypted HTTP protocol enforcement command

12.4.1 Syntax

srvconf enforce_https [<status>]

```
<status> ::= true | false
```

12.4.2 Purpose

When the optional parameter *<status>* is omitted this command reports whether the HTTPS protocol is enforced. When the optional parameter *<status>* is present this command sets the status of enforcement of HTTPS protocol.

12.4.3 Examples

```
CLI{admin}> srvconf enforce_https
Enforce HTTPS: False
CLI{admin}> srvconf enforce_https true
Operation completed successfully
```

12.5 Show or change Telnet information

12.5.1 Syntax

srvconf telnet

srvconf telnet <enable> [<port>]

```
<enable> ::= true | false
<port> ::= <number>
```

12.5.2 Purpose

This command shows or changes Telnet service information, depending on the presence of additional arguments. Telnet service can be enabled or disabled, Telnet port can be changed. If Telnet service is disabled, the argument for the port is ignored and can be omitted.

12.5.3 Example

```
CLI{admin}> srvconf telnet
Telnet: enabled
   Telnet Port: 23
CLI{admin}> srvconf telnet false
CLI{admin}> srvconf telnet
Telnet: disabled
CLI{admin}> srvconf telnet true 24
CLI{admin}> srvconf telnet
Telnet: enabled
```

```
   Telnet Port: 24
```

12.6 Show or change SSH information

12.6.1 Syntax

srvconf ssh

srvconf ssh <enable> [<port> <auth-method>]

```
<enable> ::= true | false
<port> ::= <number>
<auth-method> ::= password | pubkey | password_pubkey
```

12.6.2 Purpose

This command shows or changes SSH service information, depending on the presence of additional arguments. SSH can be enabled or disabled, SSH port and authorization method can be changed. If SSH service is disabled, arguments for the port and authorization method are ignored and can be omitted.

12.6.3 Example

```
CLI{admin}> srvconf ssh
SSH: enabled
   SSH Port: 22
   Authorization Method: Password Or Public Key
CLI{admin}> srvconf ssh false
CLI{admin}> srvconf ssh
SSH: disabled
CLI{admin}> srvconf ssh true 22 password
CLI{admin}> srvconf ssh
SSH: enabled
   SSH Port: 22
   Authorization Method: Password Only
```

12.7 Show or change SMTP information

12.7.1 Syntax

srvconf smtp

srvconf smtp address <address>

srvconf smtp server <server>

srvconf smtp recipients <default-recipients>

```
<address> ::= <string>
<server> ::= <string>
<default-recipients> ::= <string> | <default-recipients> , <string>
```

12.7.2 Purpose

This command shows or changes SMTP service information, depending on the presence of additional arguments. Own e-mail address, SMTP server name/address and the list of default recipients can be changed.

12.7.3 Example

```
CLI{admin}> srvconf smtp
SMTP:
   Own Address: smartrack@smartrack
   Server: smtp.test.com
   Default Recipients:
CLI{admin}> srvconf smtp address smartrack@somedomain.com
CLI{admin}> srvconf smtp server smtp.somedomain.com
CLI{admin}> srvconf smtp recipients
report@anotherdomain.com,smrc@anotherdomain.com
CLI{admin}> srvconf smtp
SMTP:
   Own Address: smartrack@somedomain.com
   Server: smtp.somedomain.com
   Default Recipients: report@anotherdomain.com,smrc@anotherdomain.com
```

12.8 Show or change SNMP information

12.8.1 Syntax

srvconf snmp

srvconf snmp enable <enable> <enable-v1v2>

srvconf snmp readcommunity <string>

srvconf snmp writecommunity <string>

srvconf snmp sysname <string>

srvconf snmp syscontact <string>

srvconf snmp syslocation <string>

srvconf snmp trapdestination <string>

srvconf snmp v2traps <enable-v2traps>

```
<enable> ::= true | false
<enable-v1v2> ::= true | false
<enable-v2traps> ::= true | false
```

12.8.2 Purpose

This command shows or changes SNMP service information, depending on the presence of additional arguments. If called without arguments, the command shows current SNMP settings. Otherwise, the first additional argument is a keyword that indicates which parameter(s) should be changed.

The following SNMP service parameters can be changed with this command.

- Service enables: separately for SNMP V3 service and V1/V2 versions of the protocol
- Read-only community (“public” by default)
- Read-write community (“private” by default)
- System name (an arbitrary string)
- System contact (name and e-mail address of the contact person)
- System location (an arbitrary string)
- Default trap destination (a host name or an IP address)
- Use V2 format for traps (a logical flag)

The command, when called with additional arguments, restarts the SNMP server, so that parameter changes have immediate effect.

12.8.3 Example

```
CLI{admin}> srvconf snmp
SNMP:
   state: enabled
   V1/V2: enabled
   Read Community: public
   Write Community: private
   Sys Name: Sample system name
   Sys Contact: syscontact@nVent.com
   Sys Location: New facility
   Trap Destination: 192.168.1.16
   V2 Traps: enabled
```

The following command enables only the SNMP V3 protocol:

```
CLI{admin}> srvconf snmp enable true false
```

The following command disables the SNMP V3 service:

```
CLI{admin}> srvconf snmp enable false true
CLI{admin}> srvconf snmp
SNMP:
   state: disabled
   V1/V2: enabled
```

Configure other SNMP parameters:

```
CLI{admin}> srvconf snmp readcommunity MyReadCommunity
CLI{admin}> srvconf snmp writecommunity MyWriteCommunity
CLI{admin}> srvconf snmp sysname " Guardian Management Gateway "
CLI{admin}> srvconf snmp syscontact "contact@somedomain.com"
```

```
CLI{admin}> srvconf snmp syslocation "Manufacturing facility"
CLI{admin}> srvconf snmp trapdestination 192.168.1.124
CLI{admin}> srvconf snmp v2traps false
```

Now read the SNMP parameters back:

```
CLI{admin}> srvconf snmp
SNMP:
   state: enabled
   V1/V2: disabled
   Read Community: MyReadCommunity
   Write Community: MyWriteCommunity
   Sys Name: Guardian Management Gateway
   Sys Contact: contact@somedomain.com
   Sys Location: Manufacturing facility
   Trap Destination: 192.168.1.124
   V2 Traps: disabled
```

12.9 Show or change NTP information

12.9.1 Syntax

srvconf ntp

srvconf ntp enable <enable>

srvconf ntp fromdhcp <enable-from-dhcp>

srvconf ntp servers <override> <primary> [<secondary>]

```
<enable> ::= true | false
<enable-from-dhcp> ::= true | false
<override> ::= true | false
<primary> ::= <string>
<secondary> ::= <string>
```

12.9.2 Purpose

This command shows or changes NTP service information, depending on the presence of additional arguments. If called without arguments, the command shows current NTP settings. Otherwise, the first additional argument is a keyword that indicates which parameter(s) should be changed.

The following NTP service parameters can be changed with this command.

- Service enable
- Receive NTP server addresses from DHCP (a logical flag)
- Primary and secondary NTP servers (host names or IP addresses), plus a logical flag that indicates whether the specified addresses override previously made assignments (likely obtained from DHCP).

The command, when called with additional arguments, restarts the NTP server, so that parameter changes have immediate effect.

12.9.3 Example

```
CLI{admin}> srvconf ntp
NTP: enabled
   NTP From DHCP: enabled
   Primary Server: 192.168.1.1
   Secondary Server:
```

The following command disables NTP service:

```
CLI{admin}> srvconf ntp enable false
CLI{admin}> srvconf ntp
NTP: disabled
```

Enable the NTP service and configure other NTP parameters:

```
CLI{admin}> srvconf ntp enable false
CLI{admin}> srvconf ntp enable true
CLI{admin}> srvconf ntp fromdhcp false
CLI{admin}> srvconf ntp servers true 192.168.1.3 192.168.1.4
```

Now read the NTP parameters back:

```
CLI{admin}> srvconf ntp
NTP: enabled
   NTP From DHCP: disabled
   Primary Server: 192.168.1.3
   Secondary Server: 192.168.1.4
```

13 Global configuration commands

13.1 Show global configuration information

13.1.1 Syntax

global [info]

13.1.2 Purpose

This command shows the global configuration parameters.

13.1.3 Example

```
CLI{admin}> global
Global parameters:
   Measurement Units:
      Temperature: Celsius
      Length: Meters
      Pressure: Pascals
   Web Parameters:
      Idle Detection: Enabled
      Idle Timeout: 600 sec
      Delay Before Disconnect: 15 sec
      Event Log Poll Period: 5 sec
   LCD UI Flags: 0x0
   Extended Sensor Z-Coordinate: 1 (Text)
   Transient Alarm Severity: CRITICAL
```

13.2 Set or show Transient Alarm Severity Level

13.2.1 Syntax

global transient_severity [<severity>]

```
<severity>::= critical | major | minor | info | ok
```

13.2.2 Purpose

This command shows the Transient Alarm Severity Level. Only alarms with more severe level than the Transient Alarm Severity Level severity are stored persistently. When the optional parameter *<severity>* is present this command sets the Transient Alarm Severity Level.

13.2.3 Example

```
CLI{admin}> global transient_severity
   Transient Alarm Severity: CRITICAL
CLI{admin}> global transient_severity major
```

13.3 Extended Sensor Z-Coordinate Unit

13.3.1 Syntax

global ext_zcoord [<zcoord>]

13.3.2 Purpose

This command shows the Extended Sensor Z-Coordinate Unit. When the optional numerical parameter *<zcoord>* is present this command sets the Extended Sensor Z-Coordinate Unit. The values *0* and *1* of the parameter *<zcoord>* correspond to 'rack units' and 'text', respectively. Other values are reserved.

13.3.3 Example

```
CLI{admin}> global ext_zcoord
   Extended Sensor Z-Coordinate: 0 (Rack Units)
CLI{admin}> global ext_zcoord 1
CLI{admin}> global ext_zcoord
   Extended Sensor Z-Coordinate: 1 (Text)
```

13.4 Set or show LCD UI flags

13.4.1 Syntax

global lcd_ui [<flags>]

13.4.2 Purpose

This command shows the LCD UI flags. When the optional decimal parameter *<flags>* is present this command sets the LCD UI flags.

13.4.3 Example

```
CLI{admin}> global lcd_ui 0x10
CLI{admin}> global lcd_ui
   LCD UI Flags: 0x10
```

13.5 Set or show global measurement units

13.5.1 Syntax

global measure [(Fahrenheit|Celsius) (Feet|Meters) (PSI|Pascals)]

13.5.2 Purpose

This command shows the global Measurement Units. When the optional parameters are present this command sets the Measurement Units.

13.5.3 Example

```
CLI{admin}> global measure
   Measurement Units:
      Temperature: Celsius
```

```
      Length: Meters
      Pressure: Pascals
CLI{admin}>  global measure Fahrenheit Feet PSI
```

13.6 Set or show global Web parameters

13.6.1 Syntax

global web [enable|disable <idle-timeout> <delay> <event-log-poll-period>]

13.6.2 Purpose

This command reports the global Web Session Parameters. If additional parameters are provided, this command set the global Web Session Parameters. The additional parameter defines whether the idle detection of web sessions is enabled. The integer parameters *<idle-timeout>*, *<delay>* and *<period>* correspond to the following Web Session Parameters: Idle Timeout, Log Query Polling Period, Delay Before Disconnect (in seconds), respectively.

13.6.3 Example

```
CLI{admin}> global web
   Web Parameters:
      Idle Detection: Enabled
      Idle Timeout: 600 sec
      Delay Before Disconnect: 15 sec
      Event Log Poll Period: 5 sec
CLI{admin}> global web enable 660 16 5
```

14 Server Reachability Table commands

This set of commands works with the server reachability table. For the servers in this table, the SMRC software periodically verifies their accessibility by pinging them over the network. The accessibility status is stored in the table and can be shown on request.

14.1 Show reachability table entries

14.1.1 Syntax

reachability list

reachability

14.1.2 Purpose

This command shows all entries of the reachability table. For each entry, the following information is shown:

- server address or name
- whether pinging is enabled for the server
- if pinging is enabled, the reachability status (`reachable`, `unreachable` or `waiting`)
- after how many successful ping attempts the server is considered reachable
- after how many unsuccessful ping attempts the server is considered unreachable
- interval to the next ping if current ping was successful
- interval to the next ping if current ping was unsuccessful
- interval between the moment the reachability decision is made and the moment when pinging resumes

14.1.3 Example

```
CLI{admin}> reachability list
Count of destinations: 4
1:"192.168.1.253":3:5:20:30:60:Enabled/Waiting
2:"192.168.1.93":3:4:20:30:60:Enabled/Reachable
3:"192.168.1.149":3:4:30:50:120:Disabled
4:"192.168.1.102":3:4:20:40:80:Enabled/Unreachable
```

14.2 Add a reachability table entry

14.2.1 Syntax

reachability add <server> [<successful count> [<unsuccessful count> [<seconds after successful> [<seconds after unsuccessful> [<seconds before resume> [<pinging enabled>]]]]]]

```
<pinging enabled> ::= enable | disable
```

14.2.2 Purpose

This command adds a new entry into the reachability table. All parameters except the server name (or address) are optional and can be omitted, the default values are substituted in that case. Pinging is enabled by default.

14.2.3 Example

```
CLI{admin}> reachability add 192.168.1.253 3 5 20 30 60 enable
Added 192.168.1.253 as destination 1
```

14.3 Update a reachability table entry

14.3.1 Syntax

reachability update <entry> <server> [<successful count> [<unsuccessful count> [<seconds after successful> [<seconds after unsuccessful> [<seconds before resume> [<pinging enabled>]]]]]]

reachability set <entry> <server> [<successful count> [<unsuccessful count> [<seconds after successful> [<seconds after unsuccessful> [<seconds before resume> [<pinging enabled>]]]]]]

<pinging enabled> ::= enable | disable

14.3.2 Purpose

This command modifies the existing entry in the reachability table with the specified entry number. All parameters except the server name (or address) are optional and can be omitted, the default values are substituted in that case. Pinging is enabled by default. The whole entry is replaced, previous content of the fields is not preserved even if the corresponding parameters are omitted.

14.3.3 Example

```
CLI{admin}> reachability update 1 192.168.1.102
Operation completed successfully
```

14.4 Delete a reachability table entry

14.4.1 Syntax

reachability delete <entry>

reachability remove <entry>

14.4.2 Purpose

This command deletes the existing entry in the reachability table with the specified entry number.

14.4.3 Example

```
CLI{admin}> reachability remove 1
Operation completed successfully
```

14.5 Enable a reachability table entry

14.5.1 Syntax

reachability enable <entry>

14.5.2 Purpose

This command enables the entry in the reachability table.

14.5.3 Example

```
CLI{admin}> reachability enable 2
Operation completed successfully
```

14.6 Disable a reachability table entry

14.6.1 Syntax

reachability disable <entry>

14.6.2 Purpose

This command disables the entry in the reachability table.

14.6.3 Example

```
CLI{admin}> reachability disable 2
Operation completed successfully
```

15 User Management commands

A user name should satisfy the following regular expression:

```
NAME_REGEX="^[a-z][-a-z0-9_]*\$"
```

In other words, a user name must consist of lowercase letters, digits, underscore (‘_’) and hyphen (‘-‘) and must start with a lowercase letter. No uppercase letters or special characters other than ‘_’ and ‘-‘ symbols are allowed. The length of a user name must not exceed 32.

The following commands deal with user management:

15.1 List users

15.1.1 Syntax

user list

user

15.1.2 Purpose

This command lists all known users. For each user, the following information is shown:

- user name
- whether the user is enabled
- full user name
- phone number
- e-mail address

15.1.3 Example

```
CLI{admin}> user
User (0): admin - Enabled
  Full name: Administrator
  Phone:
  e-mail:

User (1): user - Enabled
  Full name: Regular User
  Phone:
  e-mail:

User (2): guest - Enabled
  Full name: Guest User
  Phone:
  e-mail:
```

15.2 Show user information

15.2.1 Syntax

user show <index>

user show <name>

15.2.2 Purpose

This command shows information about the existing user, by its index in the list or by name. The following information about the user is shown:

- user name
- whether the user is enabled
- full user name
- phone number
- e-mail address

15.2.3 Example

```
CLI{admin}> user show 15
User (15): xyz - Enabled
  Full name: Absolutely
  Phone:
  e-mail:     abc@smartrack.com
```

15.3 Create a new user

15.3.1 Syntax

user create <name> <password> [<property>...]

```
<property> ::= fullname <string> | phone <string> | email <string> | roles <role
list> | forced <force password change> | enabled <enable> | external <enable>
<force password change> ::= yes | no
<enable> ::= yes | no
```

15.3.2 Purpose

This command creates a new user and adds it to the user list and to the system user directory. The user name and password must be specified for the new user. In addition, the following properties for the new user can be specified in the command:

- Full user name
- Phone number (an opaque string)
- E-mail address (an opaque string)
- Role list: a comma-separated list of role names
- 'Force password change' flag

- 'User enabled' flag
- 'User external' flag

The default values are empty strings for text attributes, *FALSE* for the 'Force password change' flag, *TRUE* for the 'User enabled' flag, and *FALSE* for the 'User external' flag.

The *<password>* parameter and the 'Force password change' flag are ignored when the 'User external' flag is set. If the *<password>* parameter does not satisfy the login restrictions, the *INVALID_DATA* error is reported.

15.3.3 Example

```
CLI{admin}> user create xyz A1b2C3 fullname "Absolutely" phone (123)456-78-90
User (15) xyz has been successfully created
CLI{admin}> user create ldapuser1 IgnoredPassword fullname "LDAP User 1"
external yes
User (18) ldapuser1 has been successfully created
```

15.4 Delete a user

15.4.1 Syntax

user delete <name> [remove_home]

15.4.2 Purpose

This command deletes the existing user by its name. If the parameter *remove_home* is present the home directory is deleted.

15.4.3 Example

```
CLI{admin}> user delete xyz
Operation completed successfully
CLI{admin}> user delete abc remove_home
Operation completed successfully
```

15.5 Update user properties

15.5.1 Syntax

user set <name> <property>

```
<property> ::= fullname <string> | phone <string> | email <string> |  enabled
<enable>
<enable> ::= yes | no
```

15.5.2 Purpose

This command updates properties for the existing user specified by name. The properties that can be modified by this command include:

- Full user name
- Phone number (an opaque string)

- E-mail address (an opaque string)
- 'User enabled' flag

15.5.3 Example

```
CLI{admin}> user set xyz fullname Temporary
Operation completed successfully
CLI{admin}> user set xyz enabled no
Operation completed successfully
```

15.6 Set user password

15.6.1 Syntax

user password <name> <password> [force <force password change>]

```
<force password change> ::= yes | no
```

15.6.2 Purpose

This command sets the new password for the user specified by the name. The default value for 'force password change' flag is *FALSE*.

15.6.3 Example

```
CLI{admin}> user password xyz X#j3Z! force yes
Operation completed successfully
```

15.7 Show user SNMPv3 attributes

15.7.1 Syntax

user snmp get <name>

15.7.2 Purpose

This command shows SNMPv3 parameters for the user specified by its name.

15.7.3 Example

```
CLI{admin}> user snmp get abc
User abc SNMPv3 attributes:
    SNMP                              : Enabled
    ReadWrite                         : Yes
    AuthProtocol                      : MD5
    PrivacyProtocol                   : DES
```

15.8 Set user SNMPv3 parameters

15.8.1 Syntax

user snmp set <name> <enable> <write> <auth> <privacy> <auth_as_priv> <auth_pass> [<priv_pass>]

<enable> ::= enable | disable

<write> ::= yes | no

<auth> ::= md5 | sha

<privacy> ::= des | aes

<auth_as_priv> :: yes | no

15.8.2 Purpose

This command sets SNMPv3 parameters for the user specified by its name. A new SNMP user is created by this command.

Parameter *<enable>* specifies whether or not SNMPv3 is enabled. Parameter *<write>* defines whether or not the write operation is allowed for SNMPv3. Parameters *<auth>* and *<priv>* define the authentication and privacy protocols, respectively. Parameter *<auth pass>* and *<priv pass>* define the authentication password and the privacy password, respectively. If the parameter *<auth_as_priv>* is set to *no*, then the privacy password is set to the authentication password, and the parameter *<priv pass>* is ignored. The length of the passwords must be at least 8 characters.

15.8.3 Examples

CLI{admin}> **user snmp set hello123 enable yes md5 des no PASSWORD123 PRIVACYPASS**

15.9 Enable SNMPV3 for a user

15.9.1 Syntax

user snmp enable <name>

15.9.2 Purpose

This command enables SNMPv3 for the user specified by its name.

15.9.3 Examples

CLI{admin}> **user snmp enable hello123**

Operation completed successfully

15.10 Disable SNMPV3 for a user

15.10.1 Syntax

user snmp disable <name>

15.10.2 Purpose

This command disables SNMPv3 for the user specified by its name.

15.10.3 Examples

```
CLI{admin}> user snmp disable hello123
Operation completed successfully
```

15.11 Set or show user roles

15.11.1 Syntax

user roles <name> [<role list>]

<role list> ::= <role> | <role list> , <role>

15.11.2 Purpose

This command shows the list of the roles that the specified user possesses, or sets the new role list for the user.

The parameter *<role list>* is the comma-separated list of role names. If this parameter is omitted, the current list of roles for the user is shown.

15.11.3 Example

```
CLI{admin}> user roles guest
User roles:
    UserRole
    ReadOnlyUserRole
CLI{admin}> user roles abc UserRole
Operation completed successfully
CLI{admin}> user roles testuser ReadOnlyUserRole,UserRole
Operation completed successfully
```

15.12 Show user privileges

15.12.1 Syntax

user privileges <name>

15.12.2 Purpose

This command shows the list of all privileges for the existing user, aggregated across all roles that the user possesses.

15.12.3 Example

```
CLI{admin}> user privileges guest
Privileges list for user 'guest':
  Common user
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View SNMP settings
```

```
View user settings
View Webcam settings
View IPDU configuration
Use groups
```

15.13 Set or show current measurement units

15.13.1 Syntax

user measure <name> [<temperature> <length> <pressure>]

user measure <name> [(Fahrenheit|Celsius) (Feet|Meters) (PSI|Pascals)]

```
<temperature> ::= c | f | *
<length> ::= m | e | *
<pressure> ::= psi | pas | *
```

15.13.2 Purpose

This command shows the current measurement units for the existing user, or sets new values. The measurement units include temperature degrees (Celsius, *c*, or Fahrenheit, *f*), length units (meters, *m*, or feet, *f*) and pressure units (PSI, *psi*, or Pascals, *pas*).

If the parameters *<temperature>*, *<length>* and *<pressure>* are omitted, the current measurement units for the user are shown. If the parameters are specified, the new values are set for the user. The value *** means 'keep old unit'.

15.13.3 Example

```
CLI{admin}> user measure guest
Measurement units for guest
Temperature: Fahrenheit
Length: Feet
Pressure: Pascals
CLI{admin}> user measure xyz c m psi
CLI{admin}> user measure xyz Celsius Meters PSI
```

15.14 Manage the user SSH keys

15.14.1 Syntax

user sshkey <name> (<key>| delete)

15.14.2 Purpose

This command manages user SSH keys. If no additional parameter is present the list of SSH key comments for the user is reported (usually, a comment represents a network location from which the user can log in using this key). If the *<key>* parameter is present the command adds a new value of the SSH key. The new key is represented as a text string. If the *delete* parameter is present all the SSH keys are deleted.

15.14.3 Example

```
CLI{admin}> user sshkey xyz
Keys found: 2
Key 1: from john@myhost.com
Key 2: from abc@myoffice.com
CLI{admin}>user sshkey xyz 192.168.1.253 1024 33
12130121787870229809105854022244171222161686331552232579940622752740062970790855
95824582072827869274042306145739387552476555269673967177260290827941846042869525
25999413000674811746858357448545664002699175286780641785658489317135491199379083
70765157281237146056413779035146821365210555528008882841552639993212 9
xyz@myhost.com
Operation completed successfully
CLI{admin}>user sshkey xyz delete
Operation completed successfully
```

15.15 User web session

15.15.1 Syntax

user websession <name>

user websession <name> <idle detection> <timeout> <delay> <period>

<idle detection> ::= enable | disable

15.15.2 Purpose

This command reports the Web Session Parameters for the user specified by its name. If additional parameters are provided, this command set the Web Session Parameters. The integer parameters *<timeout>*, *<delay>* and *<period>* correspond to the following Web Session Parameters: Idle Timeout, Delay Before Disconnect, Log Query Polling Period, respectively.

15.15.3 Example

```
CLI{admin}> user websession another
Idle Detection: Enabled
Idle Timeout: 600 sec
Delay Before Disconnect: 15 sec
Event Log Poll Period: 5 sec
CLI{admin}> user websession another disable 600 20 5
CLI{admin}> user websession another enable 600 20 5
```

15.16 Set or show user language setting

15.16.1 Syntax

user language [<name>] [<lang>]

<lang> ::= english | french | german

15.16.2 Purpose

When the optional parameter *<name>* is omitted this command is applied to the language setting of the current user. When the optional parameter *<name>* is present this command is applied to the language setting of the user specified by its name *<name>*.

When the optional parameter *<lang>* is omitted this command reports the user's language setting. When the optional parameter *<lang>* is present this command sets the user's language setting.

15.16.3 Example

```
CLI{admin}> user language
Current language: English
CLI{admin}> user language another
Language for user another: English
CLI{admin}> user language another german
Language for user another: Deutsch
```

16 Role Management commands

The following commands deal with role management:

16.1 List roles

16.1.1 Syntax

role list

role

16.1.2 Purpose

This command lists all existing roles. For each role, the index of the role in the list, the role name, description and the list of assigned privileges are shown.

16.1.3 Example

```
CLI{admin}> role
Role (0): AdministratorRole
Description:
This is the adiministrator role
Privileges: (0xFFFFFFFFFFFFFFFF 0x0000000000000000)
  Common user
  Administrator
  Change authentication settings
  Change date/time settings
  Change EMD configuration
  Change event settings
  Change external sensor configuration
  Change SHX and other Modbus devices configuration
  Change network configuration
  Change own password
  Change security settings
  Change SNMP settings
  Change user settings
  Change Webcam settings
  Change power supply configuration
  Clear event log
  Firmware update
  Perform reset (warm start)
  View event settings
  View event log
```

```
  View security settings
  View SNMP settings
  View user settings
  View Webcam settings
  View IPDU configuration
  Use groups
  Change groups configuration

Role (1): UserRole
Description:
This is the normal user role
Privileges: (0x0000000007FFE3F9 0x0000000000000000)
  Common user
  Change date/time settings
  Change EMD configuration
  Change event settings
  Change external sensor configuration
  Change SHX and other Modbus devices configuration
  Change network configuration
  Change own password
  Change Webcam settings
  Change power supply configuration
  Clear event log
  Firmware update
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View SNMP settings
  View user settings
  View Webcam settings
  View IPDU configuration
  Use groups
  Change groups configuration

Role (2): ReadOnlyUserRole
Description:
This is the read only user role
```

```
Privileges: (0x0000000003FE0001 0x0000000000000000)
  Common user
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View SNMP settings
  View user settings
  View Webcam settings
  View IPDU configuration
  Use groups

Role (3): ExternalUserRole
Description:
This is the role for an external user
Privileges: (0x0000000003FE0001 0x0000000000000000)
  Common user
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View SNMP settings
  View user settings
  View Webcam settings
  View IPDU configuration
  Use groups

Role (4): Technician
Description:
For Field Engineers
Privileges: (0x00000000035E0201 0x0000000000000023)
  Common user
  Change own password
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View user settings
```

```
  View IPDU configuration
  Use groups
```

16.2 Show a role

16.2.1 Syntax

role show <index>

role show <name>

16.2.2 Purpose

This command shows information about the existing role, by its index in the list or by name. The index of the role in the list, the role name, description and the list of assigned privileges are shown.

16.2.3 Example

```
CLI{admin}> role show 3
Role (3): ExternalUserRole
Description:
This is the role for an external user
Privileges: (0x0000000003FE0001 0x0000000000000000)
  Common user
  Perform reset (warm start)
  View event settings
  View event log
  View security settings
  View SNMP settings
  View user settings
  View Webcam settings
  View IPDU configuration
  Use groups
```

16.3 Create a role

16.3.1 Syntax

role create <name> <description> [<privilege mask> [<outlet mask>]]

role create <name> <description> [<privilege name>...]

16.3.2 Purpose

This command creates a new role and adds it to the list of roles. The user specifies the role name, the description and the privileges for the role. Privileges can be specified either as a hexadecimal mask, or as a sequence of privilege names assigned to the role. The default value for *<privilege mask>* is *0x1*. The optional parameter *<outlet mask>* specifies a set of outlets. The default value for *<outlet mask>* is *0x0*.

The actual privilege names are in the table below.

OFFSET	PRIVILEGE NAME	DESCRIPTION
0	*user*	Common user
1	*admin*	Administrator
2	*auth*	Change authentication settings
3	*dt*	Change date/time settings
4	*emd*	Change EMD configuration
5	*ch_evnt*	Change event settings
6	*ext_sen*	Change external sensor configuration
7	*shx*	Change SHX and other Modbus devices configuration
8	*net*	Change network configuration
9	*pwd*	Change own password
10	*ch_scr*	Change security settings
11	*ch_snmp*	Change SNMP settings
12	*ch_user*	Change user settings
13	*ch_wcam*	Change Webcam settings
14	*ch_ipdu*	Change power supply configuration
15	*clr_elog*	Clear event log
16	*update*	Firmware update
17	*reset*	Perform reset (warm start)
18	*v_evnt*	View event settings
19	*v_elog*	View event log
20	*v_scr*	View security settings
21	*v_snmp*	View SNMP settings
22	*v_user*	View user settings
23	*v_wcam*	View Webcam settings

24	*v_ipdu*	View iPDU configuration
25	*use_grp*	Use groups
26	*ch_grp*	Change group's configuration

16.3.3 Example

```
CLI{admin}> role create Technician "For Field Engineers" user dt reset
Role (4) Technician has been successfully created
CLI{admin}> role create Technician "For Field Engineers" 0x20009 0xF0
Role (4) Technician has been successfully created
```

16.4 Delete a role

16.4.1 Syntax

role delete <name>

16.4.2 Purpose

This command deletes the existing role by its name.

16.4.3 Example

```
CLI{admin}> role delete NewRole
Operation completed successfully
```

16.5 Show or set role description

16.5.1 Syntax

role description <name> [<description>]

16.5.2 Purpose

This command sets or shows the description for the role specified by its name.

If the parameter *<description>* is present, it is set as the role description string; otherwise the current description string is shown.

16.5.3 Example

```
CLI{admin}> role description Technician "For Engineers Only"
Operation completed successfully
CLI{admin}> role description Technician
Technician
Description:
For Engineers Only
Operation completed successfully
```

16.6 Add privileges to the role

16.6.1 Syntax

role privileges add <name> <privilege mask> [<outlet mask>]

role privileges add <name> [<privilege name>...]

16.6.2 Purpose

This command adds the specified privileges and, optionally, set of outlets to the specified role. Added privileges can be specified either as a hexadecimal mask, or as a sequence of privilege names. In the case of the mask, it is ORed to the existing mask of privileges for the role.

See section 16.3 for the list of privilege names.

16.6.3 Example

```
CLI{admin}> role privileges add Technician v_user
Operation completed successfully
CLI{admin}> role privileges add Technician 0x1000000 0x4
Operation completed successfully
```

16.7 Set privileges to the role

16.7.1 Syntax

role privileges set <name> <privilege mask> [<outlet mask>]

role privileges set <name> [<privilege name>...]

16.7.2 Purpose

This command sets the specified set of privileges and, optionally, set of outlets to the specified role. The privileges can be specified either as a hexadecimal mask, or as a sequence of privilege names. In the case of the mask, it replaces the existing mask of privileges for the role.

See section 16.3 for the list of privilege names.

16.7.3 Example

```
CLI{admin}> role privileges set Technician user dt
Operation completed successfully
CLI{admin}> role privileges set Technician 0x9 0x23
Operation completed successfully
```

16.8 Remove privileges from the role

16.8.1 Syntax

role privileges remove <name> <privilege mask> [<outlet mask>]

role privileges remove <name> [<privilege name>...]

16.8.2 Purpose

This command removes the specified set of privileges and, optionally, set of outlets from the specified role. The privileges being removed can be specified either as a hexadecimal mask, or as a sequence of privilege names. In the case of the mask, it is negated and the result is ANDed to the existing mask of privileges for the role.

See section 16.3 for the list of privilege names.

16.8.3 Example

```
CLI{admin}> role privileges remove Technician v_user
Operation completed successfully
CLI{admin}> role privileges remove Technician 0x300000 0x23
Operation completed successfully
```

17 Group Management commands

These commands manage the list of groups that can group together controls and sensors and perform group operations on them.

17.1 List groups

17.1.1 Syntax

group list

group

17.1.2 Purpose

This command lists all existing groups. For each group, the index of the group in the list, the group name and the list of included sensors and controls are shown.

17.1.3 Example

```
CLI{admin}> group list
0: group1
  Controls:
    0: IPDU outlet 1, delay 250, name: Outlet 1: Power Control
    1: IPDU outlet 2, delay 320, name: Outlet 2: Power Control
    2: IPDU outlet 3, delay 500, name: Outlet 3: Power Control
1: Special
  Controls:
    0: resource 1000, control 5, delay 600, name: Pin 0 Control
    1: resource 1000, control 6, delay 600, name: Pin 1 Control
  Sensors:
    0: resource 1000, sensor 1, name: Temperature
    1: resource 3000, sensor 1, name: MCB Temperature
```

17.2 Show information about a group

17.2.1 Syntax

group show <name>

group <name>

17.2.2 Purpose

This command shows information about the existing group: the group name and the list of included sensors and controls are shown.

17.2.3 Example

```
CLI{admin}> group show Special
1: Special
```

```
  Controls:
    0: resource 1000, control 5, delay 600, name: Pin 0 Control
    1: resource 1000, control 6, delay 600, name: Pin 1 Control
  Sensors:
    0: resource 1000, sensor 1, name: Temperature
    1: resource 3000, sensor 1, name: MCB Temperature
```

17.3 Add group

17.3.1 Syntax

group add <name>

group create <name>

17.3.2 Purpose

This command creates a new group and adds it to the list. The new group is empty.

17.3.3 Example

```
CLI{admin}> group add Special
Group 'Special' has been successfully created.
```

17.4 Delete group

17.4.1 Syntax

group delete <name>

group remove <name>

17.4.2 Purpose

This command deletes the existing group by name. All information about sensors and controls in the group is also deleted.

17.4.3 Example

```
CLI{admin}> group delete Special
Group 'Special' has been successfully deleted.
```

17.5 Show progress of an asynchronous assignment in the group

17.5.1 Syntax

group progress <name>

17.5.2 Purpose

This command shows progress of an asynchronous assignment that is currently in progress in the group. The number of already assigned controls and the total number of controls subject to assignment are printed.

17.5.3 Example

```
CLI{admin}> group progress Special
Asynchronous operation for the group 'Special': in progress, 50% (1 out of 2)
```

17.6 Cancel an asynchronous assignment in the group

17.6.1 Syntax

group cancel <name>

17.6.2 Purpose

This command cancels an asynchronous assignment that is currently in progress in the group.

17.6.3 Example

```
CLI{admin}> group cancel Special
Asynchronous operation for the group 'Special' has been cancelled.
```

17.7 Add a new control to the group

17.7.1 Syntax

group control add <name> <resource ID> <control number> <delay ms>

group control add <name> outlet <outlet number> <delay ms>

17.7.2 Purpose

This command adds a new control to the group *<name>*. After setting the value of this control in a group operation there should be a delay of duration *<delay ms>*.

If the outlet parameter is present, the outlet state control of the specified outlet is added to the group.

17.7.3 Example

```
CLI{admin}> group control add Special 4003 1201 3000
Control 4003/1201 has been added to group Special
CLI{admin}> group control add Special outlet 1 3000
Outlet 1 (control 4002/200) has been added to group Special
```

17.8 Delete a control from the group

17.8.1 Syntax

group control delete <name> <resource ID> <control number>

group control delete <name> outlet <outlet number>

group control delete <name> <control index>

17.8.2 Purpose

This command deletes the existing control from the group *<name>*, by its index in the list of controls or by its full designator: resource ID and control number.

If the outlet parameter is present, the outlet state control of the specified outlet is deleted from the group.

17.8.3 Example

```
CLI{admin}> group control delete Special 0
Operation completed successfully
CLI{admin}> group control delete Special 4003 1201
Operation completed successfully
```

17.9 Assign state to all controls in the group

17.9.1 Syntax

group control setstate <name> (manual <state> | auto) [asyn]

17.9.2 Purpose

This command assigns the specified (state to all controls in the group *<name>*, or sets all of them to automatic mode. The *<state>* parameter is *on*, *off*, *pulseon*, *pulseoff* or number. If the parameter `asyn` is specified, the assignment is started in asynchronous mode.

17.9.3 Example

```
CLI{admin}> group control setstate Special manual 0 asyn
Operation completed successfully
```

17.10 Assign reading to all controls in the group

17.10.1 Syntax

group control setvalue <name> <numeric reading> [asyn]

17.10.2 Purpose

This command assigns the specified (numeric) value to all controls in the group *<name>*. If the parameter `asyn` is specified, the assignment is started in asynchronous mode.

17.10.3 Example

```
CLI{admin}> group control setvalue Special 0 asyn
Operation completed successfully
```

17.11 Add a new sensor to the group

17.11.1 Syntax

group sensor add <name> <resource ID> <sensor number>

17.11.2 Purpose

This command adds a new sensor to the group *<name>*.

17.11.3 Example

```
CLI{admin}> group sensor add Special 4007 4210
```

```
Operation completed successfully
```

17.12 Delete a sensor from the group

17.12.1 Syntax

group sensor delete <name> <resource ID> <sensor number>

group sensor delete <name> <sensor index>

17.12.2 Purpose

This command deletes the existing sensor from the group *<name>*, by its index in the list of sensors or by its full designator: resource ID and sensor number.

17.12.3 Example

```
CLI{admin}> group sensor delete Special 4007 4210
Operation completed successfully
CLI{admin}> group sensor delete Special 1
Operation completed successfully
```

17.13 Get aggregate reading of sensors in the group

17.13.1 Syntax

group sensor <operation> <name>

```
operation ::= count | total | min | max | average | square | disp | state-count
| and | or | xor
```

17.13.2 Purpose

This command evaluates the specified aggregate operation over all sensors in the group and shows the result (as a numeric reading or as a state mask, depending on the operation).

17.13.3 Example

```
CLI{admin}> group sensor count Special
UINT64: 2
CLI{admin}> group sensor total Special
FLOAT64: 59.062000
CLI{admin}> group sensor and test
Aggregated event state: 0x0004
```

17.14 Set threshold and hysteresis values

17.14.1 Syntax

group sensor threshold set <name> <thr name1> <value1> [<thr name2> <value2> …]

```
<thr name> ::= ucr | umj | umn | lcr | lmj | lmn | phy | nhy
```

17.14.2 Purpose

This command sets the listed threshold and hysteresis parameters to the provided values for all the sensors in the group *<name>*.

The threshold and hysteresis names have the following meaning:

`ucr` = Upper Critical Threshold

`umj` = Upper Major Threshold

`umn` = Upper Minor Threshold

`lcr` = Lower Critical Threshold

`lmj` = Lower Major Threshold

`lmn` = Lower Minor Threshold

`phy` = Positive Hysteresis

`nhy` = Negative Hysteresis

The *<value>* parameters are either numeric or the *disable* keyword.

17.14.3 Example

In the example below the group *test* contains two temperature sensors, the Lower Minor Threshold for each sensor in the group is set to *-20* and the Upper Minor Threshold for each sensor in the group is set *80*. Other threshold and hysteresis values are not set in the group operation and, as a result, are set to the value *Not set*.

```
CLI{admin}> group list
0: test
  Sensors:
    0: resource 1000, sensor 1, name: Temperature 1
    1: resource 1000, sensor 2, name: Temperature 2
CLI{admin}> group sensor threshold  set test lmn -20 umn 80
CLI{admin}> sensor threshold 1000 1
         Lower Minor -- type: INT64; value:   -20
         Lower Major -- Not set
      Lower Critical -- Not set
         Upper Minor -- type: INT64; value:   80
         Upper Major -- Not set
      Upper Critical -- Not set
 Positive Hysteresis -- Not set
 Negative Hysteresis -- Not set
CLI{admin}> sensor threshold 1000 2
         Lower Minor -- type: INT64; value:   -20
         Lower Major -- Not set
      Lower Critical -- Not set
         Upper Minor -- type: INT64; value:   80
```

```
        Upper Major -- Not set
     Upper Critical -- Not set
Positive Hysteresis -- Not set
Negative Hysteresis -- Not set
```

18 Security commands

This set of commands works with the firewall and role-based firewalls. The commands work for either IPv4 or IPv6 protocols. When the optional *ipv6* parameter is present, the IPv6 firewall is taken into account. When the optional *ipv6* parameter is omitted, the IPv4 firewall is taken into account.

18.1 Show firewall status

18.1.1 Syntax

firewall info

firewall list

18.1.2 Purpose

This command shows the firewall status and its policies.

18.1.3 Examples

```
CLI{admin}> firewall info
Firewall: Disabled
CLI{admin}> firewall list
Firewall: Enabled
Default policy: ACCEPT
CLI{admin}> firewall list
Firewall: Enabled
Default policy: ACCEPT
1: 102.168.1.127: REJECT
2: 192.168.1.127: ACCEPT
```

18.2 Enable the firewall

18.2.1 Syntax

firewall enable [ipv6]

18.2.2 Purpose

This command enables the firewall. When the optional *ipv6* parameter is present, the IPv6 firewall is enabled. When the optional *ipv6* parameter is omitted, the IPv4 firewall is enabled.

18.2.3 Examples

```
CLI{admin}> firewall enable
```

18.3 Disable the firewall

18.3.1 Syntax

firewall disable [ipv6]

18.3.2 Purpose

This command disables the firewall. When the optional *ipv6* parameter is present, the IPv6 firewall is disabled. When the optional *ipv6* parameter is omitted, the IPv4 firewall is disabled.

18.3.3 Examples

```
CLI{admin}> firewall disable
```

18.4 Set/get the default policy

18.4.1 Syntax

firewall default [ipv6] [<policy>]

```
<policy> ::= accept | drop
```

18.4.2 Purpose

When the optional parameter <*policy*> is present, this command sets the policy specified by this parameter as the default policy. When the optional parameter <*policy*> is omitted, this command reports the default policy

When the optional *ipv6* parameter is present, the IPv6 firewall is taken into account. When the optional *ipv6* parameter is omitted, the IPv4 firewall is taken into account.

18.4.3 Examples

```
CLI{admin}> firewall default
Default policy: ACCEPT
CLI{admin}> firewall default drop
```

18.5 Add a rule

18.5.1 Syntax

firewall add [ipv6] <destination > <policy>

```
<policy> ::= accept | drop | reject
```

18.5.2 Purpose

This command adds a new rule to the firewall. Use this command when the firewall is enabled.

18.5.3 Examples

```
CLI{admin}> firewall add 102.168.1.127 reject
```

18.6 Insert a rule

18.6.1 Syntax

firewall insert [ipv6] <index> <destination> <policy>

```
<policy> ::= accept | drop | reject
```

18.6.2 Purpose

This command inserts a new rule at index *<index>* (1-based). Use this command when the firewall is enabled.

18.6.3 Examples

```
CLI{admin}> firewall insert 1 102.168.1.127 accept
```

18.7 Modify a rule

18.7.1 Syntax

firewall modify [ipv6] <index> <destination> <policy>

```
<policy> ::= accept | drop | reject
```

18.7.2 Purpose

This command modifies the rule specified by the *<index>* parameter. Use this command when the firewall is enabled.

18.7.3 Examples

```
CLI{admin}> firewall modify 1 102.168.1.127 drop
```

18.8 Delete a rule

18.8.1 Syntax

firewall delete [ipv6] <index>

18.8.2 Purpose

This command deletes the rule specified by the *<index>* parameter. Use this command when the firewall is enabled.

18.8.3 Examples

```
CLI{admin}> firewall delete 1
```

18.9 Show role-based firewall status

18.9.1 Syntax

role_firewall info

role_firewall list

18.9.2 Purpose

This command shows the role-based firewall status and its policies.

18.9.3 Examples

```
CLI{admin}> role_firewall list
RoleBasedFirewall: Disabled
CLI{admin}> role_firewall list
RoleBasedFirewall: Enabled
Default policy: ALLOW
1: ALLOW        192.168.1.149-192.168.1.144     SpecialRole
```

```
2: ALLOW        192.168.1.137-192.168.1.144     GuestRole
```

18.10 Enable the role-based firewall

18.10.1 Syntax

role_firewall enable [ipv6]

18.10.2 Purpose

This command enables the role-based firewall. When the optional *ipv6* parameter is present, the IPv6 firewall is enabled. When the optional *ipv6* parameter is omitted, the IPv4 firewall is enabled.

18.10.3 Examples

```
CLI{admin}> role_firewall enable
```

18.11 Disable the role-based firewall

18.11.1 Syntax

role_firewall disable [ipv6]

18.11.2 Purpose

This command disables the role-based firewall. When the optional *ipv6* parameter is present, the IPv6 firewall is disabled. When the optional *ipv6* parameter is omitted, the IPv4 firewall is disabled.

18.11.3 Examples

```
CLI{admin}> role_firewall disable
```

18.12 Set/get the default policy for role-based firewall

18.12.1 Syntax

role_firewall default [ipv6] [<policy>]

```
<policy> ::= allow | deny
```

18.12.2 Purpose

When the optional parameter <*policy*> is present, this command sets the policy specified by this parameter as the default policy. When the optional parameter <*policy*> is omitted, this command reports the default policy

When the optional *ipv6* parameter is present, the IPv6 firewall is taken into account. When the optional *ipv6* parameter is omitted, the IPv4 firewall is taken into account.

18.12.3 Examples

```
CLI{admin}> role_firewall default
Default policy: ALLOW
```

18.13 Add a rule to role-based firewall

18.13.1 Syntax

role_firewall add [ipv6] <policy> <start_ip> <end_ip> <role list>

<policy> ::= allow | deny

18.13.2 Purpose

This command adds a new rule to the firewall.

18.13.3 Examples

CLI{admin}> **role_firewall add allow 192.168.1.149 192.168.1.144 SpecialRole**

18.14 Insert a rule to role-based firewall

18.14.1 Syntax

role_firewall insert [ipv6] <index> <policy> <start_ip> <end_ip> <role list>

<policy> ::= allow | deny

18.14.2 Purpose

This command inserts a new rule at index *<index>* (1-based).

18.14.3 Examples

CLI{admin}> **role_firewall insert 1 allow 192.168.1.149 192.168.1.144 SpecialRole**

CLI{admin}> **role_firewall add ipv6 allow fe80::3efb:96ff:fe77:8812 fe80::3efb:96ff:fe77:88ab PowerOn,Technician**

18.15 Modify a rule in role-based firewall

18.15.1 Syntax

role_firewall modify [ipv6] <index> <policy> <start_ip> <end_ip> <role list>

<policy> ::= allow | deny

18.15.2 Purpose

This command modifies the rule specified by the *<index>* parameter.

18.15.3 Examples

CLI{admin}> **role_firewall modify 3 deny 192.168.1.149 192.168.1.144 SpecialRole**

18.16 Delete a rule in role-based firewall

18.16.1 Syntax

role_firewall delete [ipv6] <index>

18.16.2 Purpose

This command deletes the rule specified by the *<index>* parameter.

18.16.3 Examples

```
CLI{admin}> role_firewall delete 1
```

18.17 Verify that login is allowed for user

18.17.1 Syntax

role_firewall [ipv6] verify <username> <source ip>

18.17.2 Purpose

This command reports whether or not login for a specified user is allowed from the specified source IP address.

18.17.3 Examples

```
CLI{admin}> role_firewall verify guest 192.168.1.144
User guest: login from 192.168.1.144 is allowed
CLI{admin}> role_firewall verify guest 192.168.1.149
User guest: login from 192.168.1.144 is denied
```

19 Login restrictions

19.1 Get login restrictions

19.1.1 Syntax

loginrestrictions

19.1.2 Purpose

This command shows the “login restrictions” data structure. The detailed description of the restrictions is presented in the table below.

19.1.3 Examples

```
CLI{admin}> loginrestrictions
Login restrictions:
   AllowMultipleLogons: true
   PasswordAging: true
   PasswordAgingInterval (days): 4
   IdleTimeout (seconds): 0
   LockAfterFailedAttempts: 3
   LockTime (seconds): 1200
   StrongPasswords: true
   MinStrongPasswordLength: 6
   AtLeastOneLcCharacter: false
   AtLeastOneUcCharacter: false
   AtLeastOneNumCharacter: false
   AtLeastOneSpecCharacter: false
   PasswordHistoryDepth: 0
```

19.2 Set login restrictions

19.2.1 Syntax

loginrestrictions <param1> … <param13>

19.2.2 Purpose

This command sets login restrictions. The following 13 parameters are required. Boolean parameters must be specified as *false* for “false” and *true* for “true”. For numeric parameters, the new numeric value should be specified.

PARAMETER NUMBER	DESCRIPTION	TYPE
1	This parameter specifies whether multiple logons are allowed	Boolean

2	This parameter specifies whether the password aging is enabled	Boolean
3	This parameter specifies the Password Aging interval (in days)	Number
4	This parameter specifies the Idle Timeout interval (in seconds)	Number
5	This parameter specifies the number of failed login attempts after which the user is locked	Number
6	This parameter specifies the Locked interval (in seconds)	Number
7	This parameter specifies whether the Strong Password requirements are enabled	Boolean
8	This parameter specifies the Minimum Length of Strong Password	Number
9	This parameter specifies that at least one lowercase alphabetic character is required in the password	Boolean
10	This parameter specifies that at least one uppercase alphabetic character is required in the password	Boolean
11	This parameter specifies that at least one numerical character is required in the password	Boolean
12	This parameter specifies that at least one special character is required in the password	Boolean
13	This parameter indicates the number of previous passwords that shall be maintained for the account	Numerical

If the value of parameter 2 is set *false* (the password aging is not enabled) then the value of parameter 3 (the Password Aging interval) defaults to *0*.

19.2.3 Examples

```
CLI{admin}> loginrestrictions true true 90 0 3 1200  false 6 false false false false 0
```

19.3 Get user status

19.3.1 Syntax

loginrestrictions <user>

19.3.2 Purpose

This command reports whether a user is locked due to the login restrictions.

19.3.3 Example

```
CLI{admin}> loginrestrictions xyz
User xyz is not locked
CLI{admin}> loginrestrictions abc
User abc is locked
```

19.4 Unlock a user

19.4.1 Syntax

loginrestrictions unlock <user>

19.4.2 Purpose

This command unlocks a user that was locked due to the login restrictions.

19.4.3 Example

```
CLI{admin}> loginrestrictions unlock xyz
```

20 Language support

20.1 Syntax

system language [<lang>]

<lang> ::= english | french | german

20.2 Purpose

When the optional parameter *<lang>* is omitted this command reports the system language setting. When the optional parameter *<lang>* is present this command sets the system language setting.

20.3 Examples

```
CLI{admin}> system language
Current language: English
CLI{admin}> system language german
Aktuelle Sprache: Deutsch
```

21 AWS support

21.1 AWS certificates download

21.1.1 Syntax

system awsconfig [<thing name> [<directory]]

21.1.2 Purpose

This command retrieves the AWS certificates and the configuration file for the specified *<thing_name>* from the cloud. The thing must be previously registered and the certificates stored in the database by an authorized AWS user. The MAC address of the iPDU must match the MAC address stored in the database for this *<thing_name>*.

If the command succeeds, the configuration file is written into the directory *<directory>*, and the iPDU private key and certificate are written in the directory *<directory>/certs*. The *<directory>* must be the configuration directory of the iPDU. Then, after a restart, the iPDU will be able to connect to the AWS cloud.

The default value for *<thing_name>* is the serial number of the iPDU.

The default value for *<directory>* is */etc/config*.

21.1.3 Examples

```
CLI{admin}> system awsconfig 00001234
CLI{admin}> system awsconfig 00001234 /mnt/var/config
```

21.2 Shows AWS status

21.2.1 Syntax

system aws

21.2.2 Purpose

This command reports the AWS status of the device.

21.2.3 Examples

```
CLI{admin}> system aws
Connected to AWS as "Ipdu123456"
CLI{admin}> system aws
AWS configuration does not exist for this system!
```

22 Restart, reset and terminate

22.1 Restart the system

22.1.1 Syntax

restart

22.1.2 Purpose

This command restarts the system, but does not reboot the hardware and does not reset the settings to the factory defaults.

22.1.3 Examples

```
CLI{admin}> restart
```

22.2 Reboot the system

22.2.1 Syntax

reboot

22.2.2 Purpose

This command reboots the hardware, restarts the system, but does not reset the settings to the factory defaults

22.2.3 Example

```
CLI{admin}> reboot
```

22.3 Reset the system

22.3.1 Syntax

factory_reset

22.3.2 Purpose

This command erases the configuration files and restarts the system.

22.3.3 Examples

```
CLI{admin}> factory_reset
```

22.4 Terminate the system

22.4.1 Syntax

terminate

22.4.2 Purpose

This command terminates the system.

22.4.3 Examples

```
CLI{admin}> terminate
```

22.5 Debug level

22.5.1 Syntax

debug [<debuglevel>]

22.5.2 Purpose

This command shows, or sets, the debug level.

22.5.3 Example

```
CLI{admin}> debug 3
Current debug level: 3 (Error,Warning)
CLI{admin}> debug
Current debug level: 7 (Error,Warning,Info)
```

22.6 Firmware upgrade

22.6.1 Syntax

upgrade [<parameters>]

22.6.2 Purpose

This command upgrades the firmware. A new firmware file is specified by its URL. The optional *<parameters>* consists of flags and URL. Flags are listed below:

- *-i*, print information about image(s) and exit
- *-f*, ignore component versions and force an upgrade
- *-a*, activate (boot into) new firmware after the upgrade
- *-c*, copy new images to the old partition after switching
- *-b*, run in the background (suppresses all output)
- *-e*, erase the */var/nvdata/etc* directory upon reboot
- *-E*, erase the entire non-volatile partition upon reboot
- *-p product_code*, verify product code before upgrading.

If the optional parameter is not present the command outputs the list of files for upgrade in the */mnt/usb/* directory.

22.6.3 Example

```
CLI{admin}>upgrade -f tftp://192.168.1.253/latest_firmware.dat
Running: rupgrade -k /var/data/public.key  -f
tftp://192.168.1.253/latest_firmware.dat
rupgrade v0.34, (C) nVent
[I]: Current partition number: 1
[I]: Processing input file: "tftp://192.168.1.253/latest_firmware.dat"
[I]: Fetching URL "tftp://192.168.1.253/ latest_firmware.dat"
[I]: Verifying signature...
```

```
[I]: Signature is VALID
[I]: Programming kernel to partition 0...
[I]: Erasing "/dev/mtd1"...
Erasing kernel:
################################################################################
#############[100%]
[I]: Programming kernel data to "/dev/mtd1" (5439463 bytes)...
…
CLI{admin}> upgrade
List of files for upgrade in /mnt/usb/
        System Volume Information/WPSettings.dat
        upgrade_dir/latest_firmware.dat
CLI{admin}> upgrade -i tftp://80.240.102.58/ipdu-latest.dat
Running: smrcli.setuid upgrade -i tftp://80.240.102.58/latest_firmware.dat
uid=0, euid=0, rc=0, errno=0
Running: rupgrade -k /var/data/public.key  -i
tftp://80.240.102.58/latest_firmware.dat
rupgrade v0.34, (C) nVent
[I]: Current partition number: 0
[I]: Processing input file: "tftp://80.240.102.58/latest_firmware.dat"
[I]: Fetching URL "tftp://80.240.102.58/latest_firmware.dat"
[I]: Upgrade image version: 0.98.9, size: 62684796 bytes
[I]: Valid rootfs image found (version: 0.85, size: 56770641 bytes)
[I]: Valid kernel image found (version: 0.18, size: 5492907 bytes)
[I]: Valid U-Boot image found (version: 0.17, size: 420928 bytes)
[I]: Verifying signature...
[I]: Signature is VALID
```

22.7 Update the Guardian Management Gateway configuration

22.7.1 Syntax

config_write start

22.7.2 Purpose

This command initiates an immediate collection of system configuration and writing it to the Inlet EEPROM.

22.7.3 Example

```
CLI{admin}> config_write start
Reservation id: 1
```

22.8 Status of the Guardian Management Gateway configuration update

22.8.1 Syntax

config_write status <reservation id>

22.8.2 Purpose

This command reports the status of the forced configuration write.

22.8.3 Example

```
CLI{admin}> config_write status 1
Configuration write status for reservation Id 1: In Progress/Not Started
CLI{admin}> config_write status 1
Configuration write status for reservation Id 1: Completed
```

22.9 Save and load the configuration

22.9.1 Print list of upgrade files

22.9.1.1 Syntax

config list

22.9.1.2 Purpose

This command prints the list of files for upgrade in */mnt/usb/* directory.

22.9.1.3 Example

```
CLI{admin}> config list
List of configuration files in /mnt/usb/
    complete.cfg.tgz
    reduced.cfg.tgz
    tech.cfg.tgz
```

22.9.2 Load the Guardian Management Gateway configuration

22.9.2.1 Syntax

config load <URL>

22.9.2.2 Purpose

This command loads the iPDU configuration from the source *<URL>* and restarts the system. If the *<URL>* parameter specifies a relative path, then the configuration is loaded from */mnt/usb* directory.

22.9.2.3 Example

```
CLI{admin}> config load complete.cfg.tgz
Restart request has been issued
```

```
CLI{admin}> config load /tmp/complex.cfg.tgz
Restart request has been issued
CLI{admin}> config load storage@192.168.1.253:conf/complex.cfg.tgz
storage@192.168.1.253's password:
complex.cfg.tgz                                                    100%
835     0.8KB/s   00:00
Restart request has been issued
```

22.9.3 Save the Guardian Management Gateway configuration to USB Flash drive

22.9.3.1 Syntax

config save <component mask> (<file name> | <URL>)

config save (<component name>... | all) (<file name> | <URL>)

22.9.3.2 Purpose

This command saves a set of components of the iPDU configuration (in tarred and gzipped format) and copies it to the USB Flash drive or to *<URL>* destination via SCP protocol. Components can be specified either as a hexadecimal mask, or as a sequence of component names. If the *all* parameter is specified, the whole iPDU configuration is saved. If the *<file name>* parameter specifies a relative path, then the configuration is saved in */mnt/usb* directory. The suffix *.cfg.tgz* is appended to the configuration file name.

The actual component names are in the table below.

OFFSET	COMPONENT NAME	DESCRIPTION
0	global	Global settings
1	network	Network configuration settings
2	hostname	Host name
3	services	Network service configuration settings
4	users	List of users and their properties
5	roles	List of roles and their properties
6	snmp_users	SNMPv3 user settings
7	security	Security settings (firewalls and login restrictions)
8	ssl_cert	SSL certificate for the HTTP server
9	ldap	LDAP settings
10	rules	Rules for event handling with corresponding actions

11	resources	User-defined resource names
12	sensors	Configuration of physical sensors (with user-defined sensor names)
13	controls	Configuration of controls (user names assigned to controls)
14	groups	List of sensor/control groups, their contents and properties
15	managed_sensors	List of managed sensors and their properties
16	reachability	Server reachability settings
17	w1_dev_map	Resource map for 1-wire devices
18	modbus_dev_map	Resource map for Modbus devices
19	nominal	iPDU nominal frequency and voltage
20	outlet_sequence	iPDU outlet startup sequence
21	outlet_init	iPDU outlet initial state
22	outlet_names	iPDU user-defined outlet names
23	outlet_state	iPDU current outlet state

22.9.3.3 Example

```
CLI{admin}>config save 0x7 simple.cfg.tgz
Reservation id: 3
Configuration saved as: /mnt/usb/simple.cfg.tgz
CLI{admin}> config save users roles hostname /mnt/usb/short
Reservation id: 5
Configuration saved as: /mnt/usb/short.cfg.tgz
CLI{admin}> config save all total.cfg.tgz
Reservation id: 2
Configuration saved as: /mnt/usb/total.cfg.tgz
CLI{admin}> config save 0x01 storage@192.168.1.253:device_config/
Reservation id: 2
Configuration write complete in 1540 ms
```

```
scp /tmp/config_saved_2.tgz storage@192.168.1.253:device_config/
storage@192.168.1.253's password:
config_saved_2.tgz                            100%   95     0.1KB/s   00:00
```

23 LDAP configuration

23.1 Show LDAP configuration

23.1.1 Syntax

ldap

ldap info

23.1.2 Purpose

This command shows the LDAP service configuration of the device.

23.1.3 Example

```
CLI{admin}> ldap info
LDAP: enabled
   Server URI: ppswinldap.winldap.pps
   Server type: Active Directory
   Use SSL: true
   SSL Port: 389
   Server certificate: /etc/ssl/certs/rootca.crt.win
   Use anonymous bind: false
   Bind DN: nssproxy@winldap.pps
   Bind password: nssproxy
   Search base DN: dc=winldap,dc=pps
   Login name attribute: sAMAccountName
   User entry object class: User
   User search subfilter:
CLI{admin}> ldap info
LDAP: disabled
   Server URI: ldapserver.ppstest
   Server type: OpenLDAP
   Use SSL: false
   Use anonymous bind: false
   Bind DN: nssproxy
   Bind password: nssproxy
   Search base DN: CN=ppstest
   Login name attribute:
   User entry object class:
   User search subfilter:
```

23.2 Disable LDAP

23.2.1 Syntax

ldap disable

23.2.2 Purpose

This command disables the LDAP service.

23.2.3 Example

```
CLI{admin}> ldap disable
```

23.3 Set the LDAP configuration

23.3.1 Syntax

ldap <server-uri> <server-type> <use-ssl> [<ssl-port> <ssl-cert-file-path>] <use-anonymous> [<bind-dn> <bind-password>] [<search-base-dn>] [<login-attribute>] [<user-entry-object-class>] [<user-search-subfilter>] [<extra-config-options>]

```
<server-type> ::= openldap | activedirectory

<use-ssl> ::= true | false

<use-anonymous> ::= true | false
```

23.3.2 Purpose

This command sets the LDAP service configuration.

If the parameter *<use-ssl>* is set *false*, then the parameters *<ssl-port>* and *<ssl-cert-file-path>* are omitted.

If the parameter *<use-anonymous>* is set *true*, then the parameters *<bind-dn>* and *<bind-password>* are omitted.

23.3.3 Examples

```
CLI{admin}> ldap ldap://ldapserver.ppstest openldap false false nssproxy
nssproxy CN=ppstest uid user

ldap ldap://ldapserver.ppstest openldap false false
cn=nssproxy,ou=users,dc=ppstest nssproxy dc=ppstest

ldap ldap://ppswinldap.winldap.pps activedirectory false false
nssproxy@winldap.pps nssproxy   dc=winldap,dc=pps

ldap ldap://ldapserver.ppstest openldap true 24 /etc/ssl/certs/ldap.crt false
cn=nssproxy,ou=users,dc=ppstest nssproxy dc=ppstest
```

24 SSL certificate management

24.1 Show active certificate

24.1.1 Syntax

sslcert [active]

24.1.2 Purpose

This command shows the active SSL certificate.

24.1.3 Example

```
CLI{admin}>sslcert
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            de:d6:98:00:8f:6f:d2:fd
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=AU, ST=New South Wales, L=Sydney, O=SomeWhere Ltd,
OU=TechDept, CN=John Smith/emailAddress=john.smith@somewhere.com
        Validity
            Not Before: Feb 17 11:41:42 2017 GMT
            Not After : Feb 12 11:41:42 2037 GMT
        Subject: C=AU, ST=New South Wales, L=Sydney, O=SomeWhere Ltd,
OU=TechDept, CN=John Smith/emailAddress=john.smith@somewhere.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:e6:17:99:02:f7:c2:64:41:c4:bb:3f:3b:6e:25:
                    2d:5b:be:d1:ae:4e:4e:9a:09:72:8d:fc:bd:4a:8e:
                    c3:69:07:a9:f5:8f:46:3c:00:39:d3:9b:8e:73:1a:
                    51:6c:1f:0d:0d:12:10:0e:b0:83:79:db:c8:89:72:
                    7f:33:c1:31:b2:b1:ed:58:e7:45:12:e5:1d:a2:a2:
                    27:e1:04:c5:1c:39:5e:7b:3f:92:e2:b1:d2:50:ff:
                    a2:f1:d3:48:5c:b9:62:01:cd:fa:1e:e7:f2:dc:05:
                    76:d1:49:a2:64:de:fd:14:8d:54:ae:9a:ef:ec:4b:
                    7a:d1:c8:60:b8:6a:6b:2b:77:fa:fd:51:b0:00:5a:
                    f3:44:ff:2a:88:5d:a5:fe:1e:01:35:00:99:ca:fd:
                    a0:80:cf:52:62:3e:3a:67:06:a9:d9:57:a3:d5:59:
```

```
                    8e:bf:ef:74:5c:58:40:4b:de:56:43:d6:df:6d:d5:
                    94:ac:7d:11:56:26:78:c5:58:3c:0e:90:23:43:14:
                    b3:cf:ae:d5:37:86:4d:a7:b4:3a:1d:23:e3:9c:e4:
                    b1:14:c2:67:a7:01:dd:15:38:2b:77:0a:84:44:02:
                    e0:27:b9:13:35:9a:4f:66:8c:1b:ef:5a:d9:49:73:
                    cc:e2:5d:04:23:21:f4:8e:81:5b:ec:28:ea:12:30:
                    1a:43
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7A:0C:33:41:71:27:7A:E8:14:B5:1C:04:F0:B6:79:FD:B5:7E:8D:06
            X509v3 Authority Key Identifier:

keyid:7A:0C:33:41:71:27:7A:E8:14:B5:1C:04:F0:B6:79:FD:B5:7E:8D:06

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         47:a1:af:7c:91:06:41:a7:09:26:0e:ed:b0:04:ee:d5:e1:0d:
         29:3b:fa:7e:4c:7f:60:ec:20:06:14:7c:60:5b:f5:0e:1f:e3:
         38:99:b0:cc:80:b1:2b:4f:97:35:70:a4:4b:3a:42:64:fb:1b:
         47:be:e3:fc:bc:6e:d5:c7:56:e9:97:53:cc:b3:0e:d8:13:f1:
         cd:79:05:5e:b6:ee:fa:19:b0:e5:97:62:8d:19:3b:ef:8e:d1:
         14:83:1c:ef:cb:a2:72:be:d9:f2:c3:2b:81:62:85:c3:58:f3:
         2f:2d:d5:63:1c:ef:e7:5d:df:68:00:96:f5:00:b1:5a:0e:44:
         98:a6:72:5e:5b:da:91:b4:2d:97:0e:46:8c:42:9f:c2:a9:1f:
         c9:73:f1:aa:a8:79:28:6b:1d:2d:fd:32:c8:38:b5:82:28:e9:
         62:dd:6f:3f:07:b9:71:0f:fe:cd:91:6f:41:67:26:68:63:86:
         ec:c4:d7:ec:82:82:6f:8e:aa:30:98:31:3e:69:d2:11:28:a5:
         11:ad:89:44:8e:82:f5:75:e6:d0:17:de:4f:f5:2c:fa:eb:ce:
         73:e4:28:3b:98:fd:31:54:0c:b2:6c:44:71:1d:7a:d6:6e:90:
         80:37:96:e2:54:01:60:21:4f:32:31:b4:58:07:9e:90:31:cb:
         92:79:f4:6c
```

24.2 Show list of certificates

24.2.1 Syntax

sslcert list

24.2.2 Purpose

This command shows the list of SSL certificates.

24.2.3 Example

```
CLI{admin}>sslcert list
        domain.com.csr
        test.pem
        test2.csr
        test2.pem

certs/978b6fb835b71caaa02548c70235d55d915ad2523367813b81c249ff7cd00bdc.pem
certs/978b6fb835b71caaa02548c70235d55d915ad2523367813b81c249ff7cd00bdc.pub
        myCsr.csr
```

24.3 Show certificate info

24.3.1 Syntax

sslcert show <cert-or-csr-file>

24.3.2 Purpose

This command shows information about a certificate or CSR.

24.3.3 Example

```
CLI{admin}>sslcert show
certs/978b6fb835b71caaa02548c70235d55d915ad2523367813b81c249ff7cd00bdc.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            13:c8:f3:c7:66:64:dd:a8:1d:ca:4d:72:ed:ae:b8:47:37:30:1f:81
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington
C=US
        Validity
            Not Before: Jul 16 16:16:54 2018 GMT
            Not After : Dec 31 23:59:59 2049 GMT
        Subject: CN=AWS IoT Certificate
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
```

```
                    00:d9:52:7d:05:19:1b:1e:25:e0:c9:8a:ab:37:9b:
                    b9:68:3a:92:89:f6:7d:1b:c2:09:a4:5a:01:6e:d2:
                    e1:d8:88:f2:55:bf:0c:de:45:26:ea:06:ea:e5:12:
                    1e:4c:50:72:81:48:79:f3:e0:b2:d8:0c:ab:8d:7b:
                    37:34:3d:8d:79:6d:cc:8e:48:36:d7:dd:2e:14:f5:
                    3f:4f:17:25:db:cd:cd:b9:fe:3d:75:c6:b1:35:69:
                    25:94:de:a2:6e:d1:00:73:e0:eb:47:b1:1e:5a:a2:
                    9d:7c:80:f9:c4:27:b5:86:b2:f8:81:26:76:2c:de:
                    40:72:5d:54:ce:24:32:73:bf:81:66:9e:24:02:4f:
                    6d:05:5c:0b:07:30:26:b8:87:7b:0b:13:07:53:d3:
                    c8:ac:d6:fd:31:cd:9f:c6:6c:32:f8:4d:6f:05:ef:
                    fa:09:ad:ce:ff:69:2e:04:9e:28:b2:6a:37:1c:63:
                    f8:b8:0e:03:ae:ae:ea:63:0b:7e:e3:0d:7b:4d:5c:
                    53:37:9b:73:3f:8f:f5:39:5e:1c:1f:bc:dc:a4:e9:
                    86:50:44:64:eb:36:d1:66:fd:30:c3:e2:2b:c5:00:
                    d1:9f:50:de:e9:db:92:96:2a:e7:ae:18:60:19:61:
                    9f:72:a6:23:02:40:44:d1:c3:1a:bf:ec:83:56:9f:
                    65:af
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:

keyid:82:6C:68:4C:37:A8:02:4C:17:D0:0B:6C:8F:05:FD:34:E8:65:A8:3B

            X509v3 Subject Key Identifier:
                E1:4F:26:16:E0:92:F3:37:F6:CE:AA:CA:90:AE:9F:E0:06:57:39:F4
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature
    Signature Algorithm: sha256WithRSAEncryption
         c5:35:35:39:e8:b6:01:b9:84:39:0f:d9:10:b2:24:27:b7:35:
         73:61:7f:03:5f:78:7e:d9:7c:74:2b:e5:10:a6:99:71:0e:d3:
         3f:32:f7:b6:86:6e:54:d3:67:ed:db:04:50:a2:5e:ce:5a:86:
         84:77:b4:b2:30:d3:3e:b5:da:ba:e6:d8:69:50:f7:89:b8:74:
         5d:1d:7f:53:6d:4f:3c:17:dc:85:d4:c9:ac:d7:92:52:a9:45:
         1d:20:e5:92:13:ab:9e:ba:98:26:66:89:7e:89:12:b2:a4:61:
         89:65:44:7a:82:44:7f:0a:e3:80:3a:18:ec:74:53:f8:60:e1:
```

```
02:42:61:e1:9f:c0:5f:ac:6c:96:d1:a9:86:ab:c8:ea:48:6b:
e4:33:94:05:4c:66:2a:c8:8d:30:8a:32:36:d9:be:33:2e:cc:
11:ef:1b:28:e7:9c:a2:ca:6e:d3:2c:c4:9b:2d:2a:c2:77:f2:
5d:ae:82:81:fa:ba:e9:60:01:67:51:bb:22:85:03:76:fe:25:
d5:f7:35:2c:28:a7:d1:61:dd:f5:32:83:33:ad:3c:be:3b:f1:
d2:74:a5:c2:0e:fb:14:22:07:84:7d:36:7a:c4:a7:3d:82:38:
39:fb:1f:f6:7d:05:6f:e7:31:4b:e8:8d:31:f6:8c:cc:13:1e:
```

24.4 Generate certificate or certificate sign request

24.4.1 Syntax

sslcert generate cert <cert_file>

sslcert generate csr <csr_file>

24.4.2 Purpose

If the *cert* parameter is present this command generates a new self-signed certificate and stores it at the URL specified by *<cert_file>* parameter.

If the *csr* parameter is present this command generates a certificate sign request (CSR) and stores it at the URL specified by *<csr_file>* parameter.

24.4.3 Example

```
CLI{admin}>sslcert generate cert temp
Generating a 4096 bit RSA private key
.....................................................................................
.....................++
..............................++
writing new private key to '/home/root/temp.pem.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile-de-France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SomeWhere Ltd
Organizational Unit Name (eg, section) []:TechDepartment
```

```
Common Name (e.g. server FQDN or YOUR name) []:John Smith
Email Address []:somebody@somewhere.com
CLI{admin}> sslcert generate csr request
Generating a 2048 bit RSA private key
..+++
.....................................................................
.....................................................................
..............+++
writing new private key to '/home/root/request.csr.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile-de-France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SomeWhere Ltd
Organizational Unit Name (eg, section) []:TechDepartment
Common Name (e.g. server FQDN or YOUR name) []:John Smith
Email Address []:somebody@somewhere.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456
An optional company name []:
```

24.5 Delete certificate or certificate sign request

24.5.1 Syntax

sslcert delete <cert-or-csr-file>

24.5.2 Purpose

This command deletes a certificate or a CSR stored in *<cert-or-csr-file>* file.

24.5.3 Example

```
CLI{admin}>sslcert delete temp.pem
```

24.6 Install certificate

24.6.1 Syntax

sslcert install <cert-file>

24.6.2 Purpose

This command installs a certificate from *<cert-file>* file.

24.6.3 Example

```
CLI{admin}>sslcert install scp://remote@192.168.1.143:temp.pem
remote@192.168.1.143's password:*****
temp.pem                                      100% 5427     5.3KB/s   00:00
Operation completed successfully
```

24.7 Copy certificate

24.7.1 Syntax

sslcert copy <source_location> <dest_location>

24.7.2 Purpose

This command copies a certificate or CSR from one location to another. Either of *<source_location>* or *<dest_location>* can be an SCP URL (starting with *scp://*), in this case the corresponding file is copied from or to a remote location. However, copying between two local locations is also supported by this command. A relative path is relative to the user's home directory.

24.7.3 Example

```
CLI{admin}>sslcert copy scp://john@192.168.1.93:/etc/ssl/certs/WoSign.pem
IpduSign.pem
john@192.168.1.93's password:******
WoSign.pem                          100% 1956     1.9KB/s   00:00
Operation completed successfully
```

25 Restricted Service Agreement

25.1 Get current status and full text of the Restricted Service Agreement

25.1.1 Syntax

restricted_service_agreement

25.1.2 Purpose

This command reports the current status and the full text of the Restricted Service Agreement.

25.1.3 Example

```
CLI{admin}> restricted_service_agreement
Enforced: No
Text:
Unauthorized access prohibited; all access and activities not explicitly
authorized by the management are unauthorized. All activities are monitored
and logged.

There is no privacy on this system.
Unauthorized access and activities or any criminal activity will be
reported to the appropriate authorities.
```

25.2 Show, or set, the Restricted Service Agreement status

25.2.1 Syntax

restricted_service_agreement enforce [(yes|no)]

25.2.2 Purpose

This command shows, or sets, the Restricted Service Agreement status.

25.2.3 Example

```
CLI{admin}> restricted_service_agreement enforce
Enforced: No
CLI{admin}> restricted_service_agreement enforce yes
```

25.3 Edit the Restricted Service Agreement

25.3.1 Syntax

restricted_service_agreement edit

25.3.2 Purpose

This command enables the user to edit the Restricted Service Agreement status via *vi* editor.

25.3.3 Example

```
CLI{admin}> restricted_service_agreement edit
```

26 Modbus-related commands

26.1 Discover the Modbus device

26.1.1 Syntax

discover [modbus <interface>:<address>]

26.1.2 Purpose

This command initiates the discovery process of the Modbus system. If the optional parameters are present, a directed discovery, which is much faster, is initiated. The *<interface>* and *<address>* parameters are the external interface port number and the Modbus address of the device, respectively.

26.1.3 Examples

```
CLI{admin}> discover
CLI{admin}> discover modbus 1:7
```

26.2 Show and set Modbus serial attributes

26.2.1 Syntax

modbus_parameters [<param1> <param2> <param3>]

modbus [<param1> <param2> <param3>]

modbus_parameters <index> <param>

modbus <index> <param>

modbus (tcp | <index>) <ip address>

```
<param> ::= <speed>:<parity>:<data-bits>:<stop-bits>[:ascii]
<speed> ::= integer
<parity> ::= N | O | E
<data-bits> ::= 5 | 6 | 7 | 8
<stop-bits> ::= 1 | 2
```

26.2.2 Purpose

The command without parameters shows the current Modbus serial attributes for the three physical interfaces and several TCP interfaces. The command with three parameters sets the Modbus serial attributes for the physical interfaces 1, 2 and 3. The command with two parameters sets either the Modbus serial attributes for the physical interface specified by the *<index>* (*<index>= 1,2,3*) parameter or the IP address for a Modbus TCP interface (*<index>= 8,9,10,..*). The option *tcp* is interchangeable with *8*.

The *<speed>* term specifies the baud rate (for example, 600, 19200, 38400 or 115200). The *<parity>* term specifies the parity checking. It can be *N* (none), *O* (odd), *E* (even). The *<data-bits>* and *<stop-bits>* terms specifies the number of data bits and stop bits, respectively. The *ascii* term, if specified, indicates that the ASCII version of the Modbus protocol is used.

26.2.3 Examples

```
CLI{admin}> modbus_parameters 19200:O:8:1 19200:O:8:1 9600:N:8:1:ascii
```

```
CLI{admin}> modbus_parameters
Modbus Parameters:
Interface 1: 19200:O:8:1
Interface 2: 19200:O:8:1
Interface 3: 9600:N:8:1:ascii
TCP Interface (8): 192.168.1.97
TCP Interface (9): 192.168.1.97
CLI{admin}> modbus 2 38400:O:8:1
CLI{admin}> modbus tcp 192.168.1.97
CLI{admin}> modbus 8 192.168.1.97
```

27 Revision history

27.1 Revision 1.0

Initial revision of the document for customers