TMA CLI

TMA CLI

USER and REFERENCE MANUAL

Version: 1.5 182085

Telindus Technical Publications – Geldenaaksebaan 335 · B-3001 Leuven · Belgium – tel. +32 16 382011

Copyright notice

This publication could include technical inaccuracies or typographical errors, for which Telindus never can or shall be held liable. Changes are made periodically to the information herein; these changes will be incorporated in new editions of this publication. Telindus may make improvements and/or changes in the product(s) described in this publication at any time, without prior notice.

Version 1.5 24 December 2002

MRP 182085

Preface

Organisation of this manual

This manual contains three main parts.

Part	This part …
User manual	introduces TMA CLI and explains how to install it. It also shows you the basic functions of TMA CLI.
Reference manual	details important topics about TMA CLI. It contains a complete description for lookup purposes.
Annexes	gives additional information.

The following table gives an overview of the chapters in the user manual.

Chapter	This chapter …
1	gives an introduction to TMA CLI.
2	explains how to install TMA CLI on a Windows 95 / 98 / NT / 2000, a Sun Solaris and a HP-UX system.
3	briefly describes how to connect the workstation running TMA CLI with a Telindus device.
4	shows you how to open a TMA CLI session on a Telindus device. It also says something about the TMA CLI command line prompt and environment variables.
5	introduces terms such as *containment tree*, *object*, *attribute*, *group*, etc.
6	teaches you the basics of the TMA CLI commands.
7	explains how you can user define values for attributes that have an integer as value. You can do this using the *custom.txt* file.

The following table gives an overview of the chapters in the reference manual.

Chapter	This chapter …
8	is a reference to all commands available in TMA CLI.
9	describes the communication parameters in the *Cms2Serv.ini* file.
10	covers troubleshooting including possible error messages.

The following table gives an overview of the annexes.

Annex	This annex …
Annex A	presents all error codes and their description.
Annex B	gives a list of abbreviations.
Annex C	shows ordering information.
Annex D	tells you how to obtain the required licence key.

Conventions used in this manual

Typographical conventions

The following typographical conventions are used in this manual.

The format …	is used to indicate …
Normal	normal text.
Italic	• new or emphasised words • file names and directory paths, e.g. *C:\Program Files\TMA\bin\Tma.exe*
`Computer`	computer output and code examples, e.g. `NOK,1,1,Invalid command`.
`Computer Bold`	text you have to enter at the prompt, e.g. **`Get sysName`**.
Narrow	objects and attributes in the containment tree of a device when they are mentioned in the normal text. I.e. when they are not a part of computer input or output.
Blue	references to other parts in the manual, e.g. refer to Chapter xx - Technical specifications.
Blue underlined	a hyperlink to a web site, e.g. http://www.telindus.com

Icons

The following icons are used throughout the manual.

Icon	Name	Description
ℹ	Remark	Useful information or tips.
⚠	Caution	Read the text that follows carefully in order to insure correct operation.

Continued on next page

Conventions used in this manual (continued)

Command syntax symbols

The following symbols are used for describing the syntax of TMA CLI commands:

Symbol	Name	is used to …
“ “	Double quotes	delimit composed literal strings that have to be interpreted as one string. e.g. `“Edit Configuration”`
< >	Angle brackets	delimit literal strings representing a parameter. e.g. `<object specification>`
[ ]	Square brackets	delimit optional items. e.g. `get [-r]  (recursive get)`
( )	Parentheses	group items in a simple value specification whereof you have to select only one item. e.g. `(a | b)   (can be "a" or "b")`
{ }	Curled brackets	group items in a complex value specification; select one or more items. e.g. `{a	b} (can be "", "a", "b", "ab", …)`
: : =	Production symbol	declare production rules. e.g. `<value specification> ::= <simple value specification>	<complex value specification>`
|	Disjunction symbol	combine several options from which you can choose. e.g. `<value > = <simple_value >	<struct_value>`
…	Horizontal ellipsis	indicate that some portion of the code has been omitted.

Software version

This manual describes the features of TMA CLI version S0106/01100.

Your feedback

Your satisfaction about this purchase is an extremely important priority to all of us at Telindus. Accordingly, all electronic, functional and cosmetic aspects of this new unit have been carefully and thoroughly tested and inspected. If any fault is found with this unit or should you have any other quality-related comment concerning this delivery, please submit the Quality Comment Form on our web page at http://www.telindusproducts.com/quality.

Table of contents

Continued on next page

Table of contents (continued)

Continued on next page

Table of contents (continued)

User manual

1. Introduction to TMA CLI

This chapter gives an introduction to TMA CLI. The following table gives an overview of this chapter.

1.1 What is TMA CLI?

TMA CLI is an acronym for Telindus Maintenance Application Command Line Interface. The main purpose of TMA CLI is to use its commands in scripts in order to automate management actions. This is particularly useful in large networks. TMA CLI is a complementary product to TMA and TMA for HP OpenView.

TMA CLI is available on:

- Windows 95 / 98 / NT 4.0 / 2000
- Sun Solaris 2.6 / 7
- HP-UX 10.20 / 11.0.

1.2 Which features has TMA CLI?

TMA CLI offers the following features:

- Full control over any Telindus device in your network using a command-line interface.
- Connectivity over an IP network.
- Has two operating modes:
  - interactive mode
  - non-interactive or script mode.
- Allows the management of Telindus devices exactly like the graphical version of TMA:
  - Reading and changing the configuration of a device.
  - Retrieving status information of a device, including the current alarm status.
  - Retrieving statistical information from the device.
  - Performing actions.

1.3 What is a licence key?

To enable the TMA CLI software you need to enter a licence key. This is a unique code. For more information, refer to Section 2.5 - How to obtain and install the licence key.

2. Installing TMA CLI

This chapter explains how to install TMA CLI on the Windows 95 / 98 / NT and Sun Solaris platform. First it gives you the system requirements. Read these requirements carefully to make sure your computer will be able to run TMA CLI.

The following table gives an overview of this chapter.

2.1 System requirements

TMA CLI is designed to run under Windows 95 / 98 / NT / 2000, Sun Solaris and HP-UX. The following table gives the system requirements for running TMA CLI:

	System		
	Windows 95 / 98 / NT / 2000	**Sun Solaris**	**HP-UX**
System specification	Pentium 150 or more	Sun Ultra 10 or more	HP 9000 715 or more
Operating system version	Windows 95 / 98 / NT 4.0 / 2000	Solaris 2.6 / 7	HP-UX 10.20 / 11.0
Available disk space	• 1 Mb for TMA CLI • 11 Mb for the model files [1]	• 12 Mb for TMA CLI • 9 Mb for the model files [1]	• 12 Mb for TMA CLI • 9 Mb for the model files [1]
Recommended RAM	• 16 MB for Windows 95/98 • 32 MB for Windows NT 4.0 • 64 MB for Windows 2000	64 Mb or more	64 Mb or more
Communication port	• serial COM port: 9600, 8+N and / or • Ethernet card	Ethernet card	Ethernet card
Networking	TCP/IP networking installed and configured	TCP/IP networking installed and configured	TCP/IP networking installed and configured
Additional hardware	CD-ROM drive	CD-ROM drive	CD-ROM drive

[1] Because the model files delivered with TMA CLI evolve as the devices evolve, their size tends to increase with each new release. The figures specified in the table above, is the size of TMA CLI and the model files as it was when this manual was written. For future releases of TMA CLI, these figures may be incorrect.

2.2 The three installation components of TMA CLI

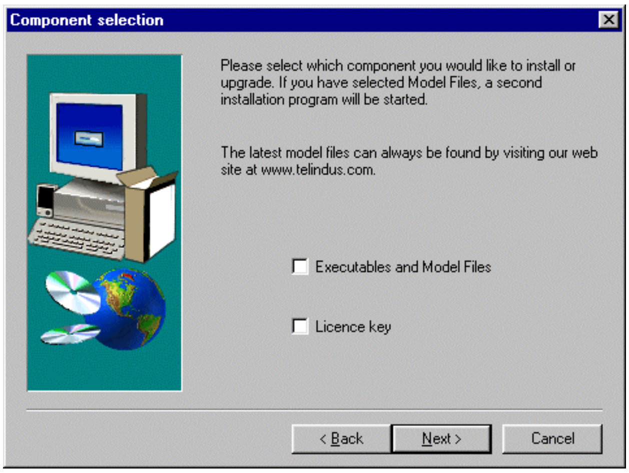
The installation of TMA CLI comprises three components as listed below:

Component	Description
TMA CLI executable	Provides the TMA command line interface.
Model files	Provide, per access device type, the information TMA CLI needs to connect and exchange information with the access device.
Licence key	Enables the use of TMA CLI.

2.3 Installing TMA CLI on Windows 95 / 98 / NT / 2000

This section explains how to install TMA CLI on the Windows 95 / 98 / NT / 2000 platform.

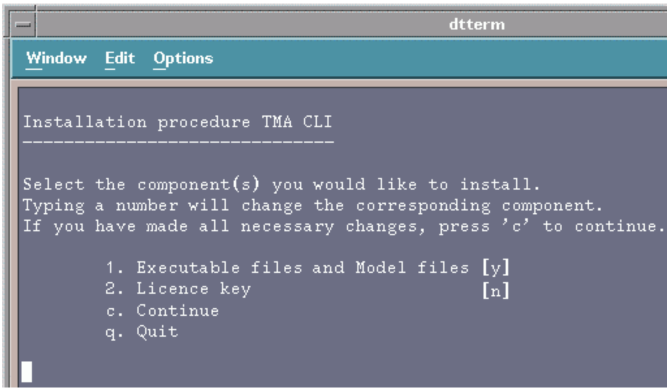
The installation files of TMA CLI are located on the CD-ROM of TMA for HP OpenView. In order to install TMA CLI on a Windows 95 / 98 / NT / 2000 system, proceed as follows:

Step	Action
1	Insert the installation CD in the CD-ROM drive.
2	Windows automatically starts the set-up procedure for TMA for HP OpenView. Press the *Cancel* button.
3	Run the following set-up program instead: *D:\NT\TmaCli\Setup.exe*. Possibly you have to replace ‘*D*’ by another letter you use for your CD-ROM drive.
4	The InstallShield Wizard guides you through the set-up process.
5	At a certain point, you get the *Component selection* screen: At this point you can install … • the TMA executable and the model files and / or • the licence key.  Component selection Please select which component you would like to install or upgrade. If you have selected Model Files, a second installation program will be started. The latest model files can always be found by visiting our web site at www.telindus.com. □ Executables and Model Files □ Licence key
6	The InstallShield Wizard guides you through the rest of the set-up process.

2.4 Installing TMA CLI on Sun Solaris and HP-UX

This section explains how to install TMA CLI on the Sun Solaris or HP-UX platform.

The installation files of TMA CLI are located on the CD-ROM of TMA for HP OpenView. In order to install TMA CLI on a Sun Solaris or HP-UX system, proceed as follows:

Step	Action
1	Insert the installation CD in the CD-ROM drive.
2	Start the *install* script from the following directory on the CD-ROM: • *SOL/TmaCli* in case of Sun Solaris • *HPUX/TmaCli* in case of HP-UX.
3	The script guides you through the set-up process.
4	At a certain point, you get the following screen: At this point you can install … • the TMA executable and the model files and / or • the licence key.  Installation procedure TMA CLI ------------------------------ Select the component(s) you would like to install. Typing a number will change the corresponding component. If you have made all necessary changes, press 'c' to continue. 1. Executable files and Model files [y] 2. Licence key [n] c. Continue q. Quit
5	The script guides you through the rest of the set-up process.

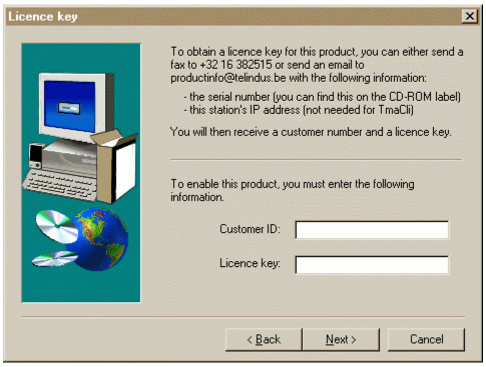
2.5 How to obtain and install the licence key

To enable the TMA CLI software you need to enter a licence key. The required licence key can be obtained by sending a fax or an email to Telindus (Refer to Annex D: licence key request).

To install the licence key, proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Rerun the installation as described in …
• Section 2.3 - Installing TMA CLI on Windows 95 / 98 / NT / 2000
or
• Section 2.4 - Installing TMA CLI on Sun Solaris and HP-UX.</td></tr>
<tr><td>2</td><td>When you are prompted to select which component you want to install, only select the licence key component.

For Windows, the following *Licence key* window appears:



To obtain a licence key for this product, you can either send a fax to +32 16 382515 or send an email to productinfo@telindus.be with the following information:

- the serial number (you can find this on the CD-ROM label)
- this station's IP address (not needed for TmaCli)

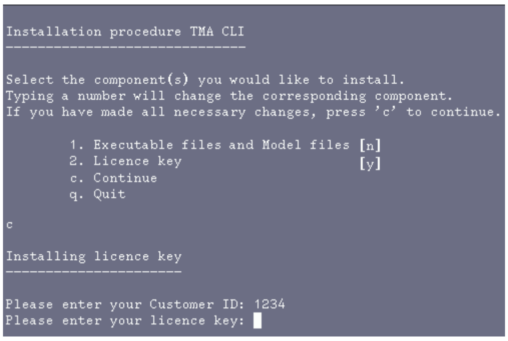
You will then receive a customer number and a licence key.

To enable this product, you must enter the following information.

Customer ID:

Licence key:

For Sun Solaris and HP-UX, the following *Licence key* window appears:



```
Installation procedure TMA CLI
------------------------------

Select the component(s) you would like to install.
Typing a number will change the corresponding component.
If you have made all necessary changes, press ’c’ to continue.

        1. Executable files and Model files [n]
        2. Licence key                      [y]
        c. Continue
        q. Quit

c

Installing licence key
----------------------

Please enter your Customer ID: 1234
Please enter your licence key: █
```
</td></tr>
<tr><td>3</td><td>Enter the customer ID which you received after you submitted your licence key request.</td></tr>
<tr><td>4</td><td>Enter the licence key which you received after you submitted your licence key request.</td></tr>
</table>

2.6 How to upgrade the model files

The installation of the model files may occur separately from the TMA CLI executable installation. This because the model files evolve as the devices evolve. The model files delivered on the CD-ROM correspond to the latest firmware version of the Telindus access devices at the creation time of the CD-ROM. If, at a later time, you want to add devices with a more recent firmware version, a re-installation of the model files of these devices may be required.

Check the most recent model files on the Telindus web site.

To download and install the most recent model files, proceed as follows:

Step	Action
1	Go to the Telindus web site at http://www.telindusproducts.com and select *Products* → *Maintenance & Management* → *TMA for HP OpenView* → *Download model files upgrade*.
2	Carefully read the licence agreement for Telindus software. • If you agree with the terms stated in the agreement, then select *I agree* and continue with step 3. • If you do not agree with the terms stated in the agreement, then select *I don't agree*.

3	**Windows**
	For TMA CLI on Windows, select the following line:
	`TMA part 2: data files rev. xxx [xxxx Kbyte]`
	Save the executable file (e.g. *S0011015.exe*) in a temporary directory on your hard disk.
	UNIX
	For TMA CLI on UNIX, select the following line:
	`TMA part 2: data files UNIX rev. xxx [xxxx Kbyte]`
	Save the tar file *xxx.tar.Z* in a temporary directory on your hard disk.

4	**Windows** For TMA CLI on Windows, double click on the executable file. Follow the instructions that appear on your screen. **UNIX** For TMA CLI on UNIX, do the following: 1. If you downloaded the TAR file on a Windows station, the file name suffix has been changed by the browser into *_tar.Z*. Transfer the file to a temporary directory on your UNIX workstation and change the file extension again into *.tar.Z*. 2. Uncompress the file using the command `uncompress xxx.tar.Z`. 3. Untar the resulting file *xxx.tar* using the command `tar -xvf xxx.tar`. 4. Execute the install script from the temporary directory.

The device firmware is backwards compatible with the model files. This means that the latest TMA model files support not only the latest firmware version, but also all previous firmware versions of the device.

2.7 Location of the TMA CLI files

If you did not change the default file location during the set-up, then the TMA CLI related directories and files can be found in the directory *Program Files\TMA* (Windows) or *\opt\TMA* (UNIX). This directory has the following subdirectories:

Directory	This directory contains …
\bin	the executables. This comprises the TMA CLI executable and some extra executables TMA sometimes summons (such as TML, TmaTftp, …).
\config	the configuration files. These are the **.ini* files. If you create a *custom.txt* file, you should also place it here.
\log	the log files. Sometimes errors are logged to a file. These files can be found in this directory.
\model	the model files. TMA needs the model files to be able to communicate with the Telindus devices. These **.mod* files are located in this directory.
\picture	the subsystem picture files. These are the **.bmp* and **.def* files that are necessary to display the subsystem picture of a Telindus device.
\snmp	the Telindus MIB files. When using an SNMP browser, you need the MIB files of the Telindus devices. These **.mib* files are located in this directory.
\snmp_info	the SUM files. The *.sum files give the relationship between the attributes as they are displayed in TMA and the parameters you can see using an SNMP browser.

3. Connecting to a device

Once the TMA CLI application and the model files are installed, you are ready to interconnect the computer running TMA CLI and a Telindus device. This is explained in this chapter. First the terms *IP device*, *proxied IP device* and *non-IP device* are explained, for they are used in this and the following chapters.

The following table gives an overview of this chapter.

3.1 What are IP, proxied IP and non-IP devices?

Because in this and the following chapters the terms *IP device*, *proxied IP device* and *non-IP device* are often used, they are explained in this section.

The following table gives a definition of each term together with an example:

Term	Definition
IP device	An IP device is a Telindus access device … • in which you can configure an IP address. • that has a dedicated LAN port through which you can connect the device to a LAN. **Example** The Crocus Inverse Multiplexer is an IP device. You can configure an IP address in the Crocus Inverse Multiplexer using the crocusInvMux/lanInterface/ipAddress attribute. The Crocus Inverse Multiplexer can be connected to a LAN through its TPI port located at the back of the device. Other IP devices are for instance: Orchid 1003 LAN, Crocus Router Interface, Crocus Router 2M, Telindus 1421 SHDSL Router.
non-IP device	A non-IP device is a Telindus access device … • in which you can not configure an IP address. • has no dedicated LAN port and therefore can not be connected directly to a LAN. **Example** The Crocus SDSL F baseband modem is a non-IP device. You can not configure an IP address in the Crocus SDSL F and you can not connect it to a LAN through a dedicated LAN port. Other non-IP devices are for instance: Aster 4 F, Crocus HDSL F, Crocus SDSL F, Crocus FO10M.
proxied IP device	A proxied IP device is actually a non-IP device. This means it is a Telindus access device … • in which you can not configure an IP address. However, you can assign an IP address to the device using a management concentrator as *proxy IP device* (refer to Section 3.4 - Proxied IP connection to a non-IP device). • has no dedicated LAN port and therefore can not be connected directly to a LAN. This is done through a management concentrator which is an IP device and therefore can be connected to a LAN through its dedicated LAN port. **Example** The Crocus SDSL F baseband modem is a non-IP device. However, by connecting the Orchid 1003 LAN to the modem and by assigning an IP address to it in the Orchid 1003 LAN, the modem becomes a proxied IP device. The Orchid 1003 LAN on its turn, is connected through its LAN port to a LAN. I.e. it is as if the modem is connected to the LAN, although not directly.

3.2 Direct connection to a device

This section explains how to make a direct connection between the computer running TMA CLI and a Telindus device.

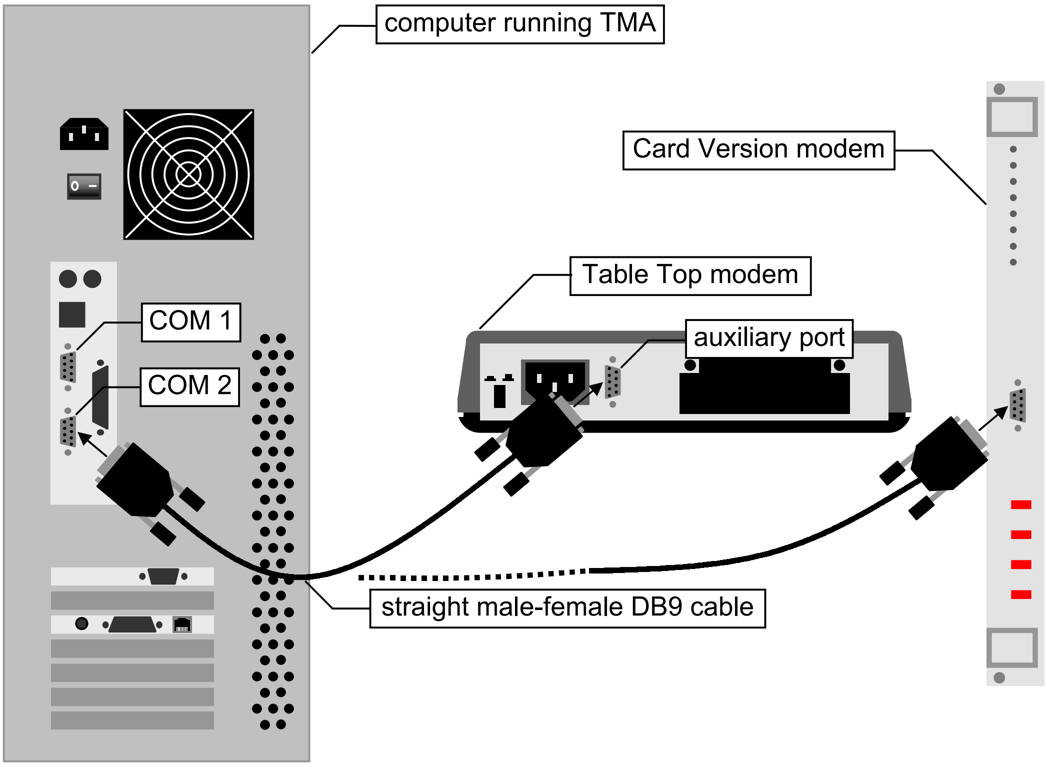
The following table gives an overview of this section.

3.2.1 What is a direct connection?

A direct connection is a connection between a COM port of the computer and the auxiliary port (also called control port) of a Telindus device. Such a connection is made by means of a straight male-female DB9 cable.

There are some exceptions. The Orchid 1003 LAN, for instance. This device has an RJ45 control port. Consequently, the connection is made by means of a DB9 - RJ45 cable. This cable is delivered with the Orchid 1003 LAN.

The following figure shows an example of a direct connection to a Table Top and Card Version modem:



Once the connection is made then the computer running TMA CLI is able to reach the modem.

3.2.2 To which Telindus devices can you make a direct connection?

Every Telindus device that is manageable with TMA has a control port. Hence, you can make a direct connection to any of these devices, regardless of the fact it is a non-IP, IP or proxied IP device.

3.2.3 DB25 – DB9 interconnection cable

If the COM port you want to use has a DB25 connector, then a different cable has to be used. Such a cable has the following layout:

Female DB25 connector for connection towards the computer			**Male DB9 connector for connection towards the Telindus device**		
Pin	Signal	Input / output	Pin	Signal	Input / output
2	TXD	output	3	TXD	input
3	RXD	input	2	RXD	output
4	RTS	output	7	RTS	input
5	CTS	input	8	CTS	output
6	DSR	input	6	DSR	output
7	GND	-	5	GND	-
20	DTR	output	4	DTR	input

3.3 IP connection to an IP device

This section explains how to make an IP connection between the computer running TMA CLI and a Telindus IP device.

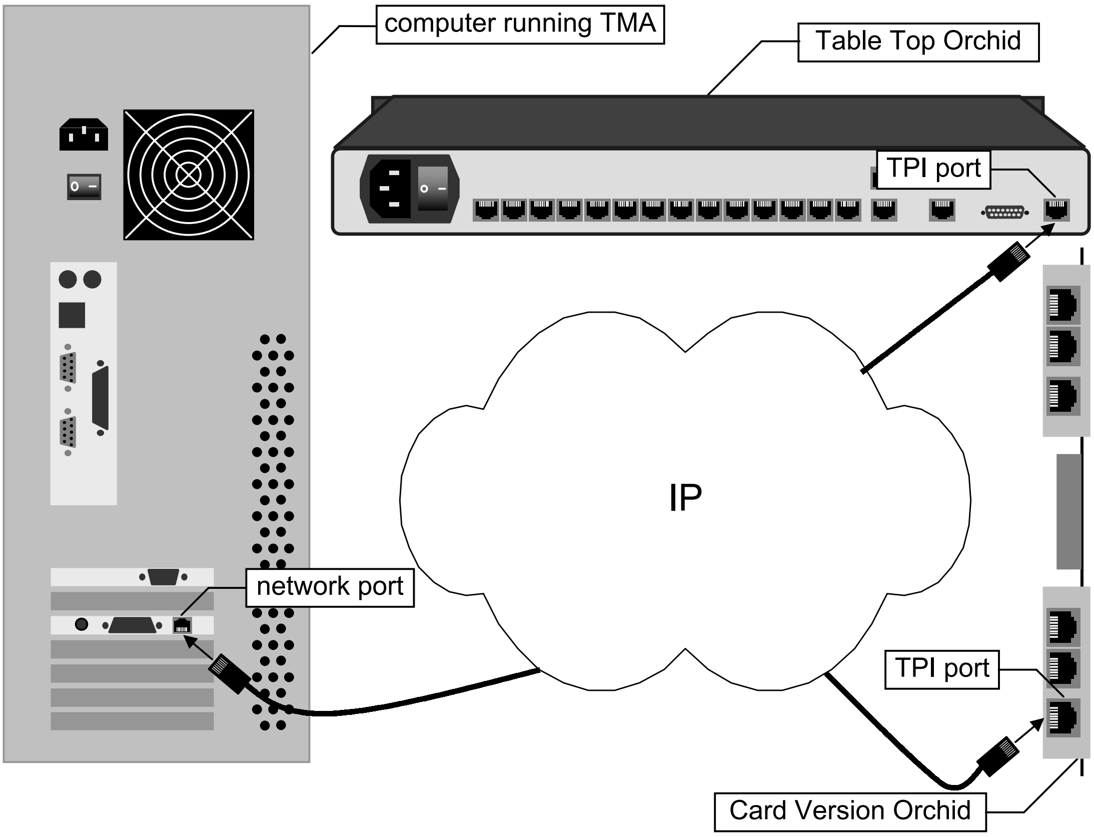
The following table gives an overview of this section.

3.3.1 What is an IP connection?

A connection via IP is a connection between:

- the LAN port of the IP device and the IP network at one side
- the network port of the computer and the IP network at the other side.

The following figure shows an example of a connection via IP to an Orchid 1003 LAN Table Top and Card Version:



As opposed to a direct connection, making the physical connection alone is not sufficient to establish an IP connection between the computer and the IP device. A few basic settings have to be made in the IP device.

3.3.2 To which Telindus devices can you make an IP connection?

You can make an IP connection to every Telindus device that has a dedicated LAN port through which you can connect the device to a LAN. Hence, you can only make a connection via IP to a Telindus IP device, not to a non-IP device.

There is a way to make an IP connection to a non-IP device. However, this involves using a management concentrator with a particular configuration. In that case, the non-IP device becomes a proxied IP device. For more information, refer to Section 3.4 - Proxied IP connection to a non-IP device.

3.3.3 Basic IP device settings to enable an IP connection

To establish an IP connection between the computer running TMA and the IP device, a few basic parameters (called attributes) have to be set in the IP device. These attributes are:

Attribute	Description
IP address	This is a unique address which is assigned to the IP device. By doing this, other devices on the IP network can contact the IP device. **Examples** The following examples display the IP address attribute location in the containment tree of some Telindus IP devices: • Orchid 1003 LAN: o1003/interfaces/lanInterface/ipAddress • Crocus Router 2M or Router Interface: crocusRouter/lanInterface/ipAddress • Crocus Inverse Multiplexer: crocusInvMux/lanInterface/ipAddress
default gateway	This is the gateway of the IP segment the IP device is connected to. I.e. the IP address of the router which handles packets destined for another network. **Examples** The following examples display the default gateway attribute location in the containment tree of some Telindus IP devices: • Orchid 1003 LAN: o1003/router/defaultRoute/gateway • Crocus Router 2M or Router Interface: crocusRouter/router/defaultRoute/gateway • Crocus Inverse Multiplexer: crocusInvMux/lanInterface/defaultRoute

For more detailed information, refer to the manual of the IP device.

Once these attributes are set then the computer running TMA CLI is able to reach the IP device over an IP network.

3.4 Proxied IP connection to a non-IP device

As stated in Section 3.3 - IP connection to an IP device, it is not possible to make a *true* IP connection to a non-IP device. This because non-IP devices do not have a dedicated LAN port through which you can connect them to a LAN. However, using a management concentrator, you can make a *proxied* IP connection to a non-IP device. This is explained in this section.

The following table gives an overview of this section.

3.4.1 What is a management concentrator?

A management concentrator is a device that collects management information from the network units (i.e. the Telindus devices in the network) and passes it to a network management system (e.g. HP OpenView). So a management concentrator is situated between the network units and the network management system. In other words, it is a key element for centralised network management.

3.4.2 Which Telindus devices are management concentrators?

Examples of management concentrator are:

- the Orchid 1003 LAN (the predecessor of the Telindus 1035 Orchid).
- the Telindus 1031, 1032, 1033 and 1034 Router (note that their management concentrator capabilities are limited due to hardware limitations).
- the Telindus 1035 Orchid (the successor of the Orchid 1003 LAN).

For more detailed information on these devices, refer to their user manuals.

3.4.3 A management concentrator as a proxy IP device

You can connect a Telindus non-IP device (e.g. a Crocus modem) to a management concentrator. In the management concentrator configuration, you can assign an IP address to this non-IP device. In that case, the *non-IP device* becomes a *proxied IP device*. I.e. you can now access the non-IP device over an IP network through the management concentrator. In other words, the management concentrator acts as *proxy IP device* for the non-IP device.

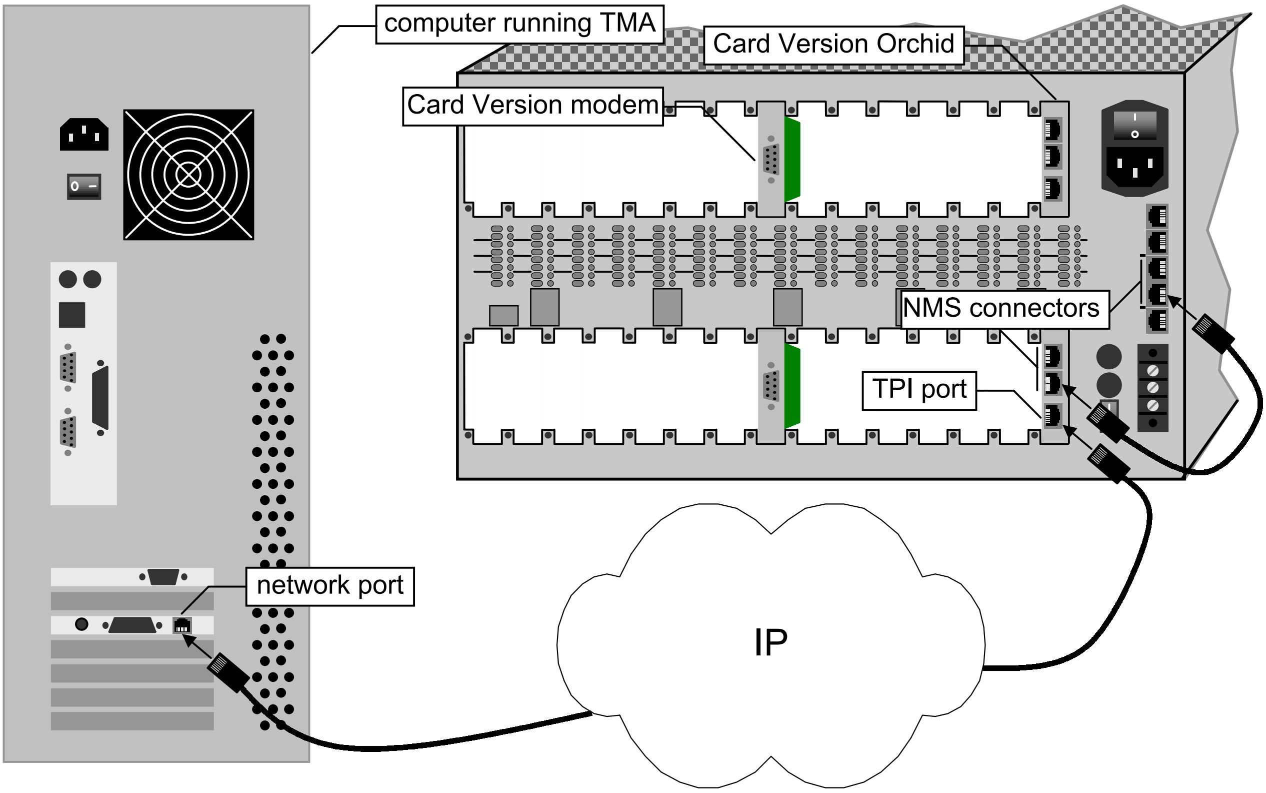
This implies that as opposed to the management concentrator, the non-IP device is not directly connected through a dedicated LAN port to the IP network. An example of an IP connection between the computer running TMA CLI and a non-IP device is given in the following paragraph.

3.4.4 A proxied IP connection to a non-IP device – example

Consider the following components which have to be interconnected:

- the computer running TMA CLI
- the IP network
- a CN4 card nest
- a Card Version modem
- a Card Version Orchid 1003 LAN.

The following figure shows the interconnection between these components:



The following table gives an overview of the connections shown in the figure above. The overview starts at the computer running TMA CLI.

The …	is connected to the …	via …
network port of the computer	IP network	an RJ45 network cable.
TPI port of the Orchid 1003 LAN	IP network	an RJ45 network cable.
NMS port of the Orchid 1003 LAN	NMS port of the CN4 card nest	a straight RJ45 cable.
NMS port of the CN4 card nest	high speed NMS bus of the Card Version modem	the high speed NMS bus on the backplane of the CN4 card nest.

As opposed to a direct connection, making the physical connection alone is not sufficient to establish a proxied IP connection between the computer and the non-IP device. A few basic settings have to be made in the Orchid 1003 LAN. These are shown in the following paragraph.

3.4.5 Basic management concentrator settings to enable a proxied IP connection

To establish an IP connection between the computer running TMA and the non-IP device, a few basic parameters (called attributes) have to be set in the management concentrator:

- The first two attributes are the IP address and default gateway of the management concentrator itself. These parameters are already explained in Section 3.3 - IP connection to an IP device.
- The other attribute is the object table of the management concentrator: nmsgroup/objectTable. The relevant elements in this table are:

Attribute	Description
ipAddress	Use this attribute to assign an IP address to the non-IP device. The IP address should belong to the sub-network of the management concentrator. For example, if the IP address of the management concentrator is 192.168.4.5 with subnetmask 255.255.255.0 then the IP address of the network unit should also be within the range from 192.168.4.1 to 192.168.4.254.
addressType	The management concentrator has to know how it can contact the connected non-IP device. Therefore, use the addressType attribute to specify the type of address: • a relative address • an absolute address. For more information on these address types, refer to Section 4.2.6 - Connecting using absolute addressing.
addressValue	If you set the addressType attribute to absolute, then use the addressValue attribute to specify the absolute address value.

Continued on next page

nmsgroup/objectTable (continued)

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>port
or
exitPort</td><td>The exit port specifies through which port of the management concentrator the network unit can be reached.
In case of the exitPort attribute, the underlying attributes are:
<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>exitPortType</td><td>Use this attribute to specify whether the non-IP device is reachable through …<ul><li>one of the asynchronous ports of the management concentrator (port). In this case, also set the attribute portNumber.</li><li>the high speed bus of the CN4 Card Nest (rack). In this case, also set the attributes cardnestAddress, cardPosition and modem.</li></ul></td></tr>
<tr><td>portNumber</td><td>Use this attribute to specify through which asynchronous port of the management concentrator the non-IP device can be reached.</td></tr>
<tr><td>cardnestAddress</td><td>Use this attribute to specify the CN4 Card Nest address in which the non-IP device (Card Version) resides.</td></tr>
<tr><td>cardPosition</td><td>Use this attribute to specify the position of the non-IP device (Card Version) in the CN4 Card Nest.</td></tr>
<tr><td>modem</td><td>Use this attribute to select the device on the non-IP multi-device (Card Version Twin, Quad, etc.): A, B, C or D.</td></tr>
</table>
</td></tr>
</table>

For more detailed information, refer to the manual of the management concentrator.

Once these attributes are set then the computer running TMA CLI is able to reach the non-IP device over an IP network.

4. Opening a TMA CLI session

Once the physical connections are made, you are able to open a TMA CLI session on a Telindus device. First this chapter explains how to display the help on the TMA CLI connection possibilities. Then it describes the different possibilities.

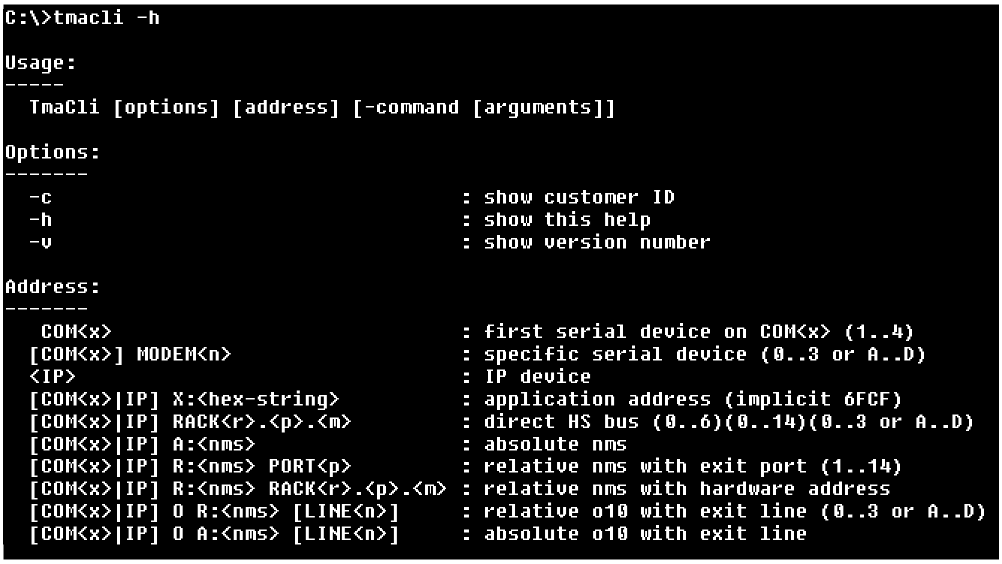
The following table gives an overview of this chapter.

4.1 Displaying help on TMA CLI connection possibilities

To display a condensed version of the TMA CLI connection possibilities, proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Open a shell program. E.g. MS-DOS in the Windows operating system.</td></tr>
<tr><td>2</td><td>At the prompt, type <b>TmaCli</b> or <b>TmaCli -h</b>.</td></tr>
<tr><td>3</td><td>

The following is displayed:



```
C:\>tmacli -h

Usage:
-----
  TmaCli [options] [address] [-command [arguments]]

Options:
-------
  -c                                  : show customer ID
  -h                                  : show this help
  -v                                  : show version number

Address:
-------
   COM<x>                             : first serial device on COM<x> (1..4)
  [COM<x>] MODEM<n>                   : specific serial device (0..3 or A..D)
  <IP>                                : IP device
  [COM<x>|IP] X:<hex-string>          : application address (implicit 6FCF)
  [COM<x>|IP] RACK<r>.<p>.<m>         : direct HS bus (0..6)(0..14)(0..3 or A..D)
  [COM<x>|IP] A:<nms>                 : absolute nms
  [COM<x>|IP] R:<nms> PORT<p>         : relative nms with exit port (1..14)
  [COM<x>|IP] R:<nms> RACK<r>.<p>.<m> : relative nms with hardware address
  [COM<x>|IP] O R:<nms> [LINE<n>]     : relative o10 with exit line (0..3 or A..D)
  [COM<x>|IP] O A:<nms> [LINE<n>]     : absolute o10 with exit line
```

All the different connection options are explained in the following section.

</td></tr>
</table>

4.2 The TMA CLI connection possibilities

The following table lists once more the different connection possibilities as they are displayed in the TMA CLI help. All the different possibilities are explained in the following sections.

Connection possibility	This connection possibility is explained in Section …
`COM<x>`	4.2.1 - Connecting direct, page 32
`[COM<x>] MODEM<n>`	4.2.2 - Connecting direct and selecting a modem, page 33
`<IP>`	4.2.3 - Connecting over IP, page 34
`[COM<x>|IP] X:<hex-string>`	4.2.4 - Connecting to an application, page 35
`[COM<x>|IP] RACK<r>.<p>.<m>`	4.2.5 - Connecting to a non-configured device, page 36
`[COM<x>|IP] A:<nms>`	4.2.6 - Connecting using absolute addressing, page 37
`[COM<x>|IP] R:<nms> PORT<p>`	4.2.7 - Connecting using relative addressing and an exit port, page 38
`[COM<x>|IP] R:<nms> RACK<r>.<p>.<m>`	4.2.8 - Connecting using relative addressing and a hardware address, page 39
`[COM<x>|IP] O R:<nms> [LINE<n>]`	4.2.9 - Connecting using relative addressing and an exit line, page 41
`[COM<x>|IP] O A:<nms> [LINE<n>]`	4.2.10 - Connecting using absolute addressing and an exit line, page 42

If you have TMA for HP OpenView, you also have the possibility to use the name resolution feature of TMA for HP OpenView to open a TMA CLI session on a Telindus device. For more information, refer to Section 4.4 - Connecting using name resolution.

4.2.1 Connecting direct

```
TmaCli COM<x>
```

Use this command if you want to make a direct (also called serial) connection as described in Section 3.2 - Direct connection to a device. In that case you have to specify which COM port of the computer is connected to the control port of the Telindus device.

When making a direct connection to a Telindus device that is being managed by a management concentrator, you might experience connection problems. In this case, it is best to make a connection through the management concentrator (i.e. the management concentrator as proxy).

Address argument

Replace `COM<x>` by one of the four possibilities listed below:

- `COM1`
- `COM2`
- `COM3`
- `COM4`

4.2.2 Connecting direct and selecting a modem

```
TmaCli [COM<x>] MODEM<n>
```

Some Telindus devices incorporate several devices on one card. This is called a *multi-device*. The different devices on a multi-device are referred to using A, B, C, D, etc.

Examples of multi-devices are:

- Crocus SHDSL CV Twin: incorporates 2 SHDSL modems on one card.
- Crocus 2M CNV CV Twin: incorporates 2 interface converters on one card.
- Crocus SDSL CV Quad: incorporates 4 SDSL modems on one card.

If you connect to such a multi-device, you can specify on which device (A, B, C, D, etc.) you want open the TMA CLI session.

When making a direct connection to a Telindus device that is being managed by a management concentrator, you might experience connection problems. In this case, it is best to make a connection through the management concentrator (i.e. the management concentrator as proxy).

Address arguments

Replace `COM<x>` by one of the four possibilities listed below:

- `COM1`
- `COM2`
- `COM3`
- `COM4`

If you do not specify a COM port, COM1 is taken as default.

Replace `MODEM<n>` by one of the four possibilities listed below:

- `MODEMA` or `MODEM0`
- `MODEMB` or `MODEM1`
- `MODEMC` or `MODEM2`
- `MODEMD` or `MODEM3`

Examples

If you want to connect to modem A of a Crocus FO10M CV Twin connected to COM port 1, then type:

- **`TmaCli com1 modemA`**

or

- **`TmaCli com1 modem0`**

or

- **`TmaCli modemA`**

or

- **`TmaCli modem0`**

If you want to connect to modem B of a Crocus HDSL CV Twin connected to COM port 4, then type:

- **`TmaCli com4 modemB`**

or

- **`TmaCli com4 modem1`**

4.2.3 Connecting over IP

```
TmaCli <IP>
```

Use this command if you want to make a connection over an IP network as described in Section 3.3 - IP connection to an IP device and Section 3.4 - Proxied IP connection to a non-IP device.

Address argument

Replace `<IP>` by the IP address of the Telindus device you want to connect to.

Example

If you want to connect to a device that has IP address 10.0.11.1, then type:

```
TmaCli 10.0.11.1
```

4.2.4 Connecting to an application

```
TmaCli [COM<x>|IP] X:<hex-string>
```

Use this command if you want to connect to an application. This feature allows you to open a TMA CLI session on, for example, the Alarm Manager of TMA for HP OpenView on a machine that is not running HP OpenView.

Address arguments

In case the connection between your computer running TMA CLI and the machine running the application you want to connect to is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the machine: `<IP>`.

Replace `X:<hex-string>` by the address of the application you want to connect to. In case of the Alarm Manager for example, the address is `6fbf`.

Example

If you want to connect to the Alarm Manager of TMA for HP OpenView which is running on a machine with IP address 10.0.11.1, then type:

```
TmaCli 10.0.11.1 X:6fbf
```

4.2.5 Connecting to a non-configured device

```
TmaCli [COM<x>|IP] RACK<r>.<p>.<m>
```

Use this command if you want to connect to a device (Card Version) that is inserted in a card nest CN4 which is being managed by a management concentrator, but is not (yet) configured in the object table of the management concentrator. Using this command you can nevertheless open a TMA CLI session on the non-configured device over the high-speed bus of the card nest.

This command only works …

- if the management concentrator supports the *connect to non-configured device* feature.
- if the device you want to connect to is a CMS2 device. (However, although the Aster 4 Flash is a CMS2 device, it does not support the *connect to non-configured device* feature.)

For more information on the object table, refer to Section 3.4 - Proxied IP connection to a non-IP device and the manual of the management concentrator.

Address arguments

In case the connection between your computer running TMA CLI and the management concentrator you want to connect through is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the management concentrator: `<IP>`.

Replace `RACK<r>.<p>.<m>` by the hardware address of the non-configured device. The hardware address consists of three fields. The following table explains what they mean:

Field	Field name	Description
r	rack address	The rack address can be set by means of DIP switches located at the back of the card nest. The address range goes from 0 to 6. Check the rack address of the card nest containing the Card Version modem you want to address. Enter it in the first field. For example: 2.
p	card position	Also the position of the card in the card nest has to be known for addressing purposes. The card slot range goes from 0 up to 14. Check the position of the card in the card nest. Enter it in the second field. For example: 12.
m	modem	If you want to address a certain device on a multi-device, you have to specify which device you want to address: device A, B, C or D. Verify which device you want to address. Enter it in the third field. For example: B.

Example

Suppose you want to connect to modem B of a Crocus FO10M CV Twin that is inserted on position 5 of a card nest CN4. The card nest has card nest address 3 and is under control of an Orchid 1003 LAN that has IP address 10.0.11.1. In that case, type the following:

```
TmaCli 10.0.11.1 rack3.5.B
```

4.2.6 Connecting using absolute addressing

```
TmaCli [COM<x>|IP] A:<nms>
```

Use this command to connect through a management concentrator to a device that has an absolute address.

For more information on relative and absolute addressing, refer to the Section 4.3 - Relative and absolute addressing.

Address arguments

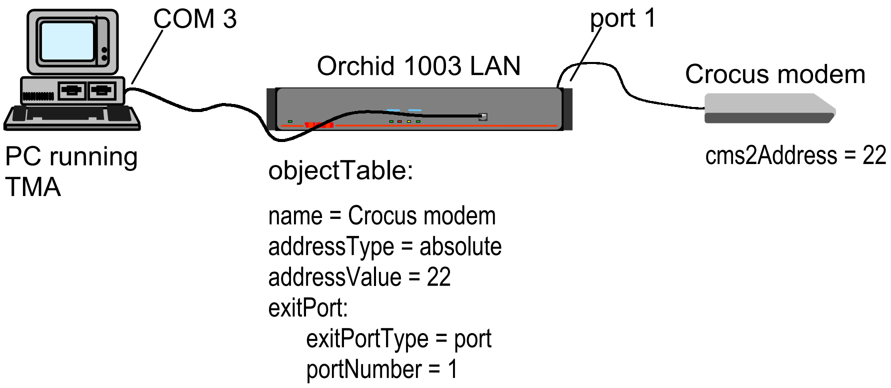
In case the connection between your computer running TMA CLI and the management concentrator you want to connect through is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the management concentrator: `<IP>`.

Replace `A:<nms>` by the absolute address as configured in the device and in the object table of the management concentrator.

Example

Suppose you have the following set-up:



objectTable:

name = Crocus modem
addressType = absolute
addressValue = 22
exitPort:
    exitPortType = port
    portNumber = 1

In order to open a TMA CLI session on the Crocus modem, type the following:

```
TmaCli com3 A:22
```

4.2.7 Connecting using relative addressing and an exit port

```
TmaCli [COM<x>|IP] R:<nms> PORT<p>
```

Use this command to connect to a device through one of the asynchronous ports of the management concentrator. The addressing method that is used is relative addressing.

For more information on relative and absolute addressing, refer to the Section 4.3 - Relative and absolute addressing.

Address arguments

In case the connection between your computer running TMA CLI and the management concentrator you want to connect through is …

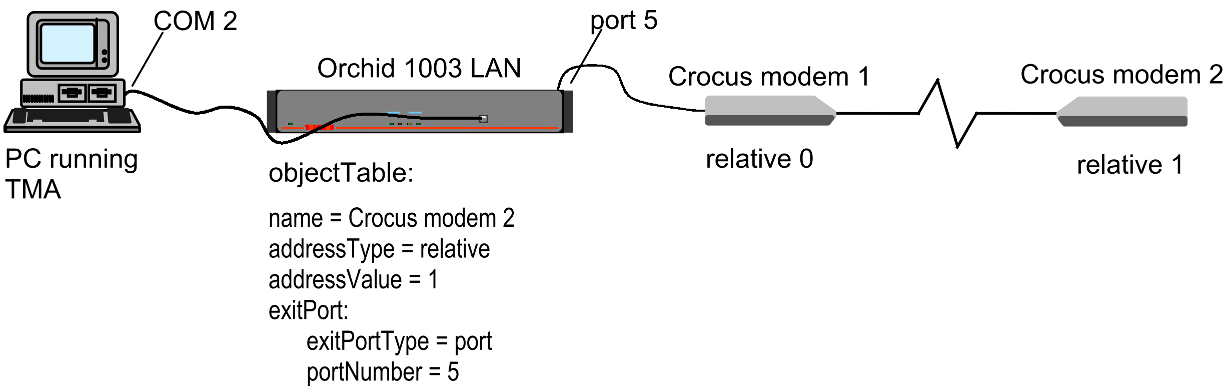
- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the management concentrator: `<IP>`.

Replace `R:<nms>` by the relative address of the device.

Replace `PORT<p>` by the asynchronous port number to which the device is connected.

Example

Suppose you have the following set-up:



name = Crocus modem 2
addressType = relative
addressValue = 1
exitPort:
    exitPortType = port
    portNumber = 5

In order to open a TMA CLI session on the Crocus modem 2, type the following:

```
TmaCli com2 r:1 port5
```

4.2.8 Connecting using relative addressing and a hardware address

```
TmaCli [COM<x>|IP] R:<nms> RACK<r>.<p>.<m>
```

Use this command to connect to a device through the high speed bus of a card nest CN4 under management of a management concentrator. The addressing method that is used is relative addressing.

For more information on relative and absolute addressing, refer to the Section 4.3 - Relative and absolute addressing.

Address arguments

In case the connection between your computer running TMA CLI and the management concentrator you want to connect through is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the management concentrator: `<IP>`.

Replace `R:<nms>` by the relative address of the device.

Replace `RACK<r>.<p>.<m>` by the hardware address of the device. The hardware address consists of three fields. The following table explains what they mean:

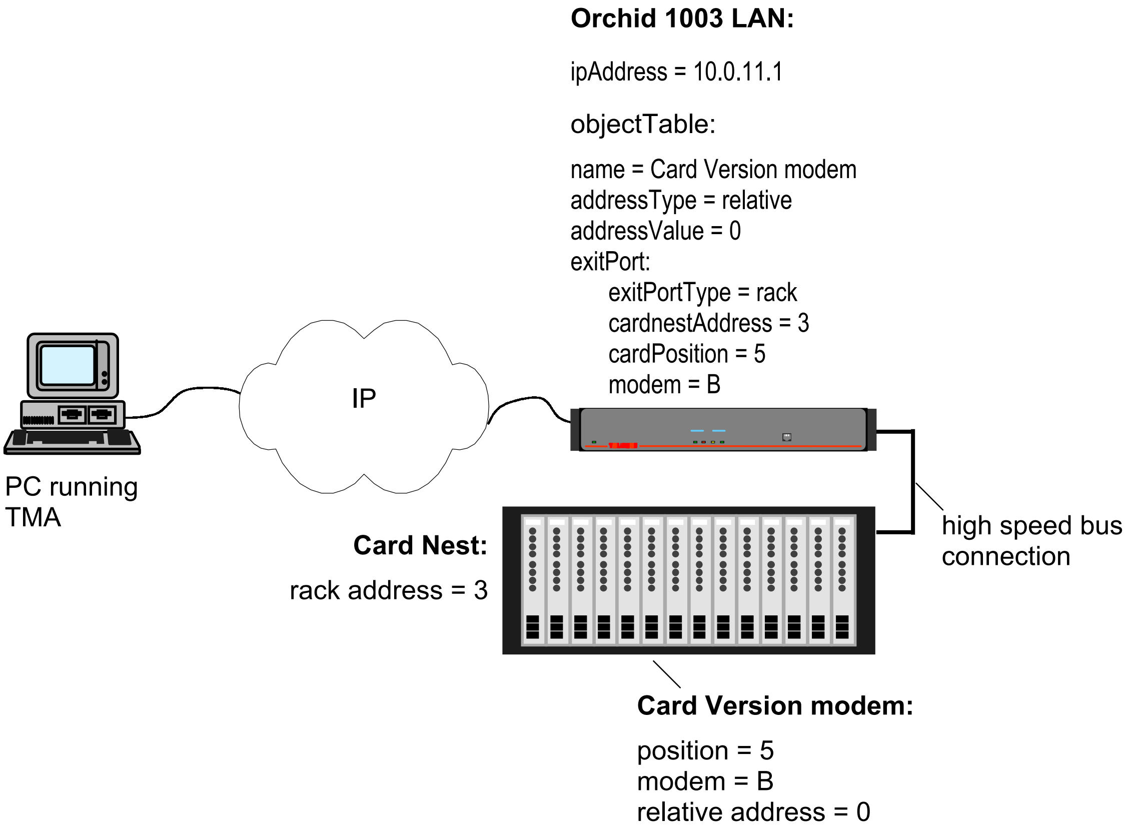
Field	Field name	Description
r	rack address	The rack address can be set by means of DIP switches located at the back of the card nest. The address range goes from 0 to 6. Check the rack address of the card nest containing the Card Version modem you want to address. Enter it in the first field. For example: 2.
p	card position	Also the position of the card in the card nest has to be known for addressing purposes. The card slot range goes from 0 up to 14. Check the position of the card in the card nest. Enter it in the second field. For example: 12.
m	modem	If you want to address a certain device on a multi-device, you have to specify which device you want to address: device A, B, C or D. Verify which device you want to address. Enter it in the third field. For example: B.

Continued on next page

Connecting using relative addressing and a hardware address (continued)

Example

Suppose you have the following set-up:



In order to open a TMA CLI session on the Card Version modem, type the following:

```
TmaCli 10.0.11.1 r:0 rack3.5.B
```

4.2.9 Connecting using relative addressing and an exit line

```
TmaCli [COM<x>|IP] O R:<nms> [LINE<n>]
```

Use this command to connect to a device located behind another device. The addressing method that is used is relative addressing.

For more information on relative and absolute addressing, refer to the Section 4.3 - Relative and absolute addressing.

Address arguments

In case the connection between your computer running TMA CLI and the first device you want to connect through is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the first device: `<IP>`.

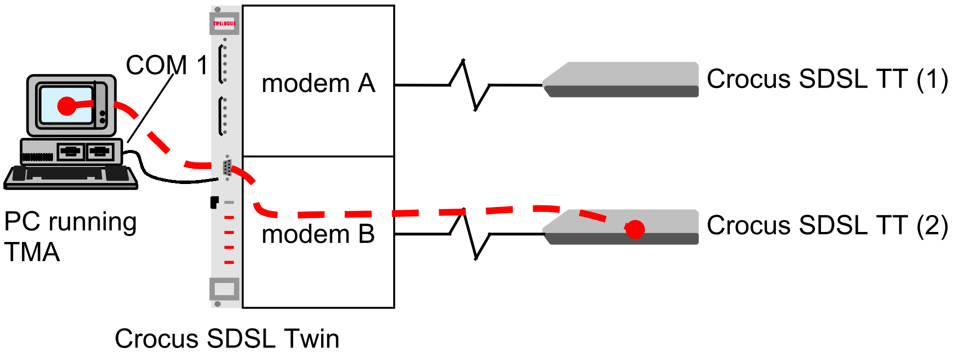
Replace `R:<nms>` by the relative address of the remote device you want to reach.

Replace `LINE<n>` by the line through which you can reach the remote device:

- A multi-device (i.e. a device that incorporates several devices on one card), consequently has several different (exit) lines (also called exit port). Actually, it has as many lines as there are devices on the card.
- A non multi-device (only one device on one card), only has one (exit) line (also called exit port). In that case you may drop the `LINE<n>` argument.

Example

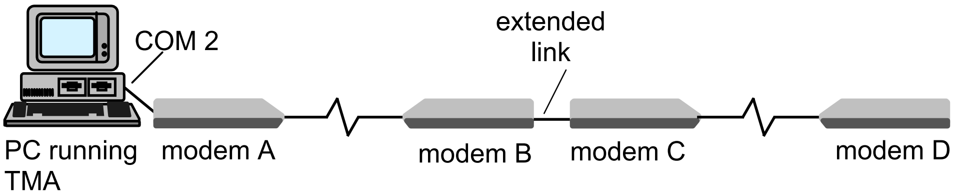
Suppose you have the following set-up:



In order to reach Crocus SDSL TT (2) through the Crocus SDSL Twin, type the following:

```
TmaCli com1 O R:1 lineB
```

Now also consider the following set-up:



In order to reach modem D through the modem A, type the following:

```
TmaCli com2 O R:3
```

4.2.10 Connecting using absolute addressing and an exit line

```
TmaCli [COM<x>|IP] O A:<nms> [LINE<n>]
```

Use this command to connect to a device located behind another device. The addressing method that is used is absolute addressing.

For more information on relative and absolute addressing, refer to the Section 4.3 - Relative and absolute addressing.

Address arguments

In case the connection between your computer running TMA CLI and the first device you want to connect through is …

- a direct connection, then specify the COM port of your computer: `COM<x>`.
- an IP connection, then specify the IP address of the first device: `<IP>`.

Replace `A:<nms>` by the absolute address of the remote device you want to reach.

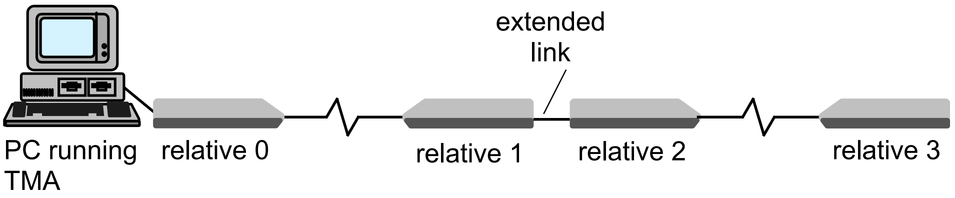
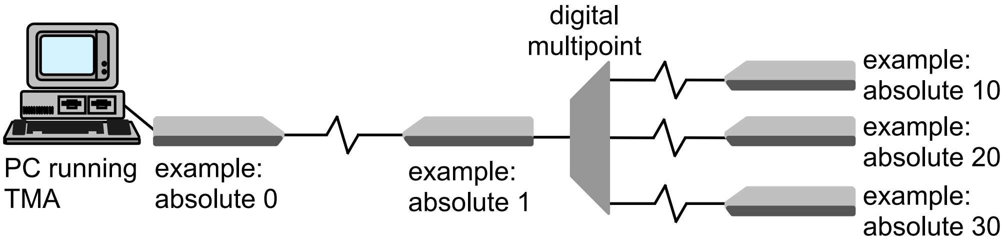
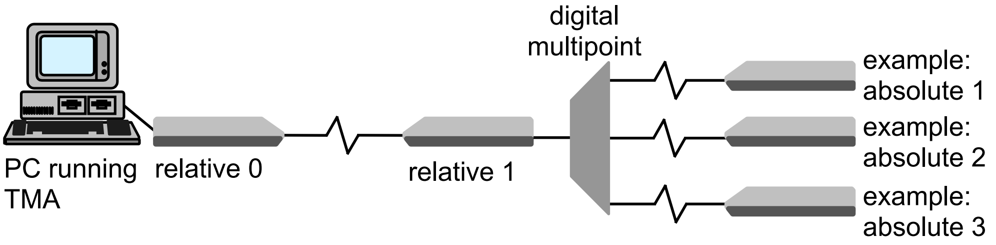
Replace `LINE<n>` by the line through which you can reach the remote device:

- A multi-device (i.e. a device that incorporates several devices on one card), consequently has several different (exit) lines (also called exit port). Actually, it has as many lines as there are devices on the card.
- A non multi-device (only one device on one card), only has one (exit) line (also called exit port). In that case you may drop the `LINE<n>` argument.

4.3 Relative and absolute addressing

This paragraph explains what relative and absolute addressing is.

If you want to open a TMA CLI session on a device, you have to specify the address of the device. You can apply three types of addressing methods:

Address type	Description
relative	This type of addressing is meant for a network topology where the modems are connected *in-line* on management level. E.g. with extended management links between two modems. An extended management link is realised with a cross connect cable between the auxiliary connectors of two modems.  extended link PC running TMA relative 0 relative 1 relative 2 relative 3 To enable relative addressing, no address has to be specified in the modem.
absolute	This type of addressing is meant for a network topology where the modems are not connected *in-line* on management level. E.g. when there is a digital multipoint device present.  digital multipoint PC running TMA example: absolute 0 example: absolute 1 example: absolute 10 example: absolute 20 example: absolute 30 To enable absolute addressing, an address has to be specified in the modem. The absolute addressing range goes from 0 up to 65535. Refer to the manual of the modem for more information.
relative and absolute	Relative and absolute addressing can be mixed. E.g. use relative addressing for the modems which are connected in-line. Use absolute addressing for the modems located after a digital multipoint device.  digital multipoint PC running TMA relative 0 relative 1 example: absolute 1 example: absolute 2 example: absolute 3

4.4 Connecting using name resolution

The Alarm Manager, a part of TMA for HP OpenView, can act as a name resolution server. As an alternative to the connection possibilities explained in Section 4.2 - The TMA CLI connection possibilities, you can start a TMA CLI session on a Telindus access device using this name resolution feature.

The following table gives an overview of this section.

4.4.1 Important remarks on name resolution

- As you will see in the following sections, name resolution makes use of the sysName and/or the name entered in the objectTable of the management concentrator. For name resolution to work properly, the sysName or objectTable name may not contain …
  - white spaces. Therefore do not use white space or use, for example, an underscore character instead. E.g. o1003lan, Orchid_1003_LAN, crocusHDSL1, etc.
  - quotes (“).
- It is *not possible* to use the sysName of a proxied IP device. Instead use the sysName of the proxy device (i.e. the management concentrator), underscore, the name of the proxied device as defined in the object table of the management concentrator. In other words: `<proxy_sysName>_<objectTable_name>`.
- If the Alarm Manager resides on a remote system, it is still possible to use its name resolution feature on your local system. You only have to define the remote system its IP address on your local system. Use the executable *DnsConfigure.exe* for this purpose. Refer to Section 4.4.4 - The executable DnsConfigure.exe.

4.4.2 Name resolution syntax for an IP device

To start a TMA CLI session on an IP device using name resolution, use the following syntax:

```
TmaCli <sysName>
```

Examples

- Suppose you have a Crocus E3 MUX connected to a LAN, with an IP address and with sysName = e3Mux. In that case, type the following:
  ```
  TmaCli e3Mux
  ```
- Suppose you have an Orchid 1003 LAN connected to a LAN, with an IP address and with sysName = Orchid_1003_LAN. In that case, type the following:
  ```
  TmaCli Orchid_1003_LAN
  ```

4.4.3 Name resolution syntax for a proxied (IP) device

To start a TMA CLI session on a proxied (IP) device using name resolution, you can use several syntax possibilities. The following paragraphs explain the different possibilities:

- Paragraph a) The syntax sysName – objectTable name, page 48
- Paragraph b) The syntax IP address – objectTable name, page 49
- Paragraph c) The syntax sysName – hardware connection, page 50
- Paragraph d) The syntax IP address – hardware connection, page 51

a) The syntax sysName – objectTable name

```
TmaCli <proxy_sysName>_<objectTable_name>
```

Examples:

- Suppose you have a Crocus HDSL connected to an Orchid 1003 LAN. The name for the Crocus HDSL in the Orchid its objectTable is hdslTT. The Orchid 1003 LAN on its turn is connected to a LAN, has an IP address and its is sysName = o1003. In that case, type the following:
  `TmaCli o1003_hdslTT`
- Suppose you have a Crocus DXC connected to an Orchid 1003 LAN. The name for the Crocus DXC in the Orchid its objectTable is Crocus_DXC. The Orchid 1003 LAN on its turn is connected to a LAN, has an IP address and its is sysName = Orchid_1003_LAN. In that case, type the following:
  `TmaCli Orchid_1003_LAN_Crocus_DXC`

b) The syntax IP address – objectTable name

```
TmaCli <proxy_IP>_<objectTable_name>
```

Example:

Suppose you have a Crocus HDSL connected to an Orchid 1003 LAN. The name for the Crocus HDSL in the Orchid its objectTable is hdslTT. The Orchid 1003 LAN on its turn is connected to a LAN and has IP address 12.3.45.100. In that case, type the following:

```
TmaCli 12.3.45.100_hdslTT
```

c) The syntax sysName – hardware connection

```
TmaCli <proxy_sysName>_<hardware_connection>
```

where the hardware connection can be one of the following two:

Hardware connection	Description
`R<nms>P<x>`	• Replace `R<nms>` by the relative address of the device. • Replace `P<x>` by the asynchronous port number of the management concentrator to which the device is connected.
`R<nms>R<r>.<p>.<m>`	• Replace `R<nms>` by the relative address of the device. • Replace `R<r>.<p>.<m>` by the hardware address of the device. The hardware address consists of: – `<r>` : rack address – `<p>` : position in the CN4 card nest – `<m>` : device position (A, B, C or D). For more information on the hardware address, refer to Section 4.2.8 - Connecting using relative addressing and a hardware address.

Note that the hardware connection syntax is different from the syntax used when connecting without name resolution (as explained in Section 4.2 - The TMA CLI connection possibilities).

Examples:

- Suppose you have a Crocus HDSL connected to an Orchid 1003 LAN via port 5. The Crocus HDSL has relative address 2. The Orchid 1003 LAN on its turn is connected to a LAN, has an IP address and its sysName = o1003. In that case, type the following:
  ```
  TmaCli o1003_R2P5
  ```
- Suppose you have a Crocus HDSL connected to an Orchid 1003 LAN via the high speed bus of the CN4 card nest. The CN4 card nest has address 1, the Crocus HDSL is in position 6 and you want to reach modem B. The Crocus HDSL has relative address 1. The Orchid 1003 LAN on its turn is connected to a LAN, has an IP address and its sysName = o1003. In that case, type the following:
  ```
  TmaCli o1003_R1R1.6.B
  ```

d) The syntax IP address – hardware connection

```
TmaCli <proxy_IP>_<hardware_connection>
```

Refer to Paragraph c) The syntax sysName – hardware connection for the explanation of the hardware connection parameter.

Example:

Suppose you have a Crocus HDSL connected to an Orchid 1003 LAN via port 5. The Crocus HDSL has relative address 2. The Orchid 1003 LAN on its turn is connected to a LAN and has IP address 12.3.45.100. In that case, type the following:

```
TmaCli 12.3.45.100_R2P5
```

4.4.4 The executable DnsConfigure.exe

What does this executable do?

It is possible that the Alarm Manager resides on a remote system. Even in that case it is possible to use its name resolution feature on your local system. Enable this by configuring the IP address of the remote system on your local system. Use the executable *DnsConfigure.exe* for this purpose.

The DnsConfigure.exe syntax

The DnsConfigure.exe syntax is as follows:

```
DnsConfigure <remote_system_IP_address>
```

or

```
DnsConfigure <remote_system_name>
```

In order to remove the path to the remote system, then type:

```
DnsConfigure -undo
```

The *DnsConfig* file

When you execute *DnsConfigure.exe*, a *DnsConfig* file is created in the directory *<TMA_path>\TMA\config*. This file contains the IP address or name of the remote system.

Typing the `DnsConfigure -undo` command removes the *DnsConfig* file from your system.

Example

Suppose the Alarm Manager runs on a machine called *MainSystem* which has IP address 12.0.34.100. Then type at the command prompt of your local system:

```
DnsConfigure 12.0.34.100
```

or

```
DnsConfigure MainSystem
```

4.5 Starting TMA CLI

TMA CLI can be started in interactive or non-interactive mode from a shell program of the operating system.

The following table gives an overview of this section.

4.5.1 Starting TMA CLI in interactive mode

What is interactive mode?

In interactive mode the user invokes commands to TMA CLI at run-time.

How to start TMA CLI in interactive mode?

To start TMA CLI in interactive mode, start a shell program (e.g. MS-DOS in the Windows operating system). At the prompt type `TmaCli` and then enter the necessary address arguments in order to reach the device on which you want to open a TMA CLI session. For more information on the different address arguments, refer to Section 4.2 - The TMA CLI connection possibilities.

The TMA CLI prompt

If the TMA CLI session was successfully started on the device, then the TMA CLI prompt appears displaying the top object of the device:

```
/crocusSDSLTT:"Edit Configuration"
>
```

For more information on the TMA CLI prompt, refer to Section 4.8 - The TMA CLI command line prompt.

The TMA CLI commands

At the prompt you can enter commands to communicate with the device. Refer to Chapter 8 - The TMA CLI commands for an overview and in-depth description of all available commands.

The *Cms2Serv.ini* file

The *Cms2Serv.ini* file, located in the *…/TMA/config* directory, contains the communications parameters. These parameters may have to be changed should you encounter difficulties connecting to a device. Refer to Chapter 9 - The Cms2Serv.ini file for a description of the communication parameters.

How to stop TMA CLI in interactive mode?

You can stop your current TMA CLI session at any moment by typing the *Disconnect* or *Exit* command. After such a command, TMA CLI closes the communication with the device and exits the command line interface.

4.5.2 Starting TMA CLI in non-interactive mode

What is non-interactive mode?

The main purpose of the non-interactive mode is to execute TMA CLI commands and scripts from another application.

The non-interactive mode syntax

Start a shell program (e.g. MS-DOS in the Windows operating system). At the prompt you can enter a command according to the following syntax:

```
TmaCli <address_arguments> -<TMA_CLI_command>
```

For more information on the different address arguments, refer to Section 4.2 - The TMA CLI connection possibilities. Refer to Chapter 8 - The TMA CLI commands for an overview and in-depth description of all available commands.

Examples

Suppose you want to retrieve all configuration attributes of a device that has the IP address 10.0.11.1. Type the following:

```
TmaCli 10.0.11.1 -get -r
```

In non-interactive mode you can easily redirect the same output to a file:

```
TmaCli 10.0.11.1 -get -r > output.txt
```

If you want to execute more than one command, then all the commands can be entered in a script file:

```
TmaCli 10.0.11.1 -exec script.cli
```

In non-interactive mode you can also interact with devices using flow control constructs of your scripting or programming environment. For more information on scripting, refer to Section 6.5.9 - Setting values obtained with the get command and Section 6.8 - Scripting.

Normally, if you want to set a textual string value that contains characters that conflict with the syntax (such as a comma, a space, an equal sign, etc.), then you have to surround the string by a set of double quote characters: “ “.

However, in MS-DOS the quote character is an escape character. This implies that you would be unable to set an attribute value containing a textual string using quotes in non-interactive TMA CLI mode. This is solved by placing a slash before each quote.

Example:

Do not use **`TmaCli 10.0.11.1 -set sysName = “Crocus SDSL TT”`**,
but use **`TmaCli 10.0.11.1 -set sysName = /“Crocus SDSL TT/”`** instead.

4.6 The TMA Comms Handler

What is the TMA Comms Handler?

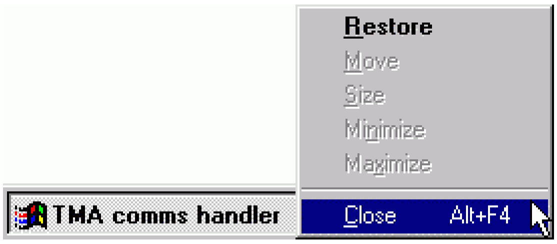
The TMA Comms Handler is a part of the TMA software that handles the communication towards the connected devices.

The TMA Comms Handler closing delay

If you connect through one of the COM ports of your computer, the TMA Comms Handler locks this COM port for other applications. If you close all TMA sessions and even if you close the TMA application, the TMA Comms Handler waits a certain period before it releases the previously used COM port(s). You can change this timeout with the WaitClose parameter in the *Cms2Serv.ini* file. As default, this timeout period is set to 5 minutes. For more information on this and other communication parameters, refer to Chapter 9 - The Cms2Serv.ini file.

How to force the TMA Comms Handler to close?

However, you can force the TMA Comms Handler to close even if the WaitClose time is not elapsed yet. To do so, proceed as follows:

Step	Action
1	Move the Windows arrow to the TMA Comms Handler icon on the taskbar.
2	Press on the right mouse button.
3	From the pop-up menu, choose *Close*.  Restore Move Size Minimize Maximize Close Alt+F4 TMA comms handler

4.7 Defining TMA CLI users and passwords

If a password has been configured in the Telindus access device, this password has to be transmitted before a TMA CLI session can be opened on this device. The TMA CLI password configuration tool *TmaUserConf.exe* allows you to create TMA users and assign a password to these users. The TMA users are related to the accounts that are defined on the management station.

Example

Suppose user Y logs on to a Windows NT management station and starts a TMA session on a Telindus access device.

Phase	Description
1	TMA CLI checks which TMA user is related with the account of user Y.
2	TMA CLI checks which password is assigned to this user.
3	TMA CLI sends this password to the Telindus access device.
4	If this password … • corresponds with the password defined in the Telindus access device, then a TMA CLI session opens on the device. • does not correspond with the password defined in the Telindus access device, then no TMA CLI session opens.

If no passwords are defined in the Telindus access devices, then you do not have to create TMA users and corresponding passwords. However, every time you start a TMA session on a device, the following warning will appear: `Warning: Could not read the user configuration file`. You can stop this warning from appearing by executing the *TmaUserConf.exe* application and closing it without entering any data.

However, because you created no TMA user with corresponding password, you will still get the following message: `Warning: Using empty password`.

4.8 The TMA CLI command line prompt

This section introduces the TMA CLI command line prompt.

Syntax

The TMA CLI command line prompt syntax is as follows:

```
"/" <list_of_object_names> ":" <group_name>
">"
```

The prompt elements

As you can see, the command prompt has the following elements:

Element	Description	Example
absolute path	Lists the currently selected object starting from the top object. The sub-objects are separated by a `/`.	`/crocusSDSLTT/modem`
separator	Separates the absolute path from the group name.	`:`
group name	Shows which group is currently selected: • `"Edit Configuration"` • `Status` • `Performance` • `Alarms`	`"Edit Configuration"`

Example

```
/crocusHDSLCV/modem/line[1]:Status
>
```

4.9 The environment variables

What are the environment variables?

The environment variables control the output behaviour of TMA CLI.

How to display the environment variables?

Using the command *setenv* you can list the environment variables and their corresponding value:

```
>setenv
SETENV LINES = 24
SETENV COLS = 80
SETENV VALUESEPARATOR = ^I
OK
```

What can you control with the environment variables?

With the environment variables you can control the following:

Variable	Description	Default value
`LINES`	This variable controls the number of lines that are visible on the terminal.	`24`
`COLS`	This variable controls the number of characters that are put on a line. The value ranges from 80 up to 512. If you enter a number outside this range, the `COLS` value is set to the nearest extreme of the range (being 80 or 215). This variable only has an impact on the output, not on the input. The input solely depends on the maximum line length that TMA CLI can parse (being 512). This variable is not used in the *get -v* command.	`80`
`VALUESEPARATOR`	This value defines the value separator character. This character is used when retrieving a table with the *get -v* command. You can specify any character.	`<tab>`

How to change the environment variables?

Using the command *setenv* you can also change the value of an environment variable. For example:

```
>setenv LINES = 50
OK
```

5. Introducing the containment tree

This chapter explains what the *containment tree* of a device is. It also introduces terms such as *object*, *attribute*, *simple value*, *structured or complex value*, etc. At the end it gives an example of a containment tree and how it looks like in TMA CLI.

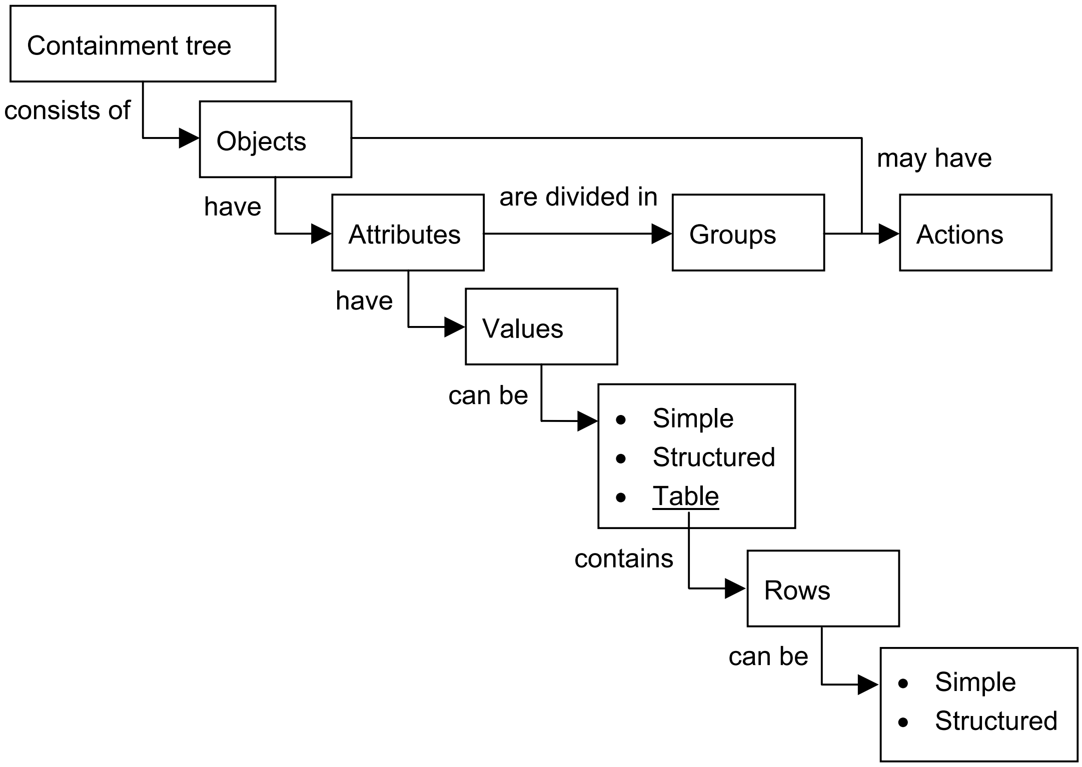
The following table gives an overview of this chapter.

5.1 What is a containment tree?

A containment tree represents the parameters of a Telindus device in a hierarchical structure. These parameters enable you to configure the device and they display information on the operational status of the device.

5.2 How is the containment tree structured?

The following figure gives a simplified representation of how a containment tree is structured:



The specific structure of the containment tree differs from device to device, but the general structure is always the same (as depicted above).

The following paragraph describes each element of the containment tree.

5.3 The containment tree terminology

The following table explains the terminology associated with the containment tree:

<table>
<tr><th>Term</th><th>Description</th></tr>
<tr><td>containment tree</td><td>The containment tree represents the hierarchical structure of a device. It is composed of a number of objects that are ordered in a tree. This tree resembles a Windows directory structure:
• it is also a levelled structure, with nodes which can be expanded or reduced
• the containment tree objects can be compared with file folders
• the objects contain attributes like file folders contain files.</td></tr>
<tr><td>object</td><td>An object represents a physical interface, an application or a combination of both. Each object has its own set of attributes.</td></tr>
<tr><td>attribute</td><td>An attribute is a parameter related to a certain object. It has a certain value.</td></tr>
<tr><td>simple value</td><td>An attribute has a certain value which is …
• changeable in case of a configuration attribute (provided you have write access)
• read only in case of a status, performance and alarm attribute.
An attribute has a simple value when it has no “underlying attributes” or “sub-attributes” which are called elements. Hence, we also call this attribute a <i>simple attribute</i>. E.g. sysName, sysLocation, sysContact and bootFromFlash are simple attributes.</td></tr>
<tr><td>structured or complex value</td><td>An attribute has a structured or complex value when it has “underlying attributes” or “sub-attributes” which are called <i>elements</i>. Hence, we also call this attribute a <i>structured</i> or <i>complex attribute</i>.
There are three different structured or complex value types:
<table>
<tr><th>Type</th><th>Description</th></tr>
<tr><td>bit string
11001100</td><td>A bit string is actually a special case of a simple value. It does have elements, in this case the separate bits, but you can not display these bits separately. I.e. you can only display the complete bit string. In other words, a bit string is actually a simple value which, in some cases, is represented as a complex value.
An example of a bit string is the attribute alarmMask.</td></tr>
<tr><td>structure
<Struct&gt;</td><td>A structure has several elements. The elements of a structure are always simple values. You can display these elements separately.
An example of a structure is the attribute alarmLevel.</td></tr>
<tr><td>table
&lt;Table&gt;</td><td>A table has several elements. The elements of a table can be simple values, complex values or a combination of both. You can display these elements separately.
An example of a table is the attribute security.</td></tr>
</table></td></tr>
</table>

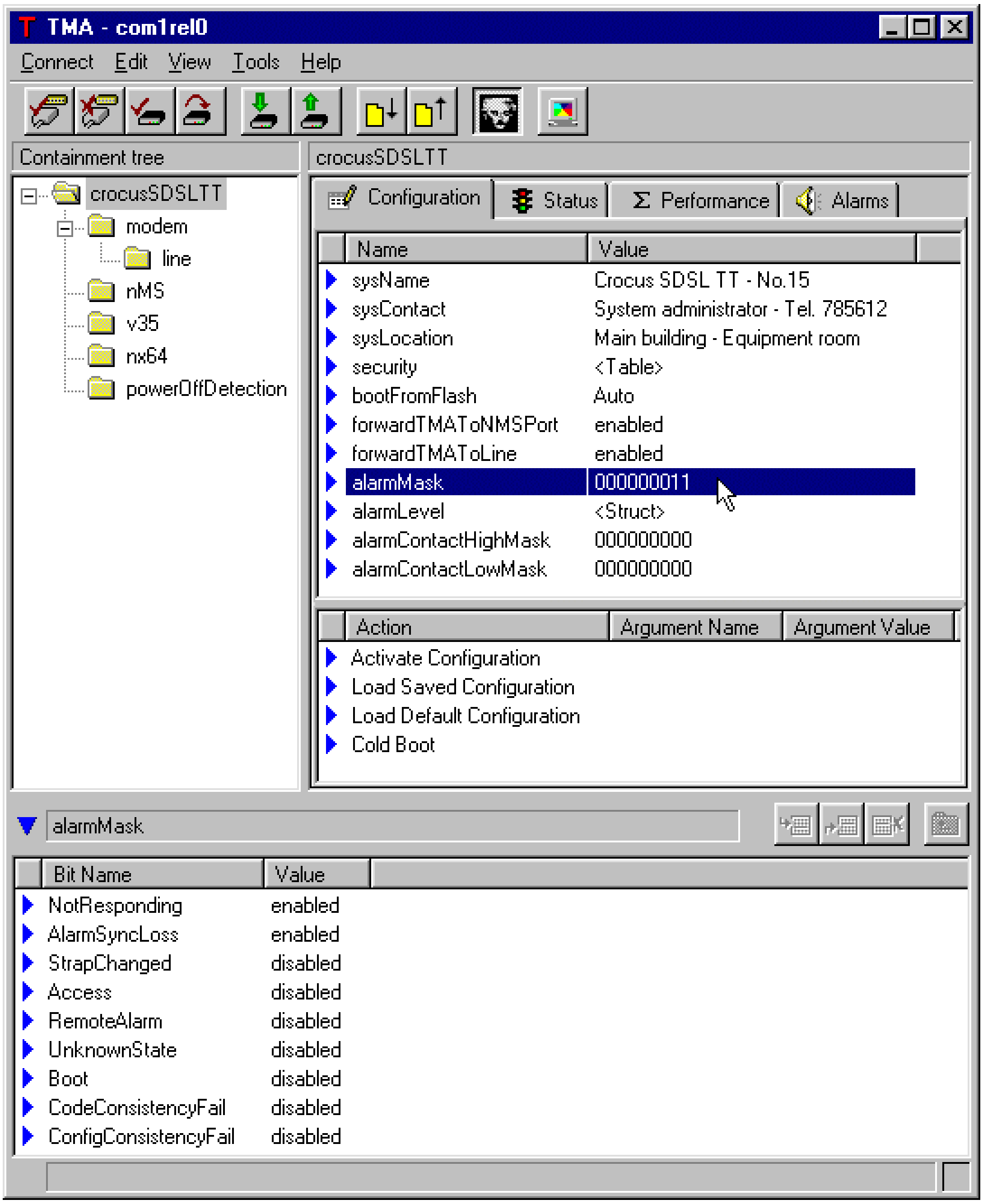
Continued on next page

The containment tree terminology (continued)

Term	Description
group	Groups assemble a set of attributes related by functionality. There are four groups: • configuration • status • performance • alarms.
action	An object may have actions assigned to it in a certain group.

5.4 A containment tree example

Consider the containment tree of a Crocus SDSL TT as an example. If we open a TMA session on this device, the graphical representation is as follows:



Name	Value
sysName	Crocus SDSL TT - No.15
sysContact	System administrator - Tel. 785612
sysLocation	Main building - Equipment room
security	<Table>
bootFromFlash	Auto
forwardTMAToNMSPort	enabled
forwardTMAToLine	enabled
alarmMask	000000011
alarmLevel	<Struct>
alarmContactHighMask	000000000
alarmContactLowMask	000000000

Action	Argument Name	Argument Value
Activate Configuration		
Load Saved Configuration		
Load Default Configuration		
Cold Boot		

Bit Name	Value
NotResponding	enabled
AlarmSyncLoss	enabled
StrapChanged	disabled
Access	disabled
RemoteAlarm	disabled
UnknownState	disabled
Boot	disabled
CodeConsistencyFail	disabled
ConfigConsistencyFail	disabled

The following paragraphs will help you to get acquainted with the containment tree, its objects and attributes as they appear in CLI.

Continued on next page

A containment tree example (continued)

If you open a TMA CLI session on this Crocus SDSL TT, the initial command line prompt looks as follows:

```
/crocusSDSLTT:"Edit Configuration"
>
```

This means you are currently in the top object crocusSDSLTT and the group Configuration. As you can see in the graphical representation, the top object crocusSDSLTT contains …

- several sub-objects. E.g. modem, nMS, powerOffDetection, etc.
- several attributes with a simple value, also called simple attributes. E.g. sysName, sysLocation, bootFromFlash, etc.
- several attributes with a structured or complex value, also called complex attributes. E.g. alarmMask, alarmLevel, security etc.

If you select a sub-object, for instance modem, the command line prompt looks as follows:

```
/crocusSDSLTT/modem:"Edit Configuration"
>
```

If you now select another group, for instance Status, the command line prompt looks as follows:

```
/crocusSDSLTT/modem:Status
>
```

Continued on next page

A containment tree example (continued)

Displaying the value of the simple attribute bootFromFlash in TMA CLI results in the following:

```
bootFromFlash = Auto
```

Displaying the value of the bit string alarmMask in TMA CLI results in the following:

```
alarmMask =
  {
  NotResponding = enabled
  AlarmSyncLoss = enabled
  StrapChanged = disabled
  Access = disabled
  RemoteAlarm = disabled
  UnknownState = disabled
  Boot = disabled
  CodeConsistencyFail = disabled
  ConfigConsistencyFail = disabled
  }
```

Because this is a bit string, it is not possible to display the value of one bit only.

Displaying the value of the complex attribute (in this case a structure) alarmLevel in TMA CLI results in the following:

```
alarmLevel =
  {
  NotResponding = 4
  AlarmSyncLoss = 4
  StrapChanged = 1
  Access = 1
  RemoteAlarm = 0
  UnknownState = 0
  Boot = 1
  CodeConsistencyFail = 1
  ConfigConsistencyFail = 1
  }
```

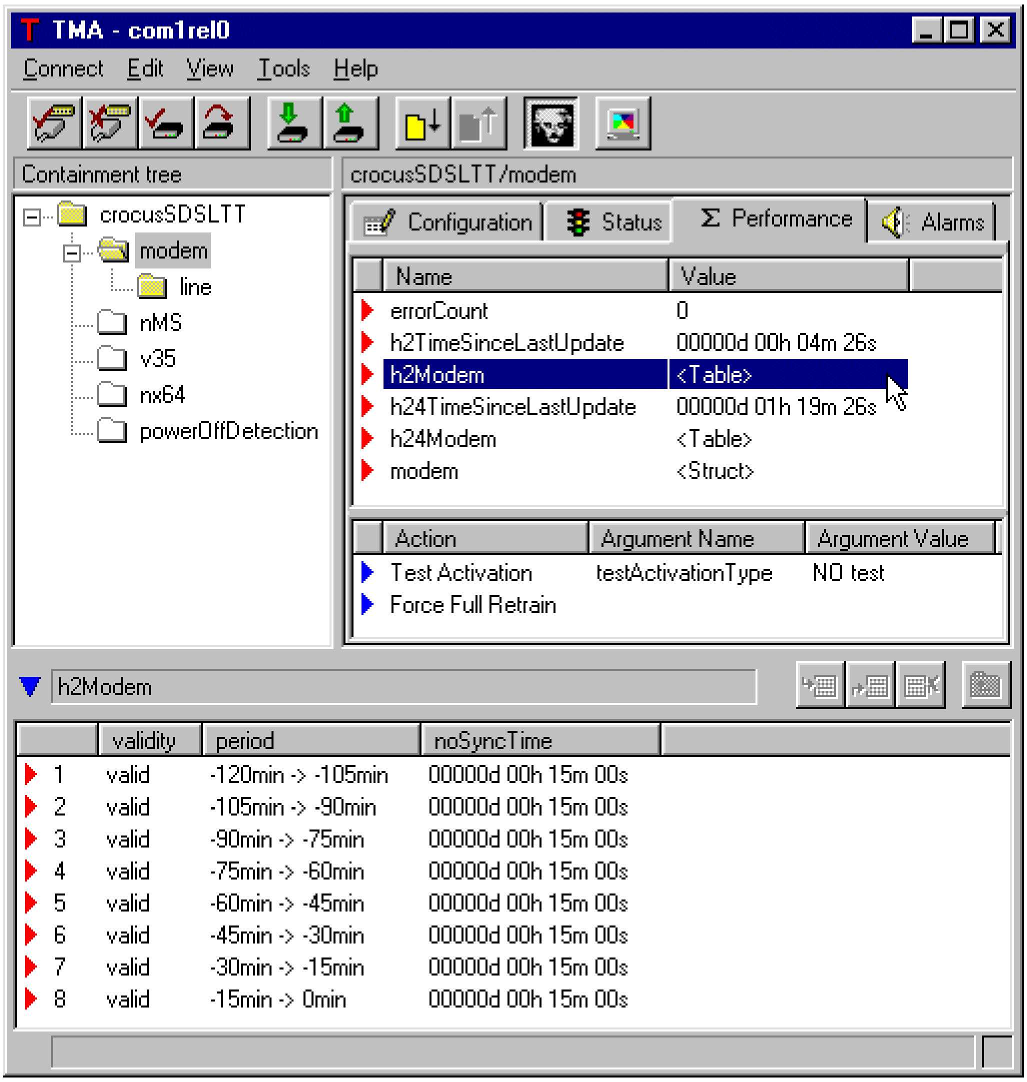
As you can see, this complex attribute is made up of several simple attributes (elements). Displaying the value of only one simple attribute (or element) results in the following:

```
alarmLevel =
  {
  NotResponding = 4
  }
```

Continued on next page

A containment tree example (continued)

If you switch to the Performance group and go to the modem object, the graphical representation in TMA is as follows:



Name	Value
errorCount	0
h2TimeSinceLastUpdate	00000d 00h 04m 26s
h2Modem	<Table>
h24TimeSinceLastUpdate	00000d 01h 19m 26s
h24Modem	<Table>
modem	<Struct>

Action	Argument Name	Argument Value
Test Activation	testActivationType	NO test
Force Full Retrain		

	validity	period	noSyncTime
1	valid	-120min -> -105min	00000d 00h 15m 00s
2	valid	-105min -> -90min	00000d 00h 15m 00s
3	valid	-90min -> -75min	00000d 00h 15m 00s
4	valid	-75min -> -60min	00000d 00h 15m 00s
5	valid	-60min -> -45min	00000d 00h 15m 00s
6	valid	-45min -> -30min	00000d 00h 15m 00s
7	valid	-30min -> -15min	00000d 00h 15m 00s
8	valid	-15min -> 0min	00000d 00h 15m 00s

The command line prompt now looks as follows:

```
/crocusSDSLTT/modem:Performance
>
```

Continued on next page

A containment tree example (continued)

The TMA window on the previous page displays the h2Modem attribute. This is a complex attribute, more specifically a table. Displaying this attribute in TMA CLI results in the following:

```
h2Modem =
  {
  [1] =
    {
    validity = valid
    period = "-120min -> -105min"
    noSyncTime = "0d 0h 15m 0s"
    }

...

  [8] =
    {
    validity = valid
    period = "-15min -> 0min"
    noSyncTime = "0d 0h 15m 0s"
    }
  }
```

As you can see in the TMA window, the h2Modem table consists of several rows. Displaying only one row (e.g. row 4) of this table results in the following:

```
h2Modem[4] =
  {
  validity = valid
  period = "-75min -> -60min"
  noSyncTime = "0d 0h 15m 0s"
  }
```

As you can see in the TMA window, each row of the h2Modem table contains three elements: validity, period and noSyncTime. Displaying only one attribute (e.g. validity) on one row of this table results in the following:

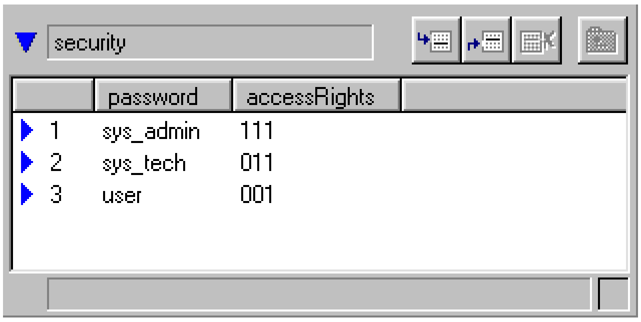
```
h2Modem[4] =
  {
  validity = valid
  }
```

Continued on next page

A containment tree example (continued)

The table discussed on the previous page is a table with a fixed length. I.e. the h2Modem table always has 8 entries. The following example is a table with variable length. I.e. the number of entries is not a fixed number.

If you return to the Configuration group and the top object crocusSDSLTT, you find the attribute security. The first time you connect to the Crocus SDSL, this table is empty. However, you can enter several lines in it. For example:



security

	password	accessRights
1	sys_admin	111
2	sys_tech	011
3	user	001

Displaying the same attribute in TMA CLI results in the following:

```
security =
  {
  [a] =
    {
    password = sys_admin
    accessRights =
      {
      ReadAccess = on
      WriteAccess = on
      SecurityAccess = on
      }
    }
  [a] =
    {
    password = sys_tech
    accessRights =
      {
      ReadAccess = on
      WriteAccess = on
      SecurityAccess = off
      }
    }
  [a] =
    {
    password = user
    accessRights =
      {
      ReadAccess = on
      WriteAccess = off
      SecurityAccess = off
      }
    }
  }
```

Now, you can see the difference between a table with a fixed length and with a variable length:

A table with …	is indexed with …
a fixed length	an integer number, e.g. `[1]`.
a variable length	the following character: `[a]`.

Continued on next page

A containment tree example (continued)

However, also in a table with a variable length you can display only one row. For example row 2 in the security table:

```
security[2] =
  {
  password = sys_tech
  accessRights =
    {
    ReadAccess = on
    WriteAccess = on
    SecurityAccess = off
    }
  }
```

Every row of the security table contains two elements:

- a simple value: password
- a bit string: accessRights. In this bit string the three bits grant:
  - ReadAccess
  - WriteAccess
  - SecurityAccess

Now that you are more or less acquainted with the Telindus containment tree as it is represented in CLI, you are ready to try the commands of CLI.

6. Basic TMA CLI commands

This chapter teaches you the basic TMA CLI commands. These commands enable you to browse through the containment tree, select another group, read values, set values and perform actions. It also shows you how to log your activity. This is particularly useful for scripting. Furthermore this chapter describes how to import a configuration and how to download new firmware.

The following table gives an overview of this chapter.

- TMA CLI commands are not case sensitive whereas their parameters are.
- Do not confuse CLI with TMA CLI! Whereas CLI is a management tool which runs on the Telindus access devices themselves, TMA CLI is a program which you have to install on a computer. TMA CLI is a more elaborate command line interface, with more features and commands then CLI. Whereas most of the TMA CLI commands are the same as the CLI commands, some commands that exist in TMA CLI do not exist in CLI. These commands are:
`exit, logging, exec, cfgload, memload and source.`

6.1 Browsing through the containment tree

Using the *select* command, you can navigate through the containment tree. The destination object is entered as an argument of the *select* command. Depending on the presence or absence of the `/<topObject>` at the beginning of the argument, we speak of an absolute or a relative path.

Path	In this case, the argument starts from …	Example
absolute	the top object.	`/crocusSDSLTT/modem/line`
relative	the next object in the object path.	`modem/line`

The following are some examples of how to browse through the containment tree.

From the current object to a sub-object (relative addressing)

```
/crocusSDSLTT:“Edit Configuration”
>select modem
OK
/crocusSDSLTT/modem:“Edit Configuration”
>
```

From the current object two sub-objects further (relative addressing)

```
/crocusSDSLTT:“Edit Configuration”
>select modem/line
OK
/crocusSDSLTT/modem/line:“Edit Configuration”
>
```

From a sub-object immediately to another sub-object (absolute addressing)

```
/crocusSDSLTT/modem/line:“Edit Configuration”
>select /crocusSDSLTT/nMS
OK
/crocusSDSLTT/nMS:“Edit Configuration”
>
```

From the current object go up one object level

```
/crocusSDSLTT/modem:“Edit Configuration”
>select ..
OK
/crocusSDSLTT:“Edit Configuration”
>
```

From the current object immediately to the top object

```
/crocusSDSLTT/modem/line:“Edit Configuration”
>select /
OK
/crocusSDSLTT:“Edit Configuration”
>
```

6.2 Selecting another group of attributes

Using the *selgrp* command you can select another group of attributes. The four possible groups are:

- “Edit Configuration”
- Status
- Performance
- Alarms

```
/crocusSDSLTT:“Edit Configuration”
>selgrp Status
OK
/crocusSDSLTT:Status
>
```

6.3 Reading attribute values

This section explains how you can use the *get* command to read a simple value, a structured (or complex) value, a simple value within a structured value, etc.

The following table gives an overview of this section.

6.3.1 Reading a simple attribute value

Reading a simple attribute value

```
/crocusSDSLTT:"Edit Configuration"
>get bootFromFlash
  bootFromFlash = Auto
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Reading a simple attribute value located on a lower object level

```
/crocusSDSLTT:"Edit Configuration"
>get modem/line/speed
  GET
      {
      SELECT modem
        {
        SELECT line
          {
          LIST
            {
            speed = "1152000 bps"
            }
          }
        }
      }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Because the *get* command for the attribute speed of the sub-object line was executed in the top object crocusSDSLTT, the underlying objects are selected by the *SELECT* command. The speed attribute of the line object is grouped by *LIST { … }*. This allows you to use the *get* command to create scripts. You can then use these scripts to easily change identical attributes on other devices of the same type. For more information on this matter, refer to Section 6.5.9 - Setting values obtained with the get command.

6.3.2 Reading a structured attribute value

There are three different structured (or complex) value types:

Type	Description	Example
bit string	A bit string is actually a special case of a simple value. It does have elements, in this case the separate bits, but you can not display these bits separately. I.e. you can only display the complete bit string. In other words, a bit string is actually a simple value which, in some cases, is represented as a structured value.	The following configuration attribute is a bit string: crocusSDSLTT/alarmMask
structure	A structure has several elements. The elements of a structure are always simple values. You can display these elements separately.	The following configuration attribute is a structure: crocusSDSLTT/alarmLevel
table	A table has several elements. The elements of a table can be simple values, complex values or a combination of both. You can display these elements separately. Because a table is a quite complicated attribute value, it is explained in Section 6.3.3 - Reading a table.	The following configuration attribute is a table: crocusSDSLTT/security

Reading a bit string

```
/crocusSDSLTT:"Edit Configuration"
>get alarmMask
  alarmMask =
    {
    NotResponding = enabled
    AlarmSyncLoss = enabled
    StrapChanged = disabled
    Access = disabled
    RemoteAlarm = disabled
    UnknownState = disabled
    Boot = disabled
    CodeConsistencyFail = disabled
    ConfigConsistencyFail = disabled
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Because this is a bit string, you can not display the elements separately. For instance, you can not display the bit value NotResponding only.

Continued on next page

Reading a structured attribute value (continued)

Reading a structure

```
/crocusSDSLTT:"Edit Configuration"
>get alarmLevel
  alarmLevel =
    {
    NotResponding = 4
    AlarmSyncLoss = 4
    StrapChanged = 1
    Access = 1
    RemoteAlarm = 0
    UnknownState = 0
    Boot = 1
    CodeConsistencyFail = 1
    ConfigConsistencyFail = 1
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Because this is a structure, you can display the elements separately. See below.

Reading only one element in a structure

```
>get alarmLevel/NotResponding
  alarmLevel =
    {
    NotResponding = 4
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.3.3 Reading a table

As already explained in Section 6.3.2 - Reading a structured attribute value, a table is a structured or complex value. I.e. it has several elements. These can be simple values, complex values or a combination of both. You can display these values separately.

Reading a table

You can, for instance, display the security table:

```
/crocusSDSLTT:"Edit Configuration"
>get security
  security =
    {
    [a] =
      {
      password = sys_admin
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = on
        }
      }
    [a] =
      {
      password = sys_tech
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = off
        }
      }
    [a] =
      {
      password = user
      accessRights =
        {
        ReadAccess = on
        WriteAccess = off
        SecurityAccess = off
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Reading a table (continued)

Reading only one row of a table

You can also read only one row of the security table. In the example below the first row of the security table is retrieved. You can see that this row consists of the simple value password and the structured value accessRights.

```
/crocusSDSLTT:"Edit Configuration"
>get security[1]
  security[1] =
    {
    password = sys_admin
    accessRights =
      {
      ReadAccess = on
      WriteAccess = on
      SecurityAccess = on
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Reading only one row of a table is only supported in the configuration group.

Reading only one element in a row of a table

You can also read only one attribute value in one row of the security table. Suppose you want to read the password attribute value in the first row of the security table:

```
/crocusSDSLTT:"Edit Configuration"
>get security[1]/password
  security[1] =
    {
    password = sys_admin
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Reading a table (continued)

Reading an element in a structured value in a row of a table

Let us now consider the objectTable attribute of the Orchid 1003 LAN. Retrieving only one row of this table results in the following:

```
/o1003/nmsgroup:"Edit Configuration"
>get objectTable[1]
  objectTable[1] =
    {
    name = "SDSL TT central"
    centralRemote = central
    configFileName = <OPT>
    ipAddress = <OPT>
    deviceType = cms2
    addressType = relative
    addressValue = 0
    exitPort =
      {
      exitPortType = port
      portNumber = 1
      cardnestAddress = 0
      cardPosition = 0
      modem = A
      }
    pollTimeOut = 3
    mapNumber = <OPT>
    filter = 0
    }
OK
/o1003/nmsgroup:"Edit Configuration"
>
```

As you can see, this row consists of several simple values and one complex value: exitPort. Because the exitPort value is a *structure* (refer to Section 6.3.2 - Reading a structured attribute value), you can also display its elements separately. So, displaying the simple value cardPosition in the complex value exitPort in the first row of the objectTable, results in the following:

```
/o1003/nmsgroup:"Edit Configuration"
>get objectTable[1]/exitPort/cardPosition
  objectTable[1] =
    {
    exitPort =
      {
      cardPosition = 0
      }
    }
OK
/o1003/nmsgroup:"Edit Configuration"
>
```

6.3.4 Reading all attribute values of the current object

If you add neither a parameter nor an option to the *get* command, then you will obtain all attributes of the current object.

Reading all attribute values of the current object

```
/crocusSDSLTT/powerOffDetection:"Edit Configuration"
>get
  mode = disabled
  alarmMask =
    {
    "Open line" = disabled
    "Remote power fail" = disabled
    "Short circuit" = disabled
    }
  alarmLevel =
    {
    "Open line" = 3
    "Remote power fail" = 3
    "Short circuit" = 3
    }
  alarmContactHighMask =
    {
    "Open line" = disabled
    "Remote power fail" = disabled
    "Short circuit" = disabled
    }
  alarmContactLowMask =
    {
    "Open line" = disabled
    "Remote power fail" = disabled
    "Short circuit" = disabled
    }
OK
/crocusSDSLTT/powerOffDetection:"Edit Configuration"
>
```

Continued on next page

Reading all attribute values of the current object (continued)

Reading all attribute values of an object on a lower object level

```
/crocusSDSLTT:"Edit Configuration"
>get modem/line
  GET
    {
    SELECT modem
      {
      SELECT line
        {
        LIST
          {
          speed = "1152000 bps"
          alarmMask =
            {
            LinkDown = disabled
            Retrain = disabled
            HighBitError = disabled
            LowBitError = disabled
            SeverelyErroredSecond = disabled
            Unavailability = disabled
            }
          alarmLevel =
            {
            LinkDown = 3
            Retrain = 2
            HighBitError = 2
            LowBitError = 1
            SeverelyErroredSecond = 2
            Unavailability = 2
            }
          alarmContactHighMask =
            {
            LinkDown = disabled
            Retrain = disabled
            HighBitError = disabled
            LowBitError = disabled
            SeverelyErroredSecond = disabled
            Unavailability = disabled
            }
          alarmContactLowMask =
            {
            LinkDown = disabled
            Retrain = disabled
            HighBitError = disabled
            LowBitError = disabled
            SeverelyErroredSecond = disabled
            Unavailability = disabled
            }
          }
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Because the *get* command for the sub-object line was executed in the top object crocusSDSLTT, the underlying objects are selected by the *SELECT* command. The attributes of the line object are grouped by *LIST { … }*. This allows you to use the *get* command to create scripts. You can then use these scripts to easily change identical attributes on other devices of the same type. For more information on this matter, refer to Section 6.5.9 - Setting values obtained with the get command.

Continued on next page

Reading all attribute values of the current object (continued)

Reading all attribute values of an object in a group different from the current one

Suppose your current location is the top object crocusSDSLTT in the *“Edit Configuration”* group. If you, for example, want to read the *status* attribute values of the line object, you can do this without having to change from the *“Edit Configuration”* group to the *Status* group. In order to do so, type the following:

```
/crocusSDSLTT:"Edit Configuration"
>get modem/line:Status
  GET
    {
    SELECT modem
      {
      SELECT line
        {
        LIST
          {
          timeSinceLastRetrain = "0d 0h 0m 0s"
          lineState = idle
          lineAttenuation = 62.0dB
          noiseMargin = -16.0dB
          ifSpeed = 0
          ifOperStatus = down
          }
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.3.5 The *get* command options

The following table shows which options can be used in combination with the *get* command. The options are explained in the following paragraphs.

Option	Description	Refer to paragraph …
`get -d`	Use this option to display only those values that are different from the default or read-only values.	a) The get –d option, page 87
`get -f`	Use this option to retrieve the full non-default configuration, independent of your current location in the containment tree. In other words, it displays all the attributes that are set to a value different from their default value. The attributes on the same level are grouped by *LIST{ … }* and an underlying object is selected by the *SELECT* command. This is the same as executing `get -r -d` *when located in the top object of the containment tree*.	b) The get –f option, page 88
`get -m`	Use this option to display as much simple attribute values on one line as possible. The number of attributes that will be printed on one line depends on the environment variable `COLS` (refer to Section 4.9 - The environment variables).	c) The get –m option, page 89
`get -r`	Use this option (recursive get) to retrieve not only the attributes of the current object, but also of the underlying objects. The attributes on the same level are grouped by *LIST{ … }* and an underlying object is selected by the *SELECT* command.	d) The get –r option, page 90
`get -s`	Use this option to use a filter on a table. By doing so you can selectively filter certain rows out of a table.	e) The get –s option, page 91
`get -t`	Use this option to retrieve nested tables. This means that in case you perform a `get -t` of a table in which structures and other tables (i.e. nested tables) appear, you retrieve all the values present in these structures and tables. The ordinary `get` command does not do this by default as some tables can be very complex and might take a very long time to retrieve. Note however that `get -f` and `get -r` implicitly retrieve nested tables.	f) The get –t command, page 94

Continued on next page

The get command options (continued)

Option	Description	Refer to paragraph …
`get -v`	Use this option to display the values of a table in rows and columns and separated by the value separator. The first row of the output table represents the value names. The used value separator depends on the environment variable `VALUESEPARATOR` (refer to Section 4.9 - The environment variables). As default, it is a `<tab>`.	g) The get –v option, page 95
`get -w`	Use this option to display the values of a complex attribute value on one line. The number of elements that will be printed on one line depends on the environment variable `COLS` (refer to Section 4.9 - The environment variables).	h) The get –w option, page 96

6.3.6 Combinations with the *get* command options

Most (but not all) *get* command options can be combined. In this way you can format the output to your needs. The following is a non-exhaustive list of combinations:

Option combination	Description
`get -d -m`	Displays only those values that are different from the default or read-only values and displays as much simple attribute values on one line as possible.
`get -d -s`	Displays only those values that are different from the default or read-only values and applies a filter on a table.
`get -d -v`	Displays only those values that are different from the default or read-only values and displays the values of a table in rows and columns and separated by the value separator.
`get -d -w`	Displays only those values that are different from the default or read-only values and displays the values of a complex attribute value on one line.
`get -r -d`	Displays the values of the current and all underlying objects and displays only those values that are different from the default or read-only values.
`get -r -m`	Displays the values of the current and all underlying objects and displays as much simple attribute values on one line as possible.
`get -r -v`	Displays the values of the current and all underlying objects and displays the values of a table in rows and columns and separated by the value separator.
`get -r -w`	Displays the values of the current and all underlying objects and displays the values of a complex attribute value on one line.
`get -t -d`	Displays only those values that are different from the default or read-only values, including those in nested tables.
`get -t -s`	Applies a filter on a table and displays the nested tables.
`get -t -v`	Displays the values of a table in rows and columns and separated by the value separator. In this case, however, *the nested tables are not displayed*.
`get -t -w`	Displays the values of a complex attribute value on one line and displays the nested tables.

a) The *get –d* option

Use this option to display only those values which are different from the default or read-only values.

```
/crocusSDSLTT:"Edit Configuration"
>get -d
  sysName = "Crocus SDSL TT - No.15"
  sysContact = "System administrator - Tel. 785612"
  sysLocation = "Main building - Equipment room"
  security =
    {
    [a] =
      {
      password = sys_admin
      }
    [a] =
      {
      password = sys_tech
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = off
        }
      }
    [a] =
      {
      password = user
      accessRights =
        {
        ReadAccess = on
        WriteAccess = off
        SecurityAccess = off
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

b) The *get –f* option

Use this option to retrieve the full non-default configuration, independent of your current location in the containment tree. In other words, it displays all the attributes that are set to a value different from their default value. It also includes the command *“Load Default Configuration”* at the beginning and *“Activate Configuration”* at the end of the attribute list.

The attributes on the same level are grouped by *LIST{ … }* and an underlying object is selected by the *SELECT* command. This allows you to use the *get* command to create scripts. You can then use these scripts to easily change identical attributes on other devices of the same type. For more information on this matter, refer to Section 6.5.9 - Setting values obtained with the get command.

- Executing `get -f` is the same as executing `get -r -d` *when located in the top object of the containment tree*.
- If you want a full configuration (i.e. not only the non-default configuration), then execute `get -r` *when located in the top object of the containment tree*.

```
/crocusSDSLTT/modem/line:"Edit Configuration"
>get -f
action "Load Default Configuration"
  SET
    {
    LIST
      {
      sysName = "Crocus SDSL"
      sysContact = "Sys Admin"
      sysLocation = "Computer Room"
      }
    SELECT modem
      {
      LIST
        {
        channel = central
        }
      SELECT line
        {
        }
      }
    SELECT nMS
      {
      }
    SELECT v35
      {
      }
    SELECT nx64
      {
      }
    SELECT powerOffDetection
      {
      }
    }
action "Activate Configuration"
OK
/crocusSDSLTT/modem/line:"Edit Configuration"
>
```

The *get -f* command is only supported in the configuration group.

c) The *get –m* option

Use this option to display as much simple attribute values on one line as possible.

The number of attributes that will be printed on one line depends on the environment variable `COLS` (refer to Section 4.9 - The environment variables).

```
/crocusSDSLTT/modem:Status
>get -m
  testType = "NO test"    testOriginator = unknown    testStatus = unknown
  errorCount = 0    ifDescr = Modem    ifType = 1    ifSpeed = 0
  ifOperStatus = down
OK
/crocusSDSLTT/modem:Status
>
```

d) The *get –r* option

Use this option (recursive get) in order to obtain not only the attributes of the current object, but also of the underlying objects.

The attributes on the same level are grouped by *LIST{ … }* and an underlying object is selected by the *SELECT* command. This allows you to use the *get* command to create scripts. You can then use these scripts to easily change identical attributes on other devices of the same type. For more information on this matter, refer to Section 6.5.9 - Setting values obtained with the get command.

```
/crocusSDSLTT:"Edit Configuration"
>get -r
  GET
    {
    LIST
      {
      sysName = "Crocus SDSL TT - No.15"
      sysContact = "System administrator - Tel. 785612"
      sysLocation = "Main building - Equipment room"
      bootFromFlash = Auto
      ...
      alarmMask =
        {
        NotResponding = enabled
        ...
        ConfigConsistencyFail = disabled
        }
      ...
      }
    SELECT modem
      {
      LIST
        {
        tests =
          {
          keyboardET = enabled
          ...
          aLDuration = "0d 0h 3m 0s"
          }
        ...
        channel = remote
        }
      SELECT line
        {
        LIST
          {
          speed = "1152000 bps"
          alarmMask =
            {
            LinkDown = disabled
            ...
            Unavailability = disabled
            }
          ...
          }
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

The *get -r* command is only supported in the configuration group.

e) The *get –s* option

Use this option to use a filter on a table. By doing so you can selectively filter certain rows out of a table.

Suppose the security table of the Orchid 1003 LAN contains the following values:

```
/o1003:"Edit Configuration"
>get security
  security =
    {
    [a] =
      {
      password = sys_admin
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = on
        FileSystemAccess = on
        }
      }
    [a] =
      {
      password = sys_tech
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = off
        FileSystemAccess = on
        }
      }
    [a] =
      {
      password = user
      accessRights =
        {
        ReadAccess = on
        WriteAccess = off
        SecurityAccess = off
        FileSystemAccess = off
        }
      }
    }
OK
/o1003:"Edit Configuration"
>
```

Suppose you only want to display that line of the security table where the password “user” is present. In that case, type the following:

```
/o1003:"Edit Configuration"
>get -s security = {[f] = {password = user}}
  security =
    {
    [a] =
      {
      password = user
      accessRights =
        {
        ReadAccess = on
        WriteAccess = off
        SecurityAccess = off
        FileSystemAccess = off
        }
      }
    }
OK
```

Continued on next page

The get –s option (continued)

In case of character strings it is possible to use wildcards. The wildcard character is an asterisk: *. Suppose you want to display that line of the security table where the password ends on “admin”. In that case, type the following:

```
/o1003:"Edit Configuration"
>get -s security = {[f] = {password = *admin}}
  security =
    {
    [a] =
      {
      password = sys_admin
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = on
        FileSystemAccess = on
        }
      }
    }
OK
/o1003:"Edit Configuration"
>
```

If you want to use the wildcard at the end of a string, it is not necessary to type the wildcard character. Suppose you want to display that line of the security table where the password begins with “sys”. In that case, type the following:

```
>get -s security = {[f] = {password = sys}}
  security =
    {
    [a] =
      {
      password = sys_admin
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = on
        FileSystemAccess = on
        }
      }
    [a] =
      {
      password = sys_tech
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = off
        FileSystemAccess = on
        }
      }
    }
OK
/o1003:"Edit Configuration"
>
```

Continued on next page

The get –s option (continued)

You can of course filter out rows specifying other values than character strings. Suppose you want to display that line of the security table where the access rights are all set to *on*. In that case, type the following:

```
/o1003:"Edit Configuration"
>get -s security = {[f] = {accessRights = {ReadAccess = on  WriteAccess = on
SecurityAccess = on  FileSystemAccess = on}}}
  security =
    {
    [a] =
      {
      password = sys_admin
      accessRights =
        {
        ReadAccess = on
        WriteAccess = on
        SecurityAccess = on
        FileSystemAccess = on
        }
      }
    }
OK
/o1003:"Edit Configuration"
>
```

f) The *get –t* command

Use this option to retrieve nested tables. This means that in case you perform a `get -t` of a table in which structures and other tables (i.e. nested tables) appear, you retrieve all the values present in these structures and tables. The ordinary `get` command does not do this by default as some tables can be very complex and might take a very long time to retrieve.

The following shows you the difference between `get` and `get -t` on a table containing nested tables:

```
/pathman/paths:"Edit Configuration"
>get pathTable
  pathTable =
    {
    [a] =
      {
      name = demo
      description = demo
      sections =
        {
        }
      }
    }
OK
/pathman/paths:"Edit Configuration"
>get -t pathTable
  pathTable =
    {
    [a] =
      {
      name = demo
      description = demo
      sections =
        {
        [a] =
          {
          description = <OPT>
          endPoint1 =
            {
            device = Orchid_ShdslCvLeftModemB
            interface = g703
            bandwidth =
              {
              pdh =
                {
                ...
                }
              }
            }
          endPoint2 =
            {
            device = Orchid_ShdslCvRightModemB
            interface = v35
            bandwidth =
              {
              speed = "4 Nx64K"
              }
            }
          type = active
          }
        }
      }
    }
OK
/pathman/paths:"Edit Configuration"
>
```

g) The *get –v* option

Use this option to display the values of a table in rows and columns and separated by the value separator. The first row of the output table represents the value names.

The used value separator depends on the environment variable `VALUESEPARATOR` (refer to Section 4.9 - The environment variables). As default, it is a `<tab>`.

Suppose the exitPortNoObjects status attribute of the Orchid 1003 LAN contains the following values:

```
/o1003/nmsgroup:Status
>get exitPortNoObjects
  exitPortNoObjects =
    {
    [1] =
      {
      cardnestAddress = 0
      cardPosition = 3
      modem = A
      }
    [2] =
      {
      cardnestAddress = 0
      cardPosition = 3
      modem = B
      }
    [3] =
      {
      cardnestAddress = 0
      cardPosition = 3
      modem = C
      }
    [4] =
      {
      cardnestAddress = 0
      cardPosition = 3
      modem = D
      }
    }
OK
/o1003/nmsgroup:Status
>
```

Displaying these values with the *get –v* command results in the following:

```
/o1003/nmsgroup:Status
>get -v exitPortNoObjects
cardnestAddress cardPosition    modem
0       3       A
0       3       B
0       3       C
0       3       D

OK
/o1003/nmsgroup:Status
>
```

h) The *get –w* option

Use this option to display the values of a complex attribute value on one line.

The number of elements that will be printed on one line depends on the environment variable `COLS` (refer to Section 4.9 - The environment variables).

```
/crocusSDSLTT:"Edit Configuration"
>get -w
  sysName = "Crocus SDSL TT - No.15"
  sysContact = "System administrator - Tel. 785612"
  sysLocation = "Main building - Equipment room"
  security =
    {
    }
  bootFromFlash = Auto
  forwardTMAToNMSPort = enabled
  forwardTMAToLine = enabled
  alarmMask = {  NotResponding = enabled    AlarmSyncLoss = enabled
                 StrapChanged = disabled    Access = disabled
                 RemoteAlarm = disabled    UnknownState = disabled
                 Boot = disabled    CodeConsistencyFail = disabled
                 ConfigConsistencyFail = disabled    }
  alarmLevel = {  NotResponding = 4    AlarmSyncLoss = 4    StrapChanged = 1
                  Access = 1    RemoteAlarm = 0    UnknownState = 0
                  Boot = 1    CodeConsistencyFail = 1
                  ConfigConsistencyFail = 1    }
  alarmContactHighMask = {  NotResponding = disabled
                            AlarmSyncLoss = disabled
                            StrapChanged = disabled    Access = disabled
                            RemoteAlarm = disabled    UnknownState = disabled
                            Boot = disabled    CodeConsistencyFail = disabled
                            ConfigConsistencyFail = disabled    }
  alarmContactLowMask = {  NotResponding = disabled
                           AlarmSyncLoss = disabled    StrapChanged = disabled

                           Access = disabled    RemoteAlarm = disabled
                           UnknownState = disabled    Boot = disabled
                           CodeConsistencyFail = disabled
                           ConfigConsistencyFail = disabled    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.4 Logging your activity

Using the *logging* command, you can redirect the screen output to a file.

This can be particularly useful when you want to create scripts. You can then use these scripts to easily change identical attributes on other devices of the same type. For more information on this matter, refer to Section 6.5.9 - Setting values obtained with the get command.

Also, using this log file, importing status or performance data into a database or a spreadsheet application is very much simplified. This is shown in the following example.

Example

Suppose you want to export the values of the performance attribute h24Modem located in the modem object of the Crocus SDSL containment tree. Suppose you want to import this data in a spreadsheet, then use the *get –v* command to format the data in rows and columns. The name and location of the log file is: *C:\data\logfile.txt*.

Proceed as follows:

```
/crocusSDSLTT:"Edit Configuration"
>selgrp Performance
OK
/crocusSDSLTT:Performance
>select modem
OK
/crocusSDSLTT/modem:Performance
>logging on C:\data\logfile.txt
OK
/crocusSDSLTT/modem:Performance
>get -v h24Modem
validity|period|noSyncTime
valid|"-24h -> -22h"|"0d 2h 0m 0s"
valid|"-22h -> -20h"|"0d 2h 0m 0s"
valid|"-20h -> -18h"|"0d 2h 0m 0s"
valid|"-18h -> -16h"|"0d 2h 0m 0s"
valid|"-16h -> -14h"|"0d 2h 0m 0s"
valid|"-14h -> -12h"|"0d 2h 0m 0s"
valid|"-12h -> -10h"|"0d 2h 0m 0s"
valid|"-10h -> -8h"|"0d 2h 0m 0s"
valid|"-8h -> -6h"|"0d 2h 0m 0s"
valid|"-6h -> -4h"|"0d 2h 0m 0s"
valid|"-4h -> -2h"|"0d 2h 0m 0s"
valid|"-2h -> 0h"|"0d 2h 0m 0s"

OK
/crocusSDSLTT/modem:Performance
>logging off
```

Now you can import the log file into a spreadsheet application or database. In MS Excel, for instance, you can easily delimit the columns by specifying the value separator. In our example the value separator is a | character.

Do not uses spaces in the file name of the log file. For example: do not use `sdsl log file.txt`, use `sdsl_log_file.txt` instead.

If you do not specify a directory path in the *logging on* command (i.e. **`logging on logfile.txt`**), the log file is placed in the directory where you invoked the TMA CLI application.

6.5 Setting attribute values

This section explains how you can use the *set* command to set a simple value, a structured (or complex) value, a simple value within a structured value, etc.

The following table gives an overview of this section.

⚠ Configuration changes are only activated after you executed the *Activate Configuration* command (refer to Section 6.7 - Performing actions).

6.5.1 Setting a simple attribute value

Setting a simple attribute value

The following examples show you how to set simple attribute values.

Editing the sysName attribute value:

```
/crocusSDSLTT:"Edit Configuration"
>set sysName = "Crocus SDSL TT - central"
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Setting the bootFromFlash attribute value:

```
/crocusSDSLTT:"Edit Configuration"
>set bootFromFlash = ”Flash 1”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Setting a simple attribute value located on a lower object level

```
/crocusSDSLTT:"Edit Configuration"
>set modem/line/speed = “128000 bps”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

If you want to set a textual string value that contains characters that conflict with the syntax (such as a comma, a space, an equal sign, etc.), then you have to surround the string by a set of double quote characters: “ “. Refer to the examples above.

6.5.2 Setting a structured attribute value

This section explains how to set the values within a *bit string* and a *structure*. For the definition of *bit string* and *structure*, refer to Section 6.3.2 - Reading a structured attribute value.

Because a *table* is a quite complicated attribute value, it is explained in Section 6.5.4 - Performing actions on a table with a variable length.

Setting values in a bit string

The following examples show you how to set (a) value(s) in a bit string.

Setting only one value in the alarmMask attribute:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmMask = {RemoteAlarm = enabled}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Setting several values in the alarmMask attribute:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmMask = {StrapChanged = enabled  Access = enabled  RemoteAlarm = enabled}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Setting several values in the timeslots attribute:

```
/crocusSDSLTT/g703:"Edit Configuration"
>set timeslots = {ts0 = off  ts1 = on  ts2 = on  ts10 = on  ts25 = on}
OK
/crocusSDSLTT/g703:"Edit Configuration"
>
```

Continued on next page

Setting a structured attribute value (continued)

If you set one or more bits of a bit string to a certain value, then the remaining bits in that bit string are reset to their default value. See the example below.

Suppose all the bits in the alarmMask attribute are *enabled*:

```
>get -w alarmMask
alarmMask = {  NotResponding = enabled    AlarmSyncLoss = enabled
                 StrapChanged = enabled    Access = enabled
                 RemoteAlarm = enabled    UnknownState = enabled
                 Boot = enabled    CodeConsistencyFail = enabled
                 ConfigConsistencyFail = enabled    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

If you now set the Boot bit within the alarmMask attribute to *disabled*, then all the remaining bits are set to their default value:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmMask = {Boot = disabled}
OK
/crocusSDSLTT:"Edit Configuration"
>get -w alarmMask
  alarmMask = {  NotResponding = enabled    AlarmSyncLoss = enabled
                 StrapChanged = disabled    Access = disabled
                 RemoteAlarm = disabled    UnknownState = disabled
                 Boot = disabled    CodeConsistencyFail = disabled
                 ConfigConsistencyFail = disabled    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Setting a structured attribute value (continued)

Setting values in a structure

The following examples show you how to set (a) value(s) in a structure.

Setting only one value in the alarmLevel attribute:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmLevel = {RemoteAlarm = 2}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Setting several values in the alarmLevel attribute:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmLevel = {StrapChanged = 2  Access = 2  RemoteAlarm = 2}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.5.3 Performing actions on a table with a fixed length

A fixed table is a table with a fixed number of rows. This means you can not add or remove rows from a fixed table.

Setting a value in a row of a fixed table

The following example enters some values in the telephone table (dialler table) of an Aster 4 modem:

```
/aster4/dialler:"Edit Configuration"
>set telephoneTable =
{
    [1] = {785612}
    [10] = {784523}
    [11] = {789512}
}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Clearing an entire row of a fixed table

The following command clears the third row in the dialler table:

```
/aster4/dialler:"Edit Configuration"
>set dialler = {[c3]}
OK
/aster4/dialler:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/aster4/dialler:"Edit Configuration"
>set dialler[c3] = { }
OK
/aster4/dialler:"Edit Configuration"
>
```

Clearing an entire fixed table

The following command deletes the entire security table:

```
/aster4/dialler:"Edit Configuration"
>set dialler = {[c]}
OK
/aster4/dialler:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/aster4/dialler:"Edit Configuration"
>set dialler[c] = { }
OK
/aster4/dialler:"Edit Configuration"
>
```

6.5.4 Performing actions on a table with a variable length

A variable length table is a table with a variable number of rows. This means you can add or remove rows from a variable length table.

Using the *set* command, you can perform different actions in a variable length table depending on the option. The options are explained in the following paragraphs.

Option	Description	Refer to paragraph …
`set [a]`	Append a row at the end of the table.	a) The set [a] option, page 105
`set [i`$_x$`]`	Insert a row in a table before row number x.	b) The set [ix] option, page 106
`set [`$_x$`]`	Change a value on row number x of a table.	c) The set [x] option, page 108
`set [d`$_x$`]`	Delete row number x from a table.	d) The set [dx] option, page 111
`set [d]`	Delete all rows from a table.	e) The set [d] option, page 112

a) The *set [a]* option

Appending a row with default values

Suppose the security table of a device is empty. Now, use the *set [a]* command to create a row in the table:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[a]}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

By using the *set [a]* command without extra arguments, all the values in the newly created row have their default value:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Appending a row and setting a value at the same time

You can append a new row to a table and set a value in this row at the same time:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[a] = {password = passw4}}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

This results in the following:

```
>get -w security
  security =
    {
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

b) The *set [i_x]* option

Inserting a row with default values

The following example creates a new row before the second row in the security table:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[i2]}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
>set security[i2] = {}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

By using the *set [i_x]* command without extra arguments, all the values in the newly created row have their default value:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

The set [ix] option (continued)

Inserting a row and setting a value at the same time

You can insert a new row in a table and set a value in this row at the same time:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[i2] = {password = passw2}}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[i2] = {password = passw2}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Both notations displayed above result in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw2    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

c) The *set [x]* option

Changing one value in a row

In the security table of our example, the passwords on the first and third row are still empty. Now, use the *set [x]* command to enter a password in the first row:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[1] = {password = passw1}}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[1] = {password = passw1}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Both notations displayed above result in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = passw1    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw2    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

The set [x] option (continued)

Changing several values in a row

Suppose you want to change the currently empty password on the third row to passw3. You also want to change the corresponding access rights: you only want to assign read access to this password. In that case, type the following:

```
/crocusSDSLTT:"Edit Configuration"
>set security =
{
  [3] =
  {
    password = passw3
    accessRights =
      {
      ReadAccess = on
      WriteAccess = off
      SecurityAccess = off
      }
  }
}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[3] =
{
  password = passw3
  accessRights =
    {
    ReadAccess = on
    WriteAccess = off
    SecurityAccess = off
    }
}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Both notations displayed above result in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = passw1    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw2    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw3    accessRights = {  ReadAccess = on
             WriteAccess = off    SecurityAccess = off    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

The set [x] option (continued)

Changing the values in a row to their default value

If you now want to reset the values of the third row to their default value, then type the following:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[3] = { }}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[1] = { }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Both notations displayed above result in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = passw1    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw2    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

d) The *set [d_x]* option

The following command deletes the third row in the security table:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[d3]}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[d3] = { }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

For the security table in our example, this results in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
security =
    {
    [a] = {  password = passw1    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw2    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw4    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

e) The *set [d]* option

The following command deletes the entire security table:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[d]}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[d] = { }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

For the security table in our example, this results in the following:

```
/crocusSDSLTT:"Edit Configuration"
>get security
  security =
    {
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.5.5 Combining actions on a table with a variable length

The table actions described in Section 6.5.4 - Performing actions on a table with a variable length can be combined.

In the following example, the combination of several table actions will:

- Delete the existing security table.
- Add two new rows with respectively passwords passw1 and passw2 to the table.
- Insert a row with password passw3 between the two rows that have been created in the previous step.
- Reset the third row to its default value.

Suppose that before you execute the combined table actions, the security table looks as follows:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = xxx    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Now execute the combined table actions as discussed above:

```
/crocusSDSLTT:"Edit Configuration"
>set security =
{
  [d]
  [a] = {password = passw1}
  [a] = {password = passw2}
  [i2] = {password = passw3}
  [3] = { }
}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

The resulting security table is now:

```
/crocusSDSLTT:"Edit Configuration"
>get -w security
  security =
    {
    [a] = {  password = passw1    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = passw3    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    [a] = {  password = ""    accessRights = {  ReadAccess = on
             WriteAccess = on    SecurityAccess = on    }  }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.5.6 Setting attribute values that have a certain unit

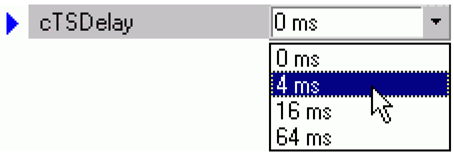
Some attribute values have a certain unit, e.g. bps, Kbps, ms, etc. Depending whether the value is an enumerated value or a real or integer value, the unit has to be specified or not. The following examples clarify this.

Setting an enumerated attribute value with unit

Because the unit of an enumerated value is part of the value field, it has to be specified.

Suppose you want to change the cTSDelay attribute of the V35 interface on a Crocus SDSL. The corresponding action in TMA is shown at the right hand side.

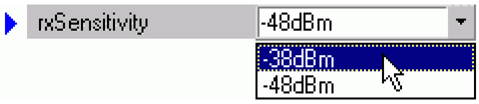
```
/crocusSDSLTT/v35:"Edit Configuration"
>set cTSDelay = "4 ms"
OK
/crocusSDSLTT/v35:"Edit Configuration"
>
```



Note that in this case the value and the unit are separated by a space, hence you have to use quotes (“ ”).

Suppose you want to change the rxSensitivity attribute on an Aster 5. The corresponding action in TMA is shown at the right hand side.

```
/aster5/line/pstn:"Edit Configuration"
>set rxSensitivity = -38dBm
OK
/aster5/line/pstn:"Edit Configuration"
>
```

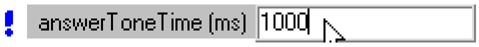


Setting a real or integer attribute value with unit

Because the unit of a real or integer value is not a part of the value field, it does not have to be specified.

Suppose you want to change the answerToneTime attribute on an Aster 5. The corresponding action in TMA is shown at the right hand side.

```
/aster5/line/pstn:"Edit Configuration"
>set answerToneTime = 1000
OK
/aster5/line/pstn:"Edit Configuration"
>
```



Note that although in the GUI of TMA the unit is displayed after the attribute name, you do not have to type this unit in TMA CLI. In other words …

- type `set answerToneTime = …`
- do not type `set answerToneTime (ms) = …` nor `set "answerToneTime (ms)" = …`

Continued on next page

Setting attribute values that have a certain unit (continued)

Setting a time value

Because the units of a time value are part of the value field, they have to be specified. However, you do not always have to specify the complete value.

Suppose you want to set the consoleNoTrafficTimeOut attribute to …

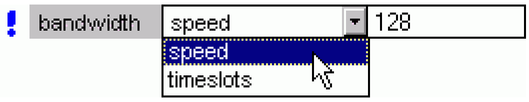
- 12345 days, 12 hours, 34 minutes and 56 seconds, then type:
  `>set consoleNoTrafficTimeOut = "12345d 12h 34m 56s"`
- 5 days, 2 hours, 4 minutes and 6 seconds, then type
  `>set consoleNoTrafficTimeOut = "00005d 02h 04m 06s"` or
  `>set consoleNoTrafficTimeOut = "5d 2h 4m 6s"`
- 2 hours, 4 minutes and 6 seconds, then type:
  `>set consoleNoTrafficTimeOut = "00000d 02h 04m 06s"` or
  `>set consoleNoTrafficTimeOut = "02h 04m 06s"` or
  `>set consoleNoTrafficTimeOut = "2h 4m 6s"`
- 55 seconds, then type:
  `>set consoleNoTrafficTimeOut = "00000d 00h 00m 55s"` or
  `>set consoleNoTrafficTimeOut = "55s"`
- 12 hours, then type:
  `>set consoleNoTrafficTimeOut = "00000d 12h 00m 00s"` or
  `>set consoleNoTrafficTimeOut = "12h 00m 00s"` or
  `>set consoleNoTrafficTimeOut = "12h"`
- 6 days and 55 seconds, then type:
  `>set consoleNoTrafficTimeOut = "00006d 00h 00m 55s"` or
  `>set consoleNoTrafficTimeOut = "6d 00h 00m 55s"` or
  `>set consoleNoTrafficTimeOut = "6d 55s"`
- etc.

6.5.7 Setting a choice value

A choice value is a value which has two or more possible *value types*. In other words, you first have to make a “pre-selection” which determines the value type (e.g. direct editable, enumerated, bit string, structured value, etc.) and then you can edit the value itself. The following example clarifies this.

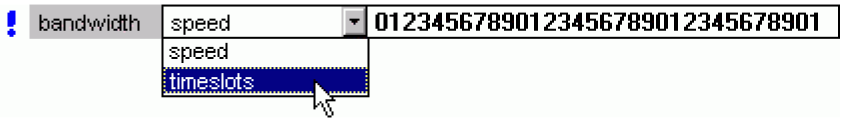
Suppose you have an attribute bandwidth for which, depending on the application, you have to specify a speed or timeslots. So, you first make a selection between the choice value speed or timeslots and then you enter the value you want (the corresponding action in TMA is shown under each CLI string):

- `>set bandwidth = {speed = 128}`



bandwidth speed 128

- `>set bandwidth = {pdh = {ts0 = 1  ts1 = 1  ...  ts31 = 1  ts32 = 1}}`



bandwidth speed 01234567890123456789012345678901

6.5.8 Setting several attribute values at once

Setting several attribute values on the current object level

In order to set several attribute values at once, you have to group the attributes using *LIST { … }*. This is shown in the following example:

```
/crocusSDSLTT:"Edit Configuration"
>set
  {
  LIST
    {
    sysName = "Crocus SDSL TT - central"
    sysContact = "System Administrator - Tel. 785612"
    sysLocation = "Main Building - Equipment Room"
    security =
      {
      [a] =
        {
        password = "sys_admin"
        }
      [a] =
        {
        password = "user"
        accessRights =
          {
          ReadAccess = on
          WriteAccess = off
          SecurityAccess = off
          }
        }
      }
    }
  }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Setting several attribute values at once (continued)

Setting several attribute values on another object level

You are also able to set several attribute values at once on another object level than the current one. In that case you have to use the *SELECT* command to move to the other object. The attributes within *SELECT* have to be surrounded by *LIST { … }*. This is shown in the following example:

```
/crocusSDSLTT:"Edit Configuration"
>set
  {
  SELECT modem
    {
    LIST
      {
      tests =
        {
        detectRDL = disabled
        }
      channel = central
      }
    SELECT line
      {
      LIST
        {
        speed = "128000 bps"
        }
      }
    }
  }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Setting several attribute values at once (continued)

Setting several attribute values on the current and another object level

Of course, combining the two previous examples allows you set several attributes on the current and other object levels at once. This is shown in the following example:

```
/crocusSDSLTT:"Edit Configuration"
>set
  {
  LIST
    {
    sysName = "Crocus SDSL TT - central"
    sysContact = "System Administrator - Tel. 785612"
    sysLocation = "Main Building - Equipment Room"
    security =
      {
      [a] =
        {
        password = "sys_admin"
        }
      [a] =
        {
        password = "user"
        accessRights =
          {
          ReadAccess = on
          WriteAccess = off
          SecurityAccess = off
          }
        }
      }
    }
  SELECT modem
    {
    LIST
      {
      tests =
        {
        detectRDL = disabled
        }
      channel = central
      }
    SELECT line
      {
      LIST
        {
        speed = "128000 bps"
        }
      }
    }
  }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.5.9 Setting values obtained with the *get* command

As said in several previous sections, you can use the *logging* command together with the *get* command in order to create scripts:

- Paragraph a) Creating a script file starting from a non-configured device, explains how you can set the attributes of a device by logging the default configuration to a script file, editing this file and then executing this script on the device.
- Paragraph b) Creating a script file starting from a configured device, explains how you can easily set identical attributes on other devices of the same type by first changing the configuration of one device, logging this configuration to a script file and then executing this script on several devices.

The script files that you create may have any extension (**.cli*, **.txt*, etc.). However, in order to distinguish your TMA CLI script files from other files, it may be useful to use the extension **.cli*.

a) Creating a script file starting from a non-configured device

Suppose you have a Crocus SDSL TT that has not been configured yet, i.e. it still has its default configuration. Suppose you want to set the following attributes on this Crocus SDSL TT:

- crocusSDSLTT/sysName = “Crocus SDSL TT – central”
- crocusSDSLTT/sysContact = “Sys Admin – Tel. 785612”
- crocusSDSLTT/sysLoaction = “Equipment Room”
- crocusSDSLTT/bootFromFlash = “Flash 1”
- crocusSDSLTT/modem/tests/detectRDL = disabled
- crocusSDSLTT/modem/channel = central
- crocusSDSLTT/modem/line/speed = “128000 bps”

Step 1

First open a TMA CLI session on the Crocus SDSL TT. Then log its full configuration:

```
/crocusSDSLTT:"Edit Configuration"
>logging on script.cli
OK
/crocusSDSLTT:"Edit Configuration"
>get -r
...
OK
/crocusSDSLTT:"Edit Configuration"
>logging off
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Creating a script file starting from a non-configured device (continued)

Step 2

Now, open the *script.cli* file in a text editor and edit it as follows:

- Remove the command prompt at the beginning and the end of the file.
- Remove the *>get –r* command at the beginning of the file.
- Replace the *GET* command at the beginning of the file by the *SET* command.
- Remove all attributes you do not want to change.
- Edit the attributes you want to change.

The following table shows you the non-edited script file and the edited script file side by side:

<table>
<tr><th>Non-edited script file</th><th>Edited script file</th></tr>
<tr><td>

```
/crocusSDSLTT:"Edit Configuration"
>get -r
  GET
    {
    LIST
      {
      sysName = ""
      sysContact = ""
      sysLocation = ""
      ...
      }
    SELECT modem
      {
      LIST
        {
        ...
        }
      SELECT line
        {
        LIST
          {
          ...
          }
        }
      }
    SELECT nMS
      {
      LIST
        {
        cms2Address = 0
        }
      }
    SELECT powerOffDetection
      {
      LIST
        {
        ...
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>logging off
```

</td><td>

```
SET
  {
  LIST
    {
    sysName = "Crocus SDSL TT - central"
    sysContact = "Sys Admin - Tel. 785612"
    sysLocation = "Equipment Room"
    bootFromFlash = "Flash 1"
    }
  SELECT modem
    {
    LIST
      {
      tests =
        {
        detectRDL = disabled
        }
      channel = central
      }
    SELECT line
      {
      LIST
        {
        speed = "128000 bps"
        }
      }
    }
  }
```

</td></tr>
</table>

Continued on next page

Creating a script file starting from a non-configured device (continued)

Step 3

Run the script on the Crocus SDSL TT using the *execute* command:

```
/crocusSDSLTT:"Edit Configuration"
>exec script.cli
...
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Step 4

Activate the new configuration using the *"Activate Configuration"* action:

```
/crocusSDSLTT:"Edit Configuration"
>action "Activate Configuration"
OK
/crocusSDSLTT:"Edit Configuration"
>
```

b) Creating a script file starting from a configured device

Suppose you have several Crocus SDSL TTs that have to be configured. Suppose that on all these Crocus SDSL TTs you want to set the following attributes:

- crocusSDSLTT/sysContact = “Admin – Tel. 785612”
- crocusSDSLTT/sysLoaction = “Equipment Room”
- crocusSDSLTT/bootFromFlash = “Flash 2”
- crocusSDSLTT/security/password = “sys_admin”
- crocusSDSLTT/modem/tests/detectRDL = disabled
- crocusSDSLTT/modem/channel = central
- crocusSDSLTT/modem/line/speed = “512000 bps”

Step 1

First open a TMA or TMA CLI session on one of the Crocus SDSL TTs. Set the attributes listed above to the wanted values, then activate this configuration.

Step 2

Now, log the non-default configuration of the Crocus SDSL TT:

```
/crocusSDSLTT:"Edit Configuration"
>logging on script.cli
OK
/crocusSDSLTT:"Edit Configuration"
>get –f
...
OK
/crocusSDSLTT:"Edit Configuration"
>logging off
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Using the *get –f* option results in a log file that only contains those values that differ from the default values. This results in a smaller, easier to edit file. It also saves bandwidth since you only upload a limited number of attributes towards the different devices.

Continued on next page

Creating a script file starting from a configured device (continued)

Step 3

Now, open the *script.cli* file in a text editor and check it. Because you used the *get –f* command, this file contains:

- The non-default configuration (i.e. only those values that differ from the default values).
- The action *“Load Default Configuration”* at the beginning of the file.
- The action *“Activate Configuration”* at the end of the file.

```
action "Load Default Configuration"
SET
  {
  LIST
    {
    sysContact = "Admin - Tel. 785612"
    sysLocation = "Equipment Room"
    security =
      {
      [a] =
        {
        password = sys_admin
        }
      }
    bootFromFlash = "Flash 2"
    }
  SELECT modem
    {
    LIST
      {
      tests =
        {
        detectRDL = disabled
        }
      channel = central
      }
    SELECT line
      {
      LIST
        {
        speed = "512000 bps"
        }
      }
    }
  SELECT nMS
    {
    }
  SELECT powerOffDetection
    {
    }
  }
action "Activate Configuration"
```

Continued on next page

Creating a script file starting from a configured device (continued)

Step 4

Run the script on the different Crocus SDSL TTs using the *execute* command. You can do this either in interactive or non-interactive mode:

Mode	Script execution

interactive mode	Open a TMA CLI session on the Crocus SDSL TT and execute the *script.cli* file:
	`/crocusSDSLTT:"Edit Configuration"`
	`>`**`exec script.cli`**
	`...`
	`OK`
	`/crocusSDSLTT:"Edit Configuration"`
	`>`**`disconnect`**

non-interactive mode	Execute the *script.cli* file from the DOS / shell prompt:
	`TmaCli <address_arguments> -exec script.cli`
	Examples:
	`TmaCli com1 -exec script.cli`
	`TmaCli 10.0.11.1 -exec script.cli`
	…

6.5.10 Setting an attribute value to its default value

In the following examples, there is no space between the two consecutive curled brackets: {}.

Resetting a simple value, a structured value or a table

The following command resets the attribute sysName to its default value:

```
/crocusSDSLTT:"Edit Configuration"
>set sysName = {}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

The following command resets the attribute alarmLevel to its default value:

```
/crocusSDSLTT:"Edit Configuration"
>set alarmLevel = {}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

The following command resets the telephoneTable of the Aster 4 to its default value:

```
/aster4/dialler:"Edit Configuration"
>set telephoneTable = {}
OK
/aster4/dialler:"Edit Configuration"
>
```

The following command resets the security table to its default value:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Resetting only one row of a table

It is possible to reset only one row of a table. Suppose you want to reset row 2 of the security table to its default value, then enter the following command:

```
/crocusSDSLTT:"Edit Configuration"
>set security = {[2] = {}}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Alternatively, you can use the following syntax:

```
/crocusSDSLTT:"Edit Configuration"
>set security[2] = {}
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Continued on next page

Setting an attribute value to its default value (continued)

Resetting an entire object

The following command resets all the attribute values in the currently selected object to their default value:

```
/crocusSDSLTT:"Edit Configuration"
>set { LIST {} }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Resetting an entire device

You can reset all the attribute values in the entire containment tree of a device by loading the default configuration:

```
/crocusSDSLTT:"Edit Configuration"
>action “Load Default Configuration”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

6.6 Adding a user instantiatable object

This section explains what user instantiatable objects are. It also explains how to add and remove user instantiatable objects.

The following table gives an overview of this section.

6.6.1 What is a user instantiatable object?

On some device (e.g. the Telindus 1421 SHDSL Router) some objects are not present in the containment tree by default. If you want to use the features associated with such an object, then you have to add the object first. An object that can be added by the user is called a *user instantiatable object*. Also referred to as a *child object* because it is added under a *parent object*.

6.6.2 How to add a user instantiatable object?

Add a user instantiatable object to the containment tree using the `set / select` command.

Adding a user instantiatable object which has no index

The following command adds the user instantiatable object proxy under the top object telindus1031. The proxy object has no index. This because you can only add one proxy object to the containment tree.

```
/telindus1031:"Edit Configuration"
>set {select proxy{}}
OK
/telindus1031:"Edit Configuration"
>
```

The curled brackets {} behind the user instantiatable object name have to be present! This because TMA CLI expects curled brackets after a `select` command within a `set` command. If you want, you can insert more select commands or a list of attributes between these curled brackets.

Adding a user instantiatable object which has an index

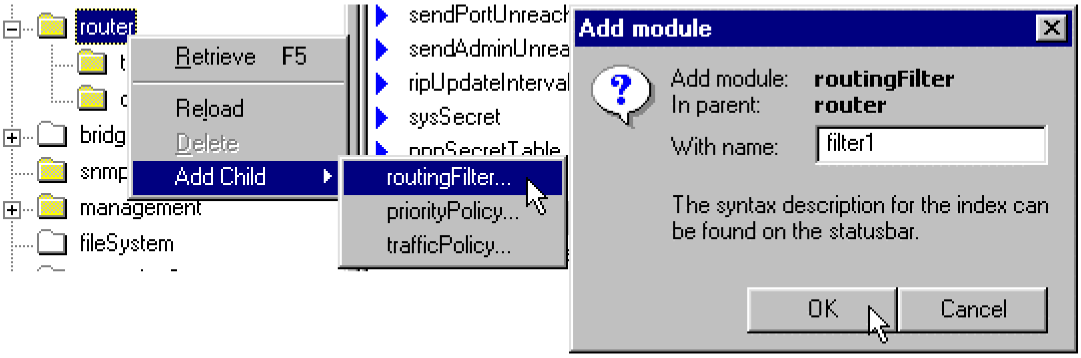
The following command adds the user instantiatable object routingFilter under the object router. You also have to specify an index for the routingFilter object (e.g. filter1). This because you can add several routingFilter objects to the containment tree. The index distinguishes the different routingFilter objects from on another.

```
/telindus1031/router:"Edit Configuration"
>set {select routingFilter[filter1]{}}
OK
/telindus1031/router:"Edit Configuration"
>
```

You can now enter the added object:

```
/telindus1031/router:"Edit Configuration"
>select routingFilter[filter1]
OK
/telindus1031/router/routingFilter[filter1]:"Edit Configuration"
>
```

The corresponding action in TMA looks as follows:

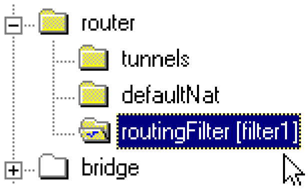


Add module

Add module: **routingFilter**
In parent: **router**
With name: filter1

The syntax description for the index can be found on the statusbar.

The corresponding result in TMA looks as follows:



6.6.3 How to remove a user instantiatable object?

Remove a previously added user instantiatable object from the containment tree using the `set / delobj` command.

Removing a user instantiatable object which has no index

The following command removes the user instantiatable object proxy under the top object telindus1031.

```
/telindus1031:"Edit Configuration"
>set {delobj proxy}
OK
/telindus1031:"Edit Configuration"
>
```

Whereas the combination `set / select` expects curled brackets {} behind the user instantiatable object name (refer to Section 6.6.2 - How to add a user instantiatable object?), the combination `set / delobj` does not.

Removing a user instantiatable object which has an index

The following command removes the user instantiatable object routingFilter[filter1] under the object router.

```
/telindus1031/router:"Edit Configuration"
>set {delobj routingFilter[filter1]}
OK
/telindus1031/router:"Edit Configuration"
>
```

6.7 Performing actions

You can initiate actions on Telindus devices using the *actions* command. An object in combination with a group may have actions assigned to them. The available actions depend on the kind of device.

Some examples of actions are: performing a cold boot, activation of a configuration, performing a test, etc.

Starting an action that has no argument

Use the following command to perform a cold boot on a device:

```
/crocusSDSLTT:"Edit Configuration"
>action “Cold Boot”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Use the following command to activate a configuration you previously entered or loaded:

```
/crocusSDSLTT:"Edit Configuration"
>action “Activate Configuration”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Starting an action that has a simple argument

Actions such as activation of a test need arguments. The following example shows you how to start an AL test on a modem:

```
/crocusSDSLTT/modem:Performance
>action “Test Activation” testActivationType = “AL test”
OK
/crocusSDSLTT/modem:Performance
>
```

In order to stop a test that is currently running, use the *“Test Activation”* action with the following argument:

```
/crocusSDSLTT/modem:Performance
>action “Test Activation” testActivationType = “NO test”
OK
/crocusSDSLTT/modem:Performance
>
```

Starting an action that has a structured argument

Let us consider the startPing action in the router object / Performance group of the Orchid 1003 LAN. The argument of this action has several sub values: ipAddress, iterations, interval, dataLength and timeOut.

Suppose you want to perform a ping to IP address 172.31.40.10 and this 30 times. The rest of the arguments may keep their default value. In that case, type:

```
/o1003/router:Performance
>action startPing pingData = {ipAddress = 172.31.40.10  iterations = 30}
OK
/o1003/router:Performance
>
```

6.8 Scripting

A script file is an ASCII text file containing a series of commands which are executed one after the other. A script file may have any extension (**.cli*, **.txt*, etc.). Scripting is mostly used in non-interactive TMA CLI mode.

Scripting is very useful to, for instance, configure a lot of devices at once or to regularly retrieve status and performance information from all the access devices in the network. The following example will give you an idea of how you can use scripting.

A scripting example

Suppose you want to retrieve some statistics from several modems in your network. The output has to be logged to the file *stats.txt* every weekday at 8:00 pm.

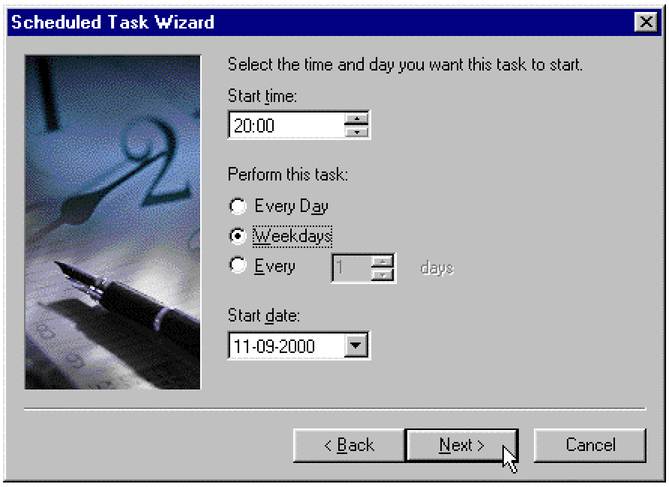
The TMA CLI script file *script.cli* may contain something like this:

```
logging on stats.txt
get sysName
get modem:Performance
get modem/line:Performance
logging off
```

Suppose you create a batch file *stats.bat* for this purpose. The batch file may contain something like this:

```
tmacli 172.31.5.1 -exec script.cli
tmacli 172.31.5.2 -exec script.cli
tmacli 172.31.5.5 -exec script.cli
tmacli 172.31.5.10 -exec script.cli
```

If you are working on a machine running Windows 95, you could use the Scheduled Task Wizard to define when the batch file should be executed:



Scheduled Task Wizard

Select the time and day you want this task to start.

Start time: 20:00

Perform this task:

○ Every Day
● Weekdays
○ Every 1 days

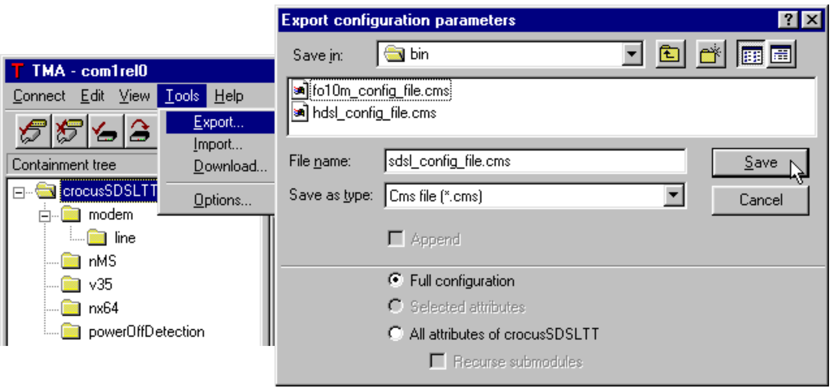
Start date: 11-09-2000

6.9 Importing a configuration file

Section 6.5.9 - Setting values obtained with the get command explained how you can easily create script files in order to set identical attributes on several devices of the same type. There is also another way to get the same result.

This involves exporting a configuration file in TMA (i.e. TMA with the GUI). The configuration file should be formatted in the binary CMS2 format. Such kind of file has the extension *.cms. In TMA CLI, you can then load the previously stored configuration using the *cfgload* command.

Suppose you have several Crocus SDSL TTs that have to be configured and you want to do this using a CMS2 configuration file. In that case, proceed as follows:

Step	Action
1	First open a TMA session on one of the Crocus SDSL TTs. Configure this Crocus SDSL TT to your needs, then activate this configuration.
2	In TMA choose *Tools* → *Export…* and save the configuration file in the *.cms format:  TMA - com1rel0 Connect Edit View Tools Help Export... Import... Download... Options... Containment tree crocusSDSLTT modem line nMS v35 nx64 powerOffDetection Export configuration parameters Save Cancel Save in: bin fo10m_config_file.cms hdsl_config_file.cms File name: sdsl_config_file.cms Save as type: Cms file (*.cms) □ Append ● Full configuration ○ Selected attributes ○ All attributes of crocusSDSLTT □ Recurse submodules

3	Now, open a TMA CLI session on another Crocus SDSL TT. Load the previously stored configuration using the *cfgload* command:
	`/crocusSDSLTT:"Edit Configuration"`
	`>`**`cfgload C:\data\sdsl_config_file.cms`**
	`OK`
	`/crocusSDSLTT:"Edit Configuration"`
	`>`

4	Activate the new configuration using the *“Activate Configuration”* action:
	`/crocusSDSLTT:"Edit Configuration"`
	`>`**`action “Activate Configuration”`**
	`OK`
	`/crocusSDSLTT:"Edit Configuration"`
	`>`

Do not uses spaces in the file name of the configuration file. For example: do not use `sdsl config file.txt`, use `sdsl_config_file.txt` instead.

If you do not specify a directory path in the *cfgload* command (i.e. **`cfgload sdsl_config.cms`**), it is assumed that the configuration file is present in the directory where you invoked the TMA CLI application.

6.10 Downloading files

Using the *memload* command you can download new firmware to the flash banks of an IP device. You can also download other files (firmware files of modems, configuration files, model files, etc.) to the file system of an IP device. These files can then be used for software consistency, configuration consistency or are just necessary to establish certain connections.

The following table gives an overview of this section:

You can only download files if you are connected via an IP connection to an IP device, not via a serial connection.

6.10.1 Downloading new firmware to an IP device

If you want to update the firmware of an IP device, then type the following command:

```
memload Txxxxxxx.00@CONTROL
```

where,

- `Txxxxxxx` is the new firmware file that is present on your computer,
- `CONTROL` is the destination.

The destination possibilities are:

- CONTROL: downloads the firmware into the non-active flash bank
- CONTROL1: downloads the firmware into flash bank 1
- CONTROL2: downloads the firmware into flash bank 2

You can only download new firmware into a non-active flash memory bank.

Example:

```
/o1003:"Edit Configuration"
>memload T1042017.00@CONTROL
OK
/o1003:"Edit Configuration"
>
```

6.10.2 Downloading files to the file system of an IP device

You can download several files towards the file system of the an IP device. For example:

- a configuration file of a certain device in order to distribute this configuration towards several devices of the same type,
- a configuration file of a specific device in order to perform configuration consistency on this device,
- a firmware file of a certain device in order to distribute this firmware towards several devices of the same type,
- the *models.nms* file,
- etc.

Downloading a firmware file of a device

Suppose you want to download a firmware file of a Crocus SDSL TT to the file system of the Orchid 1003 LAN. This, for instance, to perform software consistency. In that case, type the following:

```
/o1003:"Edit Configuration"
>memload T2122013.00@T2122013.00
OK
/o1003:"Edit Configuration"
>
```

In this command the part …

- before the @ is the source file name on your system
- behind the @ is the destination file name on the file system of the Orchid.

Downloading a configuration file of a device

Suppose you want to download a previously exported configuration file of the Crocus SDSL TT to the file system of the Orchid 1003 LAN. This, for instance, to perform configuration consistency. In that case, type the following:

```
/o1003:"Edit Configuration"
>memload sdsl_config_file.cms@sdsl_config_file.cms
OK
/o1003:"Edit Configuration"
>
```

6.10.3 Downloading new firmware to a non-IP device

As said before, using the *memload* command it is only possible to download firmware to an IP device. If you want to download new firmware to a non-IP device, you have to use the software distribution feature of one of the management concentrators. The following example gives you an idea how this works.

Example:

Suppose you have a Crocus SDSL TT somewhere in your network. It is connected to an Orchid 1003 LAN. The Orchid has IP address 10.0.11.1. The Crocus SDSL TT its name in the objectTable of the Orchid is “SDSL TT central”. The new firmware file for the Crocus SDSL TT is T2122013.00 and is located in the following directory on your machine: *C:\windows\temp*.

In order to download the new firmware to the Crocus SDSL TT, proceed as follows:

Step	Action
1	Open a TMA CLI session on the Orchid 1003 LAN: **`TmaCli 10.0.11.1`**

2	Download the Crocus SDSL TT firmware file located on your machine to the file system of the Orchid 1003 LAN:
	`/o1003:"Edit Configuration"`
	`>`**`memload C:\windows\temp\T2122013.00@T2122013.00`**
	`OK`
	`/o1003:"Edit Configuration"`
	`>`

3	Now, configure the downloadTable of the Orchid 1003 LAN as follows:
	`/o1003/deviceSoftware:"Edit Configuration"`
	`>`**`get downloadTable =`**
	`{`
	`[a] =`
	`  {`
	`  name = "SDSL TT central"`
	`  sourceFile = T2122013.00`
	`  destinationFile = CONTROL`
	`  immediateActivation = disabled`
	`  }`
	`}`
	`OK`
	`/o1003/deviceSoftware:"Edit Configuration"`
	`>`
	Activate this new configuration using the *“Activate Configuration”* action.

4	Now, you can start downloading the new firmware to the Crocus SDSL TT:
	`/o1003/deviceSoftware:"Edit Configuration"`
	`>`**`action “Start Download”`**
	`OK`
	`/o1003/deviceSoftware:"Edit Configuration"`
	`>`

7. User defining values using the *custom.txt* file

This chapter explains how you can user define values for attributes that have an integer as value. You can do this using the *custom.txt* file.

The following table gives an overview of this chapter.

7.1 What is user defining values?

What is user defining values?

The advantage of user defining values is that you are able to give a sensible name to an otherwise very cryptic integer value.

You can only user define values for integer attribute values. Not for enumerated values, IP addresses, etc. Examples of integer values are the values of the attributes cms2Address, alarmFilter, broadcastTimer, etc.

How to user define values?

In order to user define values, you have to create and edit a *custom.txt* file (this file is not created when TMA is installed).

What is the *custom.txt* file?

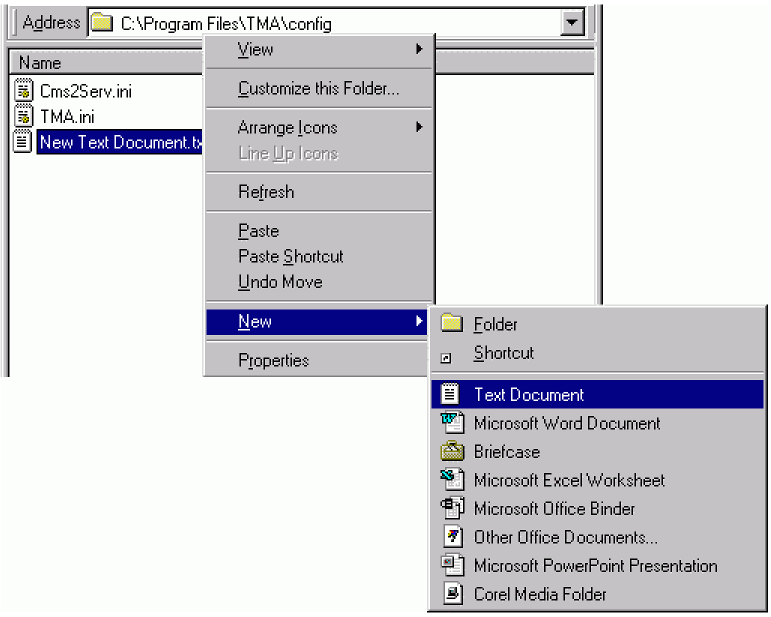
The *custom.txt* file is a plain text file containing special syntax which defines the values you want. This *custom.txt* file is loaded and parsed when TMA CLI starts.

What is the location of the *custom.txt* file?

You have to place the *custom.txt* file in the *TMA\config* directory (typically *C:\Program Files\TMA\config*).

7.2 Creating the *custom.txt* file

In order to create the *custom.txt* file, proceed as follows:

Step	Action
1	Go to the directory *TMA\config* (typically *C:\Program Files\TMA\config*).
2	Create a new (plain) text document.  Address C:\Program Files\TMA\config Name Cms2Serv.ini TMA.ini New Text Document.tx View Customize this Folder... Arrange Icons Line Up Icons Refresh Paste Paste Shortcut Undo Move New Properties Folder Shortcut Text Document Microsoft Word Document Briefcase Microsoft Excel Worksheet Microsoft Office Binder Other Office Documents... Microsoft PowerPoint Presentation Corel Media Folder
3	Rename this text document to *custom.txt*.
4	Open the *custom.txt* file and edit it using the syntax as described in the following sections.
5	Save the *custom.txt* file and start TMA CLI.

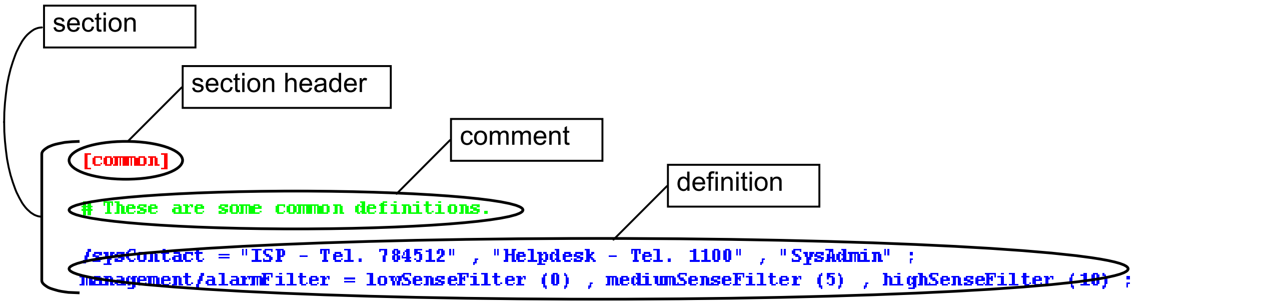
7.3 The structure of the *custom.txt* file

This section explains the general structure of the *custom.txt* file. There are three main elements that make up the *custom.txt* file:

Element	Description
section	The file is divided into a number of sections. Each section consists of a *section header* followed by a number of *definitions*. For more information on sections, refer to Section 7.4 - The sections of the custom.txt file.
definition	One section can contain one or several definitions. With these definitions you create the user defined values. For more information on definitions, refer to Section 7.5 - The definitions of the custom.txt file.
comment	Everything that is typed after a "#" character is considered as comment (until the end of the line). This allows you to add personal comments to clarify the *custom.txt* file.

Example

The following figure shows an example of the structure of a *custom.txt* file:



```
[common]

# These are some common definitions.

/sysContact = "ISP - Tel. 784512" , "Helpdesk - Tel. 1100" , "SysAdmin" ;
management/alarmFilter = lowSenseFilter (0) , mediumSenseFilter (5) , highSenseFilter (10) ;


[o1003]

nmsgroup/objectTable.name = "HDSL CV - Central x" , "HDSL TT - Central x" , "HDSL TT - Remote x" ;
nmsgroup/objectTable.configFileName = "HDSLCVCentral.cms" , "HDSLTTCentral.cms" , "HDSLTTRemote.cms" ;
deviceSoftware/downloadTable.name = "HDSL CV - Central x" , "HDSL TT - Central x" , "HDSL TT - Remote x";
deviceSoftware/downloadTable.sourceFile = "T1234567.00" , "T9876543.00" , "T9638527.00" ;


[crocusHDSLTT]

# The definition of the LinkDown alarm applies on both lines.
# The definition of the SeverelyErroredSecond alarm applies on line 1 only.

line/alarmLevel.LinkDown = "Priority = low" (0) , "Priority = medium" (5) , "Priority = high" (10) ;
/modem/line[1]/alarmLevel.SeverelyErroredSecond = low (0) , medium (5) , high (10) ;
```

7.4 The sections of the *custom.txt* file

This section gives more detailed information on the sections of the *custom.txt* file.

What is a section?

The *custom.txt* file is divided into a number of sections. Each section consists of a *section header* followed by a number of *definitions*:

```
<section> = <section_header> { <defintition> }
```

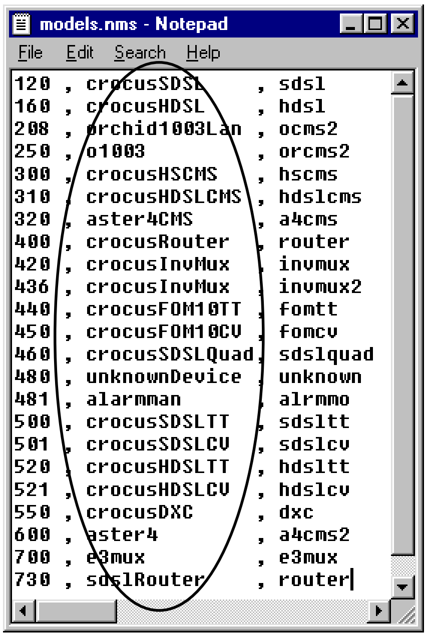
Section header syntax

The syntax for the section header is:

```
<section_header> = "[" IDENTIFIER "]"
```

Section types

There are two types of sections:

<table>
<tr><th>Section type</th><th>Description</th></tr>
<tr><td>common</td><td>This is a common section. Hence, the name of this section is <code>"common"</code>.
All definitions in this section apply (if possible) to all devices.</td></tr>
<tr><td>specific</td><td>This is a device specific section. The name of this section is the name of the top object in the containment tree of the device.
All definitions in this section apply to one specific device only.
The top object name can also be found in the second column of the <i>model.nms</i> file. E.g. for a Crocus HDSL TT this is <code>crocusHDSLTT</code>, for a Crocus DXC this is <code>crocusDXC</code>, etc.

<pre>
120 , crocusSDSL   , sdsl
160 , crocusHDSL   , hdsl
208 , orchid1003Lan , ocms2
250 , o1003   , orcms2
300 , crocusHSCMS   , hscms
310 , crocusHDSLCMS , hdslcms
320 , aster4CMS   , a4cms
400 , crocusRouter   , router
420 , crocusInvMux   , invmux
436 , crocusInvMux   , invmux2
440 , crocusFOM10TT , fomtt
450 , crocusFOM10CV , fomcv
460 , crocusSDSLQuad , sdslquad
480 , unknownDevice , unknown
481 , alarmman   , alrmmo
500 , crocusSDSLTT   , sdsltt
501 , crocusSDSLCV   , sdslcv
520 , crocusHDSLTT   , hdsltt
521 , crocusHDSLCV   , hdslcv
550 , crocusDXC   , dxc
600 , aster4   , a4cms2
700 , e3mux   , e3mux
730 , sdslRouter   , router
</pre>
</td></tr>
</table>

7.5 The definitions of the *custom.txt* file

This section gives more detailed information on the definitions of the *custom.txt* file.

What is a definition?

One section can contain one or several definitions. With these definitions you create the user defined values.

Definition syntax

A definition has the following global syntax:

```
<definition> = <attribute_specification> "=" <value_list> ";"
```

with `<attribute_specification> = <path> <attribute_name> { "." <element_name> }`

Definition description

A definition can span multiple lines. The end of the definition is marked with a `";"` character. The other elements of the definition are listed below:

Element	For more information refer to Section …
`<path>`	7.5.1 - The path, page 146.

`<attribute_name>`
`<element_name>`
	7.5.2 - The attribute and element name, page 149.

`<value_list>`	7.5.3 - The value list, page 150.

7.5.1 The path

What is a path?

A path specifies the way *to reach* a certain attribute or element through the containment tree of a device.

Path syntax

A path has the following syntax:

```
<absolute_path> = "/" { <object_name_index> "/" }
<relative_path> = <object_name> "/"
```

with

```
<object_name_index> = IDENTIFIER [ "[" NUM | IDENTIFIER "]" ]
<object_name> = IDENTIFIER;
```

Path types

There are two types of paths:

Path	Description
absolute	An absolute path starts with a `"/"` character and specifies the full path to an object (and so to an attribute or element). This means the definition applies to one specific object of the containment tree. **Example** `/modem/line[1]/alarmLevel.LinkDown` In this example, the definition *only* applies to the LinkDown element of the alarmLevel attribute in the object line[1]. The definition does *not* apply to the LinkDown element in the object line[2] or line[3].
relative	A relative path starts with the name of an object followed by a `"/"` character (and followed by another object or an attribute or element). This means the definition applies to every object in the containment tree with this name and containing the attribute or element. **Example** `line/alarmLevel.LinkDown` In this example, the definition applies to the LinkDown element of the alarmLevel attribute in all line objects. I.e. the objects line[1], line[2] and line[3].

Continued on next page

The path (continued)

Indexed objects

As you can see in the examples above, some objects in the containment tree are indexed (e.g. the objects line[1], line[2] and line[3] of a 3 pair Crocus HDSL).

As mentioned before, the following applies to indexed objects:

If you want a definition to apply …	**then …**
to one of the indexed objects specifically,	also specify the index number in the path. E.g. `/modem/line[1]/alarmLevel.LinkDown`
to all indexed objects,	just omit the index number in the path. E.g. `/modem/line/alarmLevel.LinkDown`

Remarks

- Omitting the index number of indexed objects is possible for both absolute and relative paths.
- Including the index number of indexed objects is *only* possible for absolute paths. In other words, relative paths with indexed objects are *not allowed*.
  Example:
  - `/modem/line[1]/alarmLevel.LinkDown` → ALLOWED
  - `line[1]/alarmLevel.LinkDown` → NOT ALLOWED

Continued on next page

The path (continued)

Path priority rules

You may have written more than one definition with paths that all refer to the same attribute (or element of an attribute). Moreover, these definitions may appear in the common section *and* in a device specific section.

For example, `/modem/line[1]/alarmLevel.LinkDown`, `/modem/line/alarmLevel.LinkDown` and `line/alarmLevel.LinkDown` all apply to the LinkDown element of the alarmLevel attribute.

Therefore, priorities are assigned to the different paths. These are based on three criteria:

<table>
<tr><th>Criteria</th><th>Description</th></tr>
<tr><td>Path type</td><td>The path type priorities are as follows:
<table>
<tr><th>Priority</th><th>Path</th></tr>
<tr><td>highest</td><td>Absolute path, indexed.
Example: <code>/modem/line[1]/alarmLevel.LinkDown</code></td></tr>
<tr><td>medium</td><td>Absolute path, non-indexed.
Example: <code>/modem/line/alarmLevel.LinkDown</code></td></tr>
<tr><td>lowest</td><td>Relative path (only non-indexed allowed).
Example: <code>line/alarmLevel.LinkDown</code></td></tr>
</table></td></tr>
<tr><td>Presence of indexed objects</td><td>In case two or more absolute paths containing one or more indexed objects exist, then the path that contains the indexed object with the highest location in the containment tree has the highest priority.
Example: <code>/quadE1[1]/g703[1]/alarmLevel.linkDown</code> has a higher priority than <code>/quadE1/g703[1]/alarmLevel.linkDown</code></td></tr>
<tr><td>Located in which section?</td><td>Definitions located in specific device sections have a higher priority than definitions located in the common section.</td></tr>
</table>

7.5.2 The attribute and element name

Besides the path, you also have to specify the attribute or element of an attribute in the definition.

Attribute and element syntax

Attributes are separated from the path by a `"/"` character. Elements of an attribute are separated from the attribute (and each other) by a `"."` character.

Examples

Suppose you have an Orchid 1003 LAN and you want to user define values for the attributes sysContact, alarmFilter and the element portNumber:

```
/sysContact

/management/alarmFilter

/nmsgroup/objectTable.exitPort.portNumber
```

Note that attributes located in the top object have to be preceded by a `"/"` character.

7.5.3 The value list

What is a value list?

Using the value list you actually define the values for the corresponding attribute or element as you want to see them.

Value list syntax

A value list has the following syntax:

`<value_list> = <string_value_list> | <integer_value_list> ";"`

with `<string_value_list> = { <string_value> "," } <string_value>`

where `<string_value> = ( IDENTIFIER | STRING )`

with `<integer_value_list> = { <integer_value> "," } <integer_value>`

where `<integer_value> = ( IDENTIFIER | STRING ) "(" NUM ")"`

Integer values

As mentioned before, you can only user define values for integer attribute values. A user defined integer value consists of two parts:

<table>
<tr><th>Part</th><th>Description</th></tr>
<tr><td>1</td><td>The first part is the user defined name. There are two ways to encode this name:
<table>
<tr><th>Encoding</th><th>Description</th></tr>
<tr><td>STRING</td><td>The value starts with a double quote (″) character. The end of the string is the next double quote (″) character. Within these two delimiters any character is allowed (even the <code>"#"</code> character, i.e. it will not be considered as comment).
The closing double quote should appear before the end of the line. Else the string is interpreted incorrectly.
<b>Example</b>
<code>/alarmLevel.Access =</code>
<code>"level - low" (0) , "level - high" (10) ;</code></td></tr>
<tr><td>IDENTIFIER</td><td>The value does not start with a double quote. The end of the string is the first space, tab or end of line character that is encountered.
Note that using this encoding, you can not use characters that conflict with the normal syntax (such as a comma, a space, an equal sign, etc.). Use the STRING encoding instead.
<b>Example</b>
<code>/alarmLevel.Access = low (0) , high (10) ;</code></td></tr>
</table></td></tr>
<tr><td>2</td><td>The second part is the actual value for which this name is defined. See the examples above.</td></tr>
</table>

7.6 An example of a *custom.txt* file

The following shows an example of a *custom.txt* file:

```
[common]

# These are some common definitions.

alarmLevel/Access = "Level = low" (0) , "Level = medium" (5) , "Level = high" (10) ;
management/alarmFilter = lowSenseFilter (0) , mediumSenseFilter (5) , highSenseFilter (10) ;


[crocusHDSLTT]

# The definition of the LinkDown alarm applies on both lines.
# The definition of the SeverelyErroredSecond alarm applies on line 1 only.

line/alarmLevel.LinkDown = low (0) , medium (5) , high (10) ;
/modem/line[1]/alarmLevel.SeverelyErroredSecond = low (0) , medium (5) , high (10) ;
```

Sections

The sections in this example are `common` and `crocusHDSLTT`. The `common` section applies to all devices. The section `crocusHDSLTT` applies to the Crocus HDSL F TT.

Remarks

As you can see in the example, some remarks are given. These remarks are preceded by a `"#"` character.

Definitions

Section	Description
`common`	Suppose that you want to give an understandable name to an otherwise very cryptic integer value such as the value of the attribute alarmFilter. Suppose you want to do this for all devices that have the attribute alarmFilter in their management object. In that case include the definition in the common section and format it as in the example.
`crocusHDSLTT`	Suppose that for the Crocus HDSL TTs you want give an understandable name to the values of the elements LinkDown and SeverelyErroredSecond of the alarmLevel attribute of the line objects. However, suppose you want the LinkDown definition to apply to all line objects, but the SeverelyErroredSecond definition to apply to line[1] only. In that case, format the definition as in the example.

Reference manual

8. The TMA CLI commands

This chapter gives a complete overview of all TMA CLI commands. For each command, it gives the exact syntax, a description and some examples. But first this chapter explains some keywords that are used to describe the syntax of the commands.

The following table gives an overview of this chapter:

8.1 TMA CLI command overview

The following table gives an overview of all available TMA CLI commands:

Use the command …	In order to …
`action`	trigger an action.
`cfgload`	download a binary configuration file (**.cms*) to the device.
`disconnect`	close the current TMA CLI session.
`exec`	execute a script file.
`get`	get the value of … • a simple attribute • a complex attribute • an element (i.e. a simple attribute within a complex attribute) • all the attributes in an object • all the attributes in the *“Edit Configuration”* group.
`logging on`	start logging TMA CLI input and output to a file.
`logging off`	stop logging.
`memload`	to download files to an IP device.
`select`	browse through the containment tree.
`selgrp`	to select another attribute group.
`set`	set the value of … • a simple attribute • a complex attribute
`setenv`	• list all environment variables • define an environment variable
`source`	execute a script file and export the environment variables after executing the script commands.

- TMA CLI commands are not case sensitive whereas their parameters are.
- Do not confuse CLI with TMA CLI! Whereas CLI is a management tool which runs on the Telindus access devices themselves, TMA CLI is a program which you have to install on a computer. TMA CLI is a more elaborate command line interface, with more features and commands then CLI. Whereas most of the TMA CLI commands are the same as the CLI commands, some commands that exist in TMA CLI do not exist in CLI. These commands are:
  `exit, logging, exec, cfgload, memload and source.`

Refer to the following section for an overview of the keywords that can be assigned to a command.

8.2 Keywords of the TMA CLI command syntax

TMA CLI commands have parameters that are composed as a set of keywords and values that are separated from each other by one or more white space characters. A white space character may be a space character or a tab character.

The following table gives an overview of all the keywords:

The keyword …	Specifies …
`<object_name>`	the name of an object. It is a combination of upper- and lower-case characters. If an object type requires an instance value, the ASCII representation of the instance value is given between square brackets, e.g. `line[2]`.
`<group_name>`	the name of the group to which the attribute or actions belongs. It is a combination of upper-, lower-case and white space characters.
`<attribute_name>`	the name of the attribute. It is a combination of upper- and lower-case characters.
`<element_name>`	the name of an element within a structured attribute value (also called complex attribute or complex attribute value). It is a combination of upper- and lower-case characters.
`<action_name>`	the name of the action. It is a combination of upper- and lower-case characters and embedded spaces.
`<file_name>`	the name of a file either to import or to export.
`<value>`	the representation of the value for the attribute or element. Depending on the type, the value has a different representation. Refer to the following section.

8.3 Possible values

The following table lists all possible attribute and element values:

The value …	Specifies …
`integer`	one or more decimal digits, no leading zeros, minus sign for negative values. e.g. 12, -12, not 012
`enumerated`	the textual representation of the enumerated value. If the textual representation contains characters that conflict with the syntax (such as a comma, a space, an equal sign, etc.), then the string has to be surrounded by a set of double quote characters: " ".
`printable string`	a string of printable characters. • If the printable string contains characters that conflict with the syntax (such as a comma, a space, an equal sign, etc.), then the string has to be surrounded by a set of double quote characters: " ". • A null string is represented as two consecutive double quotes: " ".
`octet string`	the hexadecimal representation of the value surrounded by: ' 'H. e.g. `'0355'H`
`bit string`	a set of bits where each bit has a name. The value of a bit string is shown in the same way as for an enumerated value.
`IP address`	a formatted type existing of four decimal values separated by a dot character.

8.4 The *action* command

Syntax

```
ACTION <action_name> [ <action_argument> = <action_value_specification> ]
```

Description

This command allows the user to start an action. Actions are related to an object in a selected group. In other words, the available actions depend on the current place in the containment tree.

For a list of all available actions, refer to the manual of the specific Telindus access device.

The parameter `<action_value_specification>` can represent the following:

<action_value_specification>	**Syntax**
simple action value specification	`<simple_action_value_specification> ::= <simple_value>`

complex action value specification	`<complex_action_value_specification> ::=`
	`“{“ { <element_name> = <value> } “}”`
	with `<value> = <simple_value> | <struct_value>`

Examples

An action without an argument:

```
/crocusSDSLTT:"Edit Configuration"
>action “Activate Configuration”
OK
/crocusSDSLTT:"Edit Configuration"
>
```

An action with a simple action value specification:

```
/crocusSDSLTT/modem:Performance
>action “Test Activation” testActivationType = “AL test”
OK
/crocusSDSLTT/modem:Performance
>
```

An action with a complex action value specification:

```
/o1003/router:Performance
>action startPing pingData = {ipAddress = 172.31.40.10  iterations = 30}
OK
/o1003/router:Performance
>
```

8.5 The *cfgload* command

Syntax

```
CFGLOAD <config_file_name>
```

Description

This command downloads a previously stored configuration from a file to an access device. The configuration file should have the binary CMS2 formatting. Such kind of file has the extension **.cms*. The file name may include a path name. White spaces within the file name are not allowed.

Use the application program TMA to export a configuration to a CMS2 formatted file. When exporting you can choose between two options:

- If the configuration file has been exported with the *Full configuration* selection, all the attributes of the device that loads the configuration file are changed.
- If the configuration file has been exported with the *Selected attributes* or *All attributes of …* selection, only the attributes that were saved will be changed. All other attributes remain unchanged.

See the TMA manual for in-depth information about exporting a configuration.

⚠ Before importing configuration files for devices which only support TMA when connected via a management concentrator (such as the Crocus HS), first perform the *“Load Default Configuration”* action.

ℹ The *cfgload* command can only be executed when the *“Edit Configuration”* group is selected.

Example

Importing the configuration file *config.cms* located on a floppy disk into a Crocus SDSL:

```
/crocusSDSLTT:"Edit Configuration"
>cfgload a:\config.cms
OK
/crocusSDSLTT:"Edit Configuration"
>
```

8.6 The *disconnect* command

Syntax

```
DISCONNECT
```

Description

This command closes the current TMA CLI session on a device.

8.7 The *exec* command

Syntax

```
EXEC <script_file_name>
```

Description

This command executes TMA CLI scripts. The file name may include a path name. White spaces within the file name are not allowed.

If you …

- have a script that, among other things, sets the environment variables,
- use the *exec* command to run the script,
- are in interactive TMA CLI mode,

… then the environment variables as set in the script are not retained when the script is finished.

Also refer to the complementary command *source* (Section 8.15 - The source command).

Example

Consider the script file *status.cli* containing the command to retrieve the line status from a device:

```
get modem/line:Status
```

From the shell prompt you can execute this script on a Crocus SDSL with IP address 172.31.5.200:

```
TmaCli 172.31.5.200 –exec status.cli
```

You can also execute the script in interactive mode:

```
/crocusSDSLTT:"Edit Configuration"
>exec status.cli
OK
/crocusSDSLTT:"Edit Configuration"
>get modem/line:Status
  GET
    {
    SELECT modem
      {
      SELECT line
        {
        LIST
          {
          timeSinceLastRetrain = "0d 0h 0m 0s"
          lineState = idle
          lineAttenuation = 62.0dB
          noiseMargin = -16.0dB
          ifSpeed = 0
          ifOperStatus = down
          }
        }
      }
    }
OK
>
```

8.8 The *get* command

Syntax

```
GET [-d] [-f] [-m] [-r] [-s] [-t] [-v] [-w] [ <specification> ]
```

Description

The *get* command retrieves the values of:

- a simple attribute
- a complex attribute
- a row of a table
- an element (within a complex attribute)
- all the attributes in an object
- all the attributes in the *"Edit Configuration"* group.

As stated in the syntax, the *get* command has two basic formats:

- The first format can be used to retrieve values of a device in such a way that the result can be used as a base for creating a *set* command.
- The second format retrieves the values from a table and will produce an output that can be used to be imported in a spreadsheet. This kind of output format can not be used as a base for creating a *set* command.

Depending on the kind of `<specification>` the *get* command gives a different output:

***Get* with …**	**Shows …**
no specification	the values of the attributes within the selected object and the selected group.
an object specification	the values of the attributes within the specified object and the selected or specified group.
an attribute specification	the value of the specified attribute.
an element specification	the value of the specified element within a complex attribute.
a row specification	the value of the specified row of the attribute (i.e. table).

Continued on next page

The get command (continued)

Options

Adding options to the *get* command allows you to format the output:

The option …	Results in showing ...
`-d`	only those values that are different from the default or read-only values.
`-f` ⚠	the full non-default configuration including the actions *“Load Default Configuration”* and *“Activate Configuration”* respectively at the beginning and the end of the list. The *get -f* command is only supported in the *“Edit Configuration”* group.
`-m`	as much simple attribute values as possible on one line while taking the environment variable `COLS` into account.
`-r` ⚠	attribute values of the current and underlying objects. The attributes on the same level are grouped by *LIST{ … }* and an underlying object is selected by the *SELECT* command. The *get -r* command is only supported in the *“Edit Configuration”* group.
`-s`	the filtered values of a table.
`-t`	the nested tables.
`-v`	the values of a table in rows and columns and separated by the value separator which is defined by the environment variable `VALUESEPARATOR`.
`-w`	the values of a complex attribute value on one line while taking the environment variable `COLS` into account.

Continued on next page

The get command (continued)

Specification

The parameter `<specification>` can represent the following:

```
<specification> ::= ( <object_specification> |
                      <attribute_specification> |
                      <row_specification> |
                      <element_specification> )
                      [ ":" <group_specification> ]
```

The following table gives a description of the different kinds of specifications:

<table>
<tr><th><b>&lt;specification&gt;</b></th><th>Description</th></tr>
<tr><td>object specification</td><td>There are two possible object specifications:
• An <i>absolute</i> object specification, i.e. starting from the top object.
e.g. /crocusSDSLTT/modem
• A <i>relative</i> object specification, i.e. starting from the current object.
e.g. modem/line</td></tr>
<tr><td>attribute specification</td><td>Specifies an attribute within an object and selected group. Possible notations are:
• the name of the attribute.
e.g. sysName
• an object specification followed by the name the attribute.
e.g. /crocusSDSLTT/sysName or modem/line/speed</td></tr>
<tr><td>row specification</td><td>This is an attribute specification followed by a set of square brackets that enclose an option. Depending on the command it is used with, an option can have different meanings:
<table>
<tr><th>Command</th><th>Option</th><th>Description</th></tr>
<tr><td>get</td><td>[$_x$]</td><td>Read row number x of a table.</td></tr>
<tr><td>set</td><td>[a]</td><td>Append a row at the end of the table.</td></tr>
<tr><td>set</td><td>[i$_x$]</td><td>Insert a row in a table before row number x.</td></tr>
<tr><td>set</td><td>[$_x$]</td><td>Change a value on row number x of a table.</td></tr>
<tr><td>set</td><td>[d$_x$]</td><td>Delete row number x from a table.</td></tr>
<tr><td>set</td><td>[d]</td><td>Delete all rows from a table.</td></tr>
</table></td></tr>
<tr><td>element specification</td><td>There are two possible element specifications:
• An attribute specification followed by a slash and an element name.
e.g. alarmMask/Boot, alarmLevel/Access, security/password
• A row specification followed by a slash and an element name.
e.g. security[1]/password, security[1]/accessRights/ReadAccess</td></tr>
</table>

Continued on next page

Specification (continued)

`<specification>`	**Description**
group specification ⚠	The group is specified at the end of the attribute / element / row specification separated by a double point ‘:’. This is only supported for the *get* command. If no group specification is present, the currently selected group is used.

Examples

Refer to Section 6.3 - Reading attribute values.

8.9 The *logging* command

Syntax

```
LOGGING ON <log_file_name>

LOGGING OFF

LOGGING
```

Description

The *logging* command controls the logging of TMA CLI input and output to a file.

<table>
<tr><th>Logging command</th><th>Description</th></tr>
<tr><td><code>LOGGING ON <log_file_name></code></td><td>Use this syntax to start logging TMA CLI input and output to a file. The file name may include a path name. White spaces within the file name are not allowed.</td></tr>
<tr><td><code>LOGGING OFF</code></td><td>Use this syntax to stop the file logging.</td></tr>
<tr><td><code>LOGGING</code></td><td>Use this syntax to check the status of the <i>logging</i> command:
<table>
<tr><th>If logging is …</th><th>and you type <code>logging</code>, then the following is displayed:</th></tr>
<tr><td>on,</td><td><code>LOGGING is on (log_file_name)</code></td></tr>
<tr><td>off,</td><td><code>LOGGING is off</code></td></tr>
</table>
</td></tr>
</table>

Example

Start logging I/O to the file *C:\data\cli\logfile1.txt*:

```
/crocusSDSLTT:"Edit Configuration"
>logging on C:\data\cli\logfile1.txt
OK
/crocusSDSLTT:"Edit Configuration"
>
```

You can always ask the status of the logging command. For the example above the result would be:

```
/crocusSDSLTT:"Edit Configuration"
>logging
LOGGING is on (logfile1.txt)
OK
/crocusSDSLTT:"Edit Configuration"
>
```

8.10 The *memload* command

Syntax

```
MEMLOAD <local_file_name>@<remote_file_name>
```

Description

This command download files to an IP device. Using this command you can download:

- new firmware to the flash banks of an IP device,
- various files to the file system of an IP device.

For more information on this matter, refer to Section 6.10 - Downloading files.

Example

Download firmware file T1042017.00 located on a floppy disk to flash bank 1 of the Orchid 1003 LAN:

```
/o1003:"Edit Configuration"
>memload a:\T1042017.00@CONTROL1
OK
/o1003:"Edit Configuration"
>
```

8.11 The *select* command

Syntax

```
SELECT <object_specification>

SELECT ..

SELECT /
```

Description

This command allows you to browse through the containment tree of a Telindus device:

Use the syntax …	In order to …
`SELECT <object_specification>`	go to the specified object. There are two possible object specifications: • An *absolute* object specification, i.e. starting from the top object. e.g. `/crocusSDSLTT/modem` • A *relative* object specification, i.e. starting from the current object. e.g. `modem/line`
`SELECT ..`	go up one level in the containment tree.
`SELECT /`	go to the top object in the containment tree.

The *select* command may be used within a structured value assignment using the *set* command. However, in this case, only a relative object specification is allowed. For more information on this subject, refer to Section 6.5.8 - Setting several attribute values at once and Section 6.5.9 - Setting values obtained with the get command.

Example

From the top object to an underlying object:

```
/crocusSDSLTT:"Edit Configuration"
>select modem/line
OK
/crocusSDSLTT/modem/line:"Edit Configuration"
>
```

8.12 The *selgrp* command

Syntax

```
SELGRP

SELGRP <group_name>
```

Description

This command selects the specified group of attributes. The different groups are:

The group …	Contains …
“Edit configuration”	configuration attributes, i.e. the device settings. Provided you have write and security access, all configuration attributes can be changed.
Status	status attributes. These give you information on the current operational state of the device.
Performance	performance attributes. These give you statistical information on the performance and efficiency of the device.
Alarms	a list of alarms and an indication of which alarm(s) is (are) currently active.

When you use the *selgrp* command without argument, the command returns the currently selected group.

⚠ If the group name contains embedded spaces, quoting is required. e.g. *“Edit Configuration”*

Example

Select the Status group:

```
/crocusSDSLTT:"Edit Configuration"
>selgrp Status
OK
/crocusSDSLTT:Status
>
```

8.13 The *set* command

Syntax

```
SET <value_specification>

SET {SELECT <object_specification>[<object_instance_value>]{}}

SET {DELOBJ <object_specification>[<object_instance_value>]}
```

Description

This command allows you to …

- change a value of a configuration attribute.
- add a user instantiatable object to the containment tree.
- delete a user instantiatable object from the containment tree.

Value specification

The parameter `<value_specification>` can represent the following:

- simple value specification
- complex value specification 1
- complex value specification 2.

These different specifications are explained in the following three paragraphs.

Definition of a simple value specification

Syntax

```
<simple_value_specification> ::= <specification> = <simple_value>
```

Description

Refer to Section 8.8 - The get command for a definition of `<specification>`.

- The `<specification>` can not be an `<object_specification>`.
- The `<specification>` may contain an absolute or a relative object specification.
- The type of the specified attribute, element or row should be a simple type.

Definition of a complex value specification 1

Syntax

```
<complex_value_specification> ::=
                   <attribute_specification> = "{" { <element_name> = <value> } "}"
```

with `<value> = <simple_value> | <struct_value>`

Description

This type of specification can be used to set the value of a complex attribute (one or more elements) within the selected object. It may start with an absolute or relative object specification.

Continued on next page

The set command (continued)

Definition of a complex value specification 2

Syntax

A shortened syntax of this type of specification can be described as:

```
SET
{
  LIST
  {
    <simple_attribute_name> = <simple_value>
    <complex_attribute_name> =
    {
      <simple_element_name> = <simple_value>
      <complex_element_name> =
      {
        { <element_name> = <value> }
      }
    }
    <table_attribute_name> =
    {
      <row_specification> =
      {
        <simple_element_specification> = <simple_value>
        <complex_element_name> = { <element_name> = <value> }
      }
    }
  }
  SELECT <relative_object_specification>
  {
    LIST
    {
      <value_specifications>
    }
  }
  SELECT <relative_object_specification>
  {
    LIST
    {
      <value_specifications>
    }
  }
}
```

Description

This type of specification can be used to set the values of a number of attributes of an object.

Continued on next page

The set command (continued)

Object instance value

If the user instantiatable object you want to add / remove has …

- no instance value (also called index), then you do not have to specify an instance value:
  - `SET {SELECT <object_specification>{}}`
  - `SET {DELOBJ <object_specification>}`

  User instantiatable objects that have no instance value are objects of which only one can be present in the containment tree.

- an instance value (also called index), then you have to specify an instance value:
  - `SET {SELECT <object_specification>[<object_instance_value>]{}}`
  - `SET {DELOBJ <object_specification>[<object_instance_value>]}`

  User instantiatable objects that have an instance value are objects of which several can be present in the containment tree. The instance value distinguishes them from one another. The parameter `<object_instance_value>` is the instance name you want to assign to the user instantiatable object. It is a character string of maximum 24 characters.

Important remark

- When adding a user instantiatable object (i.e. using the `set / select` combination), the curled brackets {} behind the user instantiatable object name have to be present! This because TMA CLI expects curled brackets after a `select` command within a `set` command. If you want, you can insert more select commands or a list of attributes between these curled brackets.
- When removing a user instantiatable object (i.e. using the `set / delobj` combination), the curled brackets {} behind the user instantiatable object may not be present!

Examples

Refer to Section 6.5 - Setting attribute values and Section 6.6.2 - How to add a user instantiatable object?.

8.14 The *setenv* command

Syntax

```
SETENV
```

```
SETENV <environment_variable> = <integer_value>
```

Description

This command sets the environment variables. These variables control the output behaviour of TMA CLI. For more information on this subject, refer to Section 4.9 - The environment variables.

Depending on the syntax, the command reacts differently:

Use the syntax …	In order to …
`SETENV`	display the environment variables.
`SETENV <environment_variable> = <integer_value>`	define an environment variable.

Environment variables are case-sensitive. They have to be typed in capital letters.

Examples

Set the environment variable COLS to 60:

```
/crocusSDSLTT:"Edit Configuration"
>setenv COLS = 60
OK
/crocusSDSLTT:"Edit Configuration"
>
```

Set the environment variable VALUESEPARATOR to /:

```
/crocusSDSLTT:"Edit Configuration"
>setenv VALUESEPARATOR = /
OK
/crocusSDSLTT:"Edit Configuration"
>
```

In order to have tabs as value separator, set the environment variable VALUESEPARATOR to ^I:

```
/crocusSDSLTT:"Edit Configuration"
>setenv VALUESEPARATOR = ^I
OK
/crocusSDSLTT:"Edit Configuration"
>
```

8.15 The *source* command

Syntax

```
SOURCE <script_file_name>
```

Description

This command executes TMA CLI scripts. The file name may include a path name. White spaces within the file name are not allowed. After execution of the script, the environment variables as set in the script are retained.

If you …

- have a script that, among other things, sets the environment variables,
- use the *source* command to run the script,
- are in interactive TMA CLI mode,

… then the environment variables as set in the script are retained when the script is finished.

Also refer to the complementary command *exec* (Section 8.7 - The exec command).

Example

Consider the script file *perform.cli* containing the command to retrieve the h2Modem performance attribute values from a device. What is more, the values have to be displayed in a table (*get –v*) and with a "|" as value separator.

```
setenv VALUESEPARATOR = |
get -v modem/h2Modem:Performance
```

Execute this script:

```
/crocusSDSLTT:"Edit Configuration"
>exec perform.cli
OK
/crocusSDSLTT:"Edit Configuration"
>setenv VALUESEPARATOR = |
OK
/crocusSDSLTT:"Edit Configuration"
>get -v modem/h2Modem:Performance
  GET
    {
    SELECT modem
      {validity|period|noSyncTime
      LIST
        {valid|"-120min -> -105min"|"0d 0h 15m 0s"
valid|"-105min -> -90min"|"0d 0h 15m 0s"
valid|"-90min -> -75min"|"0d 0h 15m 0s"
valid|"-75min -> -60min"|"0d 0h 15m 0s"
valid|"-60min -> -45min"|"0d 0h 15m 0s"
valid|"-45min -> -30min"|"0d 0h 15m 0s"
valid|"-30min -> -15min"|"0d 0h 15m 0s"
valid|"-15min -> 0min"|"0d 0h 15m 0s"
        }
      }
    }
OK
/crocusSDSLTT:"Edit Configuration"
>
```

After the script is finished, the environment variables as set in the script are retained.

9. The *Cms2Serv.ini* file

The *Cms2Serv.ini* file contains the communication parameters of TMA CLI. In some cases it may be necessary to change some of these parameters. Therefore, this section lists and explains the different *Cms2Serv.ini* file parameters. It also explains how to add parameters to the *Cms2Serv.ini* file.

The following table gives an overview of this chapter.

9.1 Parts of the *Cms2Serv.ini* file

The *Cms2Serv.ini* file is divided into several parts. Each part and its parameters is explained below. The values behind the parameters are the default values.

The [Cms2] part

The [Cms2] part contains the following parameters:

Parameter	Description
WindowSize=4	TMA CLI sends a number of frames to the device. Then TMA CLI has to receive an acknowledgement from the device before it sends the following frames. WindowSize is the number of frames which TMA CLI may send before it has to receive an acknowledgement from the device.
BufferSize=250	This is the number of bytes in one frame.
Retries=1	When there is no response from the device, TMA CLI will retransmit the current frame a number of times before ending the session. Retries sets the number of retransmissions.
LocalPollDelay=2	The local polling mechanism on itself has nothing to do with TMA CLI. Fact is that TMA CLI is not allowed to send anything during a local polling session. That is way the LocalPollDelay is used. It gives an extra delay, in seconds, on the reply of a device.
ExtTimeOut=10	This is the timeout, in seconds, for opening and closing a session. The timeout is used during *Connect to network* and *Select device* actions. When there is no response from the device after this timeout, the *open* and *close* frames are retransmitted the number of times which is specified in the Retries parameters.
PollPeriod=30	Once you are connected to a device, it is regularly polled to see whether it is still there. PollPeriod sets the poll interval in seconds.
ServerPort=31416	This is the TCP port on which the TMA Comms Handler “listens”.
WaitClose=300	The TMA Comms Handler normally closes when all TMA CLI sessions are closed. However, with WaitClose you can set a timeout (in seconds) for which the TMA Comms Handler will wait before it really closes down. If you restart a new TMA CLI session within this timeout period, the rebooting of the TMA Comms Handler will be accelerated. This because the TMA Comms Handler was not yet closed down completely.

The [Tftp] part

The [Tftp] part contains the following parameters:

Parameter	Description
Rexmt=30	Retransmission timeout, in seconds, for a TFTP connection.
Timeout=60	Total timeout, in seconds, for a TFTP connection. I.e. retransmission continues until Timeout has expired.

Continued on next page

Parts of the Cms2Serv.ini file (continued)

The [Comm] part

The [Comm] part contains the following parameters:

Parameter	Description
Speed=9600	This is the COM port speed in bits per second.
NormTimeOut=8	Normal CMS2 retransmission and reply timeout, in seconds, for a serial connection.

The [LAN] part

The [LAN] part contains the following parameters:

Parameter	Description
NormTimeOut=10	Normal CMS2 retransmission and reply timeout, in seconds, for an IP connection.

9.2 Adding parts to the *Cms2Serv.ini* file

You can set specific communication parameters for each IP address or COM port. The communication parameters in this specific part overrule those in the general part of the *Cms2Serv.ini* file.

Example

Suppose you want to change the following parameters:

- NormTimeOut for IP address 194.7.26.4
- Speed for COM port 2.

Proceed as follows:

For the part …	Proceed as follows …	
[LAN]	**Step**	**Action**
	1	Copy the [LAN] part, and paste it at the end of the *Cms2Serv.ini* file.
	2	Rename [LAN] to [LAN_194.7.26.4].
	3	Delete all the parameters except NormTimeOut.
	4	Change the NormTimeOut value from 10 to, for instance, 30.
[Comm]	**Step**	**Action**
	1	Copy the [Comm] part, and paste it at the end of the *Cms2Serv.ini* file.
	2	Rename [Comm] to [Comm_2].
	3	Delete all the parameters except Speed.
	4	Change the Speed value from 9600 to, for instance, 57600.

The added parts are then as follows:

```
[LAN_194.7.26.4]
NormTimeOut=30

[Comm_2]
Speed=57600
```

10. Troubleshooting

This section explains the error message syntax. The following table gives an overview of this chapter:

10.1 Error messages

When a TMA CLI executes a command successfully, you will see “OK” in the output. However, when the command execution failed, the reply has the following format:

If the source of the error is …	Then the format of the error message is …
a syntax error	`NOK, <line_number>, <error_code>, <error_condition>`
a failing answer from a device	`NOK, <error_code>, <error_condition>`

Refer to Annex A: error codes for a complete list of all the possible error codes.

10.2 Syntax errors

In case of a syntax error, the TMA CLI error message consists of four fields separated with a comma:

Field	Contents	Description
1	`NOK`	String meaning “not OK”.
2	`line_number`	Number of the line in which the error occurred.
3	`error_code`	Numeric value corresponding to the error condition.
4	`error_condition`	Textual description of the error condition.

10.3 Failing answer from a device

The two main causes of a failing answer from a device are:

- There is no network connection towards the device. Carefully check the connections between your computer running TMA CLI and the device you are trying to reach.
- There are no or outdated model files of the device present on your computer. Reinstall the latest model files from the Telindus web site. Refer to Section 2.6 - How to upgrade the model files.

If no answer is received from a device, the following error message appears:

```
NOK, 15, NACK received from device errorcode: <communication_error>
```

where the `<communication_error>` is one of the errors as listed in Annex A: error codes, Communication error codes.

Annexes

Annex A: error codes

This annex list all the error codes. They are divided into four groups:

- the CLI error codes
- the TMA CLI error codes
- the TFTP error codes
- the communication error codes.

CLI error codes

Error code	Error description
1	Invalid command
2	Invalid argument
3	Invalid object name
4	Invalid group name
5	Invalid attribute name
6	Invalid element name
7	Invalid row number
8	Invalid key value
9	Invalid value
10	No closing quote
11	No equal sign
12	No access rights
13	No model
14	Not connected
15	NACK received from device error code : <List of communication error codes>
16	Already connected
17	Already in top object
18	Invalid row operation
19	Invalid instance value
20	No {
21	No }
22	Invalid action name
23	Invalid object type
24	No value
25	Mismatch between { }
26	LIST not allowed
27	SELECT not allowed
28	Device not known
29	Device supports no filter
30	Row-specific get not allowed
31	Option not valid in this combination

TMA CLI error codes

Error code	Error description
101	Communication error
102	Could not execute command
103	Could not open file
104	Could not open file for writing
105	Internal error
106	Invalid option
107	Missing argument
108	Missing filename
109	Missing local filename
110	Missing remote filename
111	No response
112	Not a valid cms file
113	Rejected by device
114	Syntax error (missing @)
115	Unable to send
116	Unexpected token
117	Unknown error

TFTP error codes

Error code	Error description
200	Tftp download successful
201	Not enough arguments
202	Download cancelled
203	Send or receive timed out
204	Could not open local file
205	Could not create socket
206	Could not initiate tftp WRQ
207	Receive error
208	Local file read error
209	Send error
210	Retransmission error
211	Invalid tftp request
212	Invalid tftp packet
213	Invalid address type
214	Unable to obtain download status
215	Could not start TmaTftp.exe (Windows only) Could not start tftp (Unix only)
216	<error message from tftp server > (Unix only)

Communication error codes

Error code	Error description
0x00	Session not open
0x13	Session open by someone else
0x14	Storage limitation reached
0x15	Incomplete information
0x16	Data is inconsistent with previous information
0x17	Unknown command
0x18	Unknown object
0x19	Not answer object
0x1A	Rejected data
0x30	Loss of communication
0x31	Requested port not available
0x32	No device connected (CTS-timeout)
0x33	Port open failed
0x34	Link set-up timeout
0x35	Transmitter error
0x36	Receiver error
0x37	Command busy
0x38	Retransmission error
0x50	Unknown error

Annex B: abbreviations

The following table gives a list of abbreviations and their description.

Abbreviation	Description
CLI	Command Line Interface
GUI	Graphical User Interface
HPOV	HP OpenView
IP	Internet Protocol
LAN	Local Area Network
MIB	Management Information Base
SNMP	Simple Network Management Protocol
TCP/IP	Transport Control Protocol / Internet Protocol
TMA	Telindus Maintenance Application

Annex C: product information

The following table displays the product information of TMA CLI.

Sales code	Product name	Description
156688	TMA_CLI WINDOWS NT	Command Line Interface application for WINDOWS NT. 3-years Maintenance contract for TMA_HP/OV mandatory.
156687	TMA_CLI SUN SOLARIS	Command Line Interface application for SUN SOLARIS. 3-years Maintenance contract for TMA_HP/OV mandatory.
169442	MANUAL TMA-CLI (E)	Manuals are delivered with the product in electronic format (CD-ROM) for environmental reasons. If however a hardcopy (print-out) of the manual is required, this sales item can be used. Between brackets an indication of the language.

The following table displays the product information on the maintenance contracts for TMA CLI and related products.

Sales code	Product name	Description
163135	TMA_CLI MAINT. CONTRACT 3Y	Yearly maintenance contract for TMA_CLI with a minimum duration of 3 years.
163131	TMA_HP/OV ENTRY LEVEL + TMA_CLI MAINT. CONTRACT 3Y	Yearly maintenance contract for TMA_HP/OV Entry Level Versions combined with TMA_CLI with a minimum duration of 3 years.
163133	TMA_HP/OV UNLIMITED + TMA_CLI MAINT. CONTRACT 3Y	Yearly maintenance contract for TMA_HP/OV Unlimited Versions combined with TMA_CLI with a minimum duration of 3 years.

Annex D: licence key request

In order to obtain the required licence key, do one of the following:

- send a fax to +32 16 382515
- send an email to productinfo@telindus.be

Which information has to be supplied?

The fax or the email should contain the following information:

- your company name and location
- the serial number which you can find on the CD-ROM label, e.g. *S.N.:9943964*
- the CD-ROM code which you can find on the CD-ROM label, e.g. *V.:S0007/00900*

Licence key request fax

The following page displays a prepared licence key request fax. You can use this fax to obtain a licence key. Proceed as follows:

Step	Action
1	Make a printout of the fax document displayed on the following page.
2	Fill in the required information on the dotted lines.
3	Send the fax to +32 16 382515.

Fax

To:	TELiNDUS	**From:**	……………………………………………
Fax:	+32 16 382515	**Pages:**	1
Subject:	TMA CLI – licence key request	**Date:**	……………………………………………

Please send me a licence key for TMA CLI.

Company information

Name: ………………………………………………………………………………………

Address: ……………………………………………………………………………………

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

Telephone number: ………………………………………………………………………...

Fax number: ……………………………………………………………………………...

Email address: …………………………………………………………………………...

Other information

The serial number on the CD-ROM label is ……………..…………..…………………………..

The CD ROM code on the CD ROM label is ……………………………………………….