

---

# **TANDBERG MXP for Cisco CallManager API User Manual**

TANDBERG  
D13784 Rev 01

---

## Table Of Contents

TANDBERG MXP for Cisco CallManager SCCP API .....	1
User Manual .....	1
1 The TANDBERG API.....	3
1.1 Introduction to XML.....	4
1.2 Introduction to XML Path Language (XPath) .....	6
1.3 The TANDBERG XML Engine .....	7
1.4 The XML Documents .....	8
2 The XML-based Advanced Command Line Interface .....	12
2.1 XACLI.....	13
2.2 The Status-type root command – xstatus .....	15
2.3 The Configuration-type root command - xconfiguration.....	17
2.4 The Command-type root commands - xcommand .....	19
2.5 XML Output - xgetxml .....	21
3 API - Configurations.....	22
3.1 configuration.xml – xconfiguration .....	23
4 API - Commands .....	26
4.1 commands.xml - xcommands .....	27
5 API - Status.....	30
5.1 status.xml – xstatus .....	31

# 1 The TANDBERG API

---

This document is a guide to the API interface of the TANDBERG MXP products. All rights reserved. This document contains information that is proprietary to TANDBERG. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronically, mechanically, by photocopying, or otherwise, without the prior written permission of TANDBERG. Nationally and internationally recognized trademarks and tradenames are the property of their respective holders and are hereby acknowledged.

## **Disclaimer**

The information in this document is furnished for informational purposes only, is subject to change without prior notice, and should not be construed as a commitment by TANDBERG. The information in this document is believed to be accurate and reliable; however TANDBERG assumes no responsibility or liability for any errors or inaccuracies that may appear in this document, nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of TANDBERG.

This document was written by the Research and Development Department of TANDBERG, Norway. We are committed to maintaining a high level of quality in all our documentation. Towards this effort, we welcome your comments and suggestions regarding the content and structure of this document. Please fax or mail your comments and suggestions to the attention of:

Research and Development Department  
TANDBERG  
P.O. Box 92  
1325 Lysaker  
Norway  
Tel: +47 67 125 125  
Fax: +47 67 125 234

[www.tandberg.net](http://www.tandberg.net)

# 1.1 Introduction to XML

XML is a markup language for documents containing structured information.

All information elements in an XML document are marked by a tag and a corresponding end-tag. The end-tag has the same name as the tag, but is prefixed with a slash, "/". All tags are put within angular brackets ("< >").

## Example 1.1

Below is an example of how configurations of a Serial Port could be represented using XML.

```
<Configuration>
  <SerialPort item="1">
    <BaudRate item="1">9600</BaudRate>
    <Parity item="1">None</Parity>
    <DataBits item="1">8</DataBits>
    <StopBits item="1">1</StopBits>
    <Mode item="1">Control</Mode>
  </SerialPort>
</Configuration>
```

From the tree structure of this example we can see that `BaudRate`, `Parity`, `DataBits`, `StopBits` and `Mode` are properties of the `SerialPort`. We can distinguish between *container-elements* and *value-elements*. Container-elements contain one or more sub-elements, while value-elements contain a value. This is analogous to files and folders on a computer. Container-elements are folders that can contain sub-folders and files, while value-elements are files containing data.

In the XML structure for the Serial Port we see that the container-element `SerialPort` contains five sub-elements. All these sub-elements are value-elements, each holding values for the properties: `BaudRate`, `Parity`, `DataBits`, `StopBits` and `Mode`.

## Example 1.2

In this example we will look at element attributes. Attributes are used to add meta information to an element. Attributes are placed within the start tag of an element and different attributes are separated by space.

An XML structure representing the status of a call in a videoconferencing system is shown below:

```
<Status>
  <Call item="1">
    <State item="1">OnHook</State>
  </Call>
</Status>
```

We can see from the `status` attribute of the `Call` element that the call is disconnected. The only relevant information regarding this call is the disconnect cause value. Therefore the sub-structure of the call element contains only one value-element.

## Example 1.3

If we now look at the call element for an active call we see that `call` element contains a large sub-structure:

```
<Status>
  <Call item="1" direction="Outgoing">
    <State item="1">Connected</State>
    <RemoteNumber item="1">7150001</RemoteNumber>
    <Duration item="1">116</Duration>
    <Channel item="1" type="Outgoing">
      <Rate item="1">768</Rate>
      <Audio item="1">
        <Protocol item="1">G722</Protocol>
        <Rate item="1">64</Rate>
      </Audio>
      <Video item="1">
        <Protocol item="1">H264</Protocol>
        <Resolution item="1">CIF</Resolution>
        <Rate item="1">295</Rate>
      </Video>
    </Channel>
    <Channel item="2" type="Incoming">
      <Rate item="1">768</Rate>
      <Audio item="1">
        <Protocol item="1">G722</Protocol>
        <Rate item="1">64</Rate>
      </Audio>
      <Video item="1">
        <Protocol item="1">H264</Protocol>
        <Resolution item="1">SIF</Resolution>
        <Rate item="1">224</Rate>
      </Video>
    </Channel>
  </Call>
</Status>
```

In this example, the attributes are used to provide valuable information in addition to establishing a dependency to the underlying sub-structure of the element.

#### Example 1.4

In the above examples, all elements are having an attribute named *item*. This attribute specifies the instance number of the element. If we expand [Example 1.1](#) to a system having two serial ports, the XML structure could look like this:

```
<Configuration>
  <SerialPort item="1">
    <BaudRate item="1">9600</BaudRate>
    <Parity item="1">None</Parity>
    <DataBits item="1">8</DataBits>
    <StopBits item="1">1</StopBits>
    <Mode item="1">Control</Mode>
  </SerialPort>
  <SerialPort item="2">
    <BaudRate item="1">19200</BaudRate>
    <Parity item="1">None</Parity>
    <DataBits item="1">8</DataBits>
    <StopBits item="1">1</StopBits>
    <Mode item="1">Auto</Mode>
  </SerialPort>
</Configuration>
```

## 1.2 Introduction to XML Path Language (XPath)

XPath is a comprehensive language to address data in XML documents. It is though very simple to understand the basics. If you are able to specify the path to a file on your computer, you are able to specify the path to an element in a XML structure.

### Example 1.5

Let us go back to the serial port configurations of [Example 1.1](#).

```
<Configuration>
  <SerialPort item="1">
    <BaudRate item="1">9600</BaudRate>
    <Parity item="1">None</Parity>
    <DataBits item="1">8</DataBits>
    <StopBits item="1">1</StopBits>
    <Mode item="1">Control</Mode>
  </SerialPort>
</Configuration>
```

To specify the path to the `SerialPort` element we simply start at the root level and separate the levels in the tree structure by a slash (“/”):  
`Configuration/SerialPort`

The path to the `BaudRate` element is:  
`Configuration/SerialPort/BaudRate`

### Example 1.6

To address a specific item of an element, the item number is added within brackets (“[]”) after the element name.

The path to the `BaudRate` element of `SerialPort` item 2 in [Example 1.4](#) is:

```
Configuration/SerialPort[2]/BaudRate
```

If the item number is omitted for an element, all items of this element will be addressed. The following expression addresses the `BaudRate` element of both serial ports:

```
Configuration/SerialPort/BaudRate
```

### Example 1.7

When using XPath it is possible to omit specifying intermediate levels in the address expression. By using the powerful “double slash” you can address elements without having to specify the complete path.

The expression below addresses the `BaudRate` element of both serial ports of [Example 1.4](#):

```
Configuration//BaudRate
```

## 1.3 The TANDBERG XML Engine

---

The TANDBERG XML engine is optimized for advanced machine-machine interaction between a TANDBERG system and an external control application. The main features can be summarized to:

- Structuring of information
- Addressing using XPath
- Feedback

### 1.3.1 Structuring of Information

An application programming interface can be seen as a gate where information is exchanged between two systems - a control application and a target system. The control application transmits instructions to the target system, while the target system supplies information about how these instructions are executed, in addition to other system related information.

Thus, the exchange of information can be divided into:

1. information flowing from target, hereby called *read information (r)*
2. information flowing to target, hereby called *write information (w)*

If we now look at the TANDBERG systems we can identify three main types of information, either being *read information (r)*, *write information (w)* or *read-write information (rw)*:

1. *(r) Read information – **Status Information.***  
Information about the system and system processes, i.e. information generated by the system.  
F.ex. status about ongoing calls, network status, conference status etc.  
All status information is structured in a hierarchy, making up a database constantly being updated by the system to reflect process changes.
2. *(w) Write information – **Command Information.***  
Information supplied by the user to initiate an action.  
F.ex. instructing the system to place a call, assigning floor to a specific site, disconnecting a site etc.  
A command is usually followed by a set of parameters to specify how the given action is to be executed.
3. *(rw) Read-Write information – **Configuration Information.*** Information defining system settings. This information can both be supplied and read by the user. F.ex. default callrate, baudrate of a serial port, enabling/disabling of various features etc.  
All configuration information is structured in a hierarchy making up a database of system settings. But for the Configuration information, the data in the database can only be updated by the user/control application.

### 1.3.2 Addressing using XPath

To address information in the hierarchic structure of Status and Configuration information the TANDBERG systems support abbreviated XML Path Language (XPath). This allows the user/control application to address everything from a single element of data, f.ex. the callrate of a specific call, to larger parts of the hierarchy, f.ex. all information available for a given call.

The structuring of information together with XPath for addressing makes up powerful features like searchability and setting of multiple instances of a configuration.

## 1.4 The XML Documents

### 1.4.1 Documents

The XML Data in the TANDBERG systems are divided into three main types of documents. The division is based on whether the information is *Read Information*, *Write Information* or *Read-Write* information:

1. **Status documents (r)**: Documents holding all available Status Information in the system.  
Supported documents:
  - a. status.xml
2. **Configuration documents (rw)**: Documents holding all system configurations.  
Supported documents:
  - a. configuration.xml
3. **Command documents (w)**: Documents defining the supported system commands used to initiate system processes. This is *write* data, i.e. the parameter values for a given command are defined by the user and posted to the system. The posted values will not be returned when reading the document from the system. Reading a command document from the system returns descriptions of the supported commands with empty parameter values.  
Supported documents:
  - a. command.xml
4. **Meta Documents**: Meta documents contain information that can be referenced by other documents, e.g. value domains of configurations or command parameters.  
Supported Meta Documents:
  - a. valuespace.xml

### 1.4.2 Status Documents (r)

The Status Documents are characterised by an extensive use of XML attributes. In addition to holding information, the attributes are used to reflect the structure of the sub-elements, which are dependent on the state of the system.

#### Example 9

The element `Call` will contain different sub elements depending on the call state, call type or direction:

```
<Call item="1" direction="Incoming">
  <State item="1">Connected</State>
  <RemoteNumber item="1">9798906</RemoteNumber>
  <Duration item="1">52</Duration>
  <Channel item="1" type="Outgoing">
    <Rate item="1">768</Rate>
    <Audio item="1">
      <Protocol item="1">G722</Protocol>
      <Rate item="1">64</Rate>
    </Audio>
    <Video item="1">
      <Protocol item="1">H264</Protocol>
      <Resolution item="1">CIF</Resolution>
      <Rate item="1">243</Rate>
    </Video>
  </Channel>
  <Channel item="2" type="Incoming">
```

```

<Rate item="1">768</Rate>
<Audio item="1">
  <Protocol item="1">G722</Protocol>
  <Rate item="1">64</Rate>
</Audio>
<Video item="1">
  <Protocol item="1">H264</Protocol>
  <Resolution item="1">SIF</Resolution>
  <Rate item="1">273</Rate>
</Video>
</Channel>
</Call>
---
<Call item="2" direction="Outgoing">
  <State item="1">Hold</State>
  <RemoteNumber item="1">7150001</RemoteNumber>
  <Duration item="1">34</Duration>
  <Channel item="1" type="Outgoing">
    <Rate item="1">0</Rate>
    <Audio item="1">
      <Protocol item="1">Off</Protocol>
      <Rate item="1">0</Rate>
    </Audio>
    <Video item="1">
      <Protocol item="1">Off</Protocol>
      <Resolution item="1">Off</Resolution>
      <Rate item="1">0</Rate>
    </Video>
  </Channel>
  <Channel item="2" type="Incoming">
    <Rate item="1">0</Rate>
    <Audio item="1">
      <Protocol item="1">Off</Protocol>
      <Rate item="1">0</Rate>
    </Audio>
    <Video item="1">
      <Protocol item="1">Off</Protocol>
      <Resolution item="1">Off</Resolution>
      <Rate item="1">0</Rate>
    </Video>
  </Channel>
</Call>
---
<Call item="3">
  <State item="1">OnHook</State>
</Call>

```

### 1.4.3 Configuration documents (rw)

The structure of the Configuration documents is independent of system state, i.e. the structure will be constant in time. In addition to holding the values for the various configurations, each configuration value-element includes an attribute, `valueSpaceRef`, referencing the value domain for the configuration.

#### Example 10

From the XML structure below we see that the `BaudRate` element of `SerialPort[1]` is configured to `9600`. The `BaudRate` element references the `SerialPortBaudrate` element in the `ValueSpace` document, showing the value domain for this configuration.

```
<Configuration>>
  <SerialPort item="1">
    <BaudRate item="1"
valueSpaceRef="/ValueSpace/SerialPortBaudrate[@item='1']">9600</BaudR
ate>
    .
    .
  </SerialPort>
  .
  .
</Configuration>

---

<ValueSpace>
  <SerialPortBaudrate item="1" type="Literal">
    <Value>1200</Value>
    <Value>2400</Value>
    <Value>4800</Value>
    <Value>9600</Value>
    <Value>19200</Value>
    <Value>38400</Value>
    <Value>57600</Value>
    <Value>115200</Value>
  </SerialPortBaudrate>
</ValueSpace>
```

To change configurations, the part(s) of the document containing the configurations to be updated should be posted back to the system with the new values. This will be described thoroughly in a later sections.

#### 1.4.4 Command documents (w)

Command documents contain descriptions of the supported commands for the system. A Command consist of a Command name and a set of Command parameters. The parameter elements have attributes to denote whether the parameter is optional or required, in a addition to referencing the value domain for the given parameter.

Command parameters do not contain any values when read from the system.

##### Example 11

The command `KeyPress` is defined to take one parameter. The value domain for the parameters is referenced by the attribute `valueSpaceRef`.

```
<Command>
  <KeyPress item="1">
    <Key item="1" valueSpaceRef="/ValueSpace/KeyPad[@item='1']"
required="" />
  </KeyPress>
</Command>
```

To issue a command, the command structure is posted back to the system together with values for the various parameters. Optional parameters can be omitted when posting the structure back to the system.

##### Example 12

To emulate pressing the microphone mute button the user can post the following XML structure to the system:

```
<Command>  
  <KeyPress>  
    <Key>MicrophoneOff</Key>  
  </KeyPress>  
</Command>
```

This command must be followed by a key release command.

```
<Command>  
  <KeyRelease/>  
</Command>
```

## 2 The XML-based Advanced Command Line Interface

---

XACLI is a very flexible interface both optimized for machine-machine interaction and man-machine interaction. It is based on the powerful TANDBERG XML engine and offers many of the same features as the TANDBERG XML interface.

The main distinction between XACLI and the TANDBERG XML interface is the input format. As XACLI is a command line interface all inputs from the user/control application have to be put on one line. This differs from the XML interface where a complete XML document can be posted to the system in one operation.

A basic understanding of the information structuring in the TANDBERG XML engine is important in order to get the most out of the XACLI interface. It is therefore recommended to read the documentation of the TANDBERG XML API prior to reading this section.

## 2.1 XACLI

### 2.1.1 Accessing XACLI

XACLI can be accessed through Telnet via the LAN interface or through RS-232 by connecting a serial cable to the serial interface connector, referred to as the *Dataport*. Eight Telnet sessions can be active at the same time in addition to the RS-232 connection.

### 2.1.2 Root commands

For each of the XML documents supported by the system, there is a corresponding XACLI root command. The root command has the same name as the corresponding XML document, except that the root command is prefixed by an “x”:

XML document	XACLI root command
status.xml	xstatus
configuration.xml	xconfiguration
command.xml	xcommand

The information in the TANDBERG XML engine, is divided into three main types: *Status Information*, *Configuration Information* and *Command Information*, ref. the documentation of the TANDBERG XML API.

As there is a fundamental difference in these three main types of information, there is also three different ways of working with the information using XACLI.

### 2.1.3 Addressing

XACLI supports XPath for addressing Status Information and Configuration Information. In addition there is support for the proprietary TANDBERG SimplePath notation. With SimplePath notation an element or a group of elements are addressed by supplying a space separated list of element names (elemName) and optional element instance numbers (item):

```
<elemName> [item] <elemName> [item] ...
```

If the instance number of a given element is omitted, the expression addresses all instances of this element

#### Example 2.1

To address the BaudRate sub-element of SerialPort 2:

XPath:

```
SerialPort[2]/BaudRate
```

SimplePath:

```
SerialPort 2 BaudRate
```

To address the BaudRate sub-element of all SerialPort elements:

XPath:

```
SerialPort/BaudRate
```

SimplePath:

```
SerialPort BaudRate
```

### 2.1.4 Exposure options

By adding an exposure option after the address (XPath or SimplePath) expression, the system can be instructed to return only parts of the information within an element structure.

```
< root command> <address expression> <exposure option>
```

**Supported exposure options:**

- “-“ hides all value elements
- “--“ hides all sub-elements

**Example 2.2**

Request for *Call 1* element with no exposure option

```
xstatus call 1
```

```
*s Call 1 (direction=Incoming):
  State: Connected
  RemoteNumber: "9798906"
  Duration: 3609
  Channel 1 (type=Outgoing):
    Rate: 768
    Audio:
      Protocol: G722
      Rate: 64
    Video:
      Protocol: H264
      Resolution: CIF
      Rate: 255
  Channel 2 (type=Incoming):
    Rate: 768
    Audio:
      Protocol: G722
      Rate: 64
    Video:
      Protocol: H264
      Resolution: SIF
      Rate: 223
*s/end
```

Request for *Call 1* element with exposure option “-“:

```
xstatus call 1 -
```

```
*s Call 1 (direction=Incoming):
  Channel 1 (type=Outgoing):
    Audio:
    Video:
  Channel 2 (type=Incoming):
    Audio:
    Video:
*s/end
```

Request for *Call 1* element with exposure option “--“:

```
xstatus call 1 --
```

```
*s Call 1 (direction=Incoming):
*s/end
```

## 2.1.5 Misc

The XACLI interface is not case sensitive. XACLI allows using only partial names.

## 2.2 The Status-type root command – xstatus

The information accessible through this command is the exact same information that is available in the corresponding XML document.

To get an overview of accesible top-level elements within a status-type root command, type `?` or `help` after the status-type root command.

### Example 2.3

```
xstatus ?

- Status -

Call [1..20]           Screensaver
CallManager [1..7]    SystemUnit
DNS                   TFTP
Ethernet              URL
IP                    Video

OK
```

To access status-type data, simply type the status-type root command (`xstatus`) and then an XPath address expression or a TANDBERG SimplePath expression:

```
<status-type root command> <address expression>
```

### Example 2.4

```
xstatus call 1 remotenumber

*s Call 1 (direction=Incoming):
  RemoteNumber: "9798906"
*s/end

OK
```

### 2.2.1 Format

Status information is presented by a markup notation, similar to XML.

Main differences:

- all braces are removed in the XACLI format
- XACLI is not using end-tags, except for a tag to mark end of top element
- XACLI is using indent spaces to present the data structure
- XACLI hides instance number (*item* number in XML) of an element if there only exist one instance of a given element
- A status top level element starts with “\*s”

The below example shows XML formatting and XACLI formatting for the same status element, *IP*.

### Example 2.5

```
XML:
<Status>
```

```
<IP item="1">  
  <Address item="1">10.47.8.20</Address>  
  <SubnetMask item="1">255.255.248.0</SubnetMask>  
  <Gateway item="1">10.47.8.1</Gateway>  
</IP>  
</Status>
```

**XACLI:**

```
*s IP:  
  Address: "10.47.8.20"  
  SubnetMask: "255.255.248.0"  
  Gateway: "10.47.8.1"  
*s/end
```

**NOTE!** To write a parser for the XACLI format, the parser must keep track of the levels by counting white spaces. The indent is increased by two whitespaces for each level.

## 2.3 The Configuration-type root command - xconfiguration

The information accessible through this command is the exact same information that is available in the corresponding XML document.

To get an overview of accessible top-level configuration elements, type `?` or `help` after the configuration-type root command:

```
<configuration-type root command> ?
```

### Example 2.6

```
xconfiguration ?
```

```
- User Configurations -  
AlertSpeaker      Screensaver  
AlertTone         SerialPort  
Audio             StrictPassword  
Camera            SystemUnit  
DNS               TFTP  
Ethernet          Video  
IP                WLAN  
OptionKey
```

```
OK
```

### 2.3.1 Configuration help

To get help on configurations, type the configuration-type root command – then an address expression followed by `?` or `help`. The possible values for the elements matching the address expression will be returned.

```
<configuration-type root command> <address expr> ?/help
```

### Example 2.7

User wants to configure IP:

```
xconfiguration ip ?  
*h xConfiguration IP Assignment: <DHCP/Static>  
*h xConfiguration IP Address: <IPAddr>  
*h xConfiguration IP SubnetMask: <IPAddr>  
*h xConfiguration IP Gateway: <IPAddr>
```

**NOTE!** Only typing `xconfiguration ?` actually addresses all configuration elements within the `xconfiguration` root command. One would therefore expect that help on all configurations would be returned. But as described above this is a special case and only a listing of the top level elements are returned. To get help on all configurations supported by the system, type:

```
xconfiguration // ?
```

### 2.3.2 Configuration read

To read configurations, type the configuration-type root command followed by an address expression:

```
<configuration-type root command> <address expr>
```

### Example 2.8

User wants to read IP configurations:

```
xconfiguration ip
*c xConfiguration IP Assignment: Static
*c xConfiguration IP Address: "10.47.8.20"
*c xConfiguration IP SubnetMask: "255.255.248.0"
*c xConfiguration IP Gateway: "10.47.8.1"
```

OK

### 2.3.3 Configuration set (write)

To set configurations, the address expression following the configuration-type root command must end with a colon. The value to be set must be added after the colon:

```
<configuration-type root command> <address expr>: value
```

**NOTE!** It is required to have a whitespace between the colon and the value.

### Example 2.9

User wants to set IP assignment:

```
xconfiguration ip assignment: static
or
xconfiguration ip/assignment: static
```

## 2.4 The Command-type root commands - xcommand

To get an overview of the supported commands within a command-type root command, type ? or help after the command-type root command.

```
<command-type root command> ?
```

### Example 2.10

```
xcommand ?
```

```
- User Commands -
```

```
Boot                               ScreensaverActivate
DefaultValuesSet                   ScreensaverDeactivate
KeyPress                            ScreensaverReset
KeyRelease
```

```
OK
```

To list usage for all commands with parameters, type a double question mark after the command-type root command.

```
<command root command> ??
```

### Example 2.11

```
xcommand ??
```

### 2.4.1 Command help

To get help on a specific command, type the command-type root command – then a command name followed by ? or help:

```
<command-type root command> <command name> ?
```

### Example 2.12

```
xcommand KeyPress ?
```

```
*h xCommand KeyPress
      Key(r):
<0/1/2/3/4/5/6/7/8/9/Softkey1/Softkey2/Softkey3/Softkey4/
Softkey5/Line/Layout/Brightness/Up/Down/Right/Left/Selfview/PIP/Help/
VoiceMail/Phonebook/Services/Settings/Cancel/MicrophoneOff/VideoOff/
Star/Square/Speaker/Headset/VolumeUp/VolumeDown/OK>
```

```
OK
```

**NOTE!** Required parameters are identified by an “(r)” behind the parameter name.

### 2.4.2 Issuing a command

A command must start with a command-type root command, followed by a command name, followed by a set of parameters. Parameters values can either be specified by a markup

notation or by placing the parameter values in the sequence specified by the help text – or a combination of these methods.

### Markup notation

```
<command-type root command> <command> <parameter:value> <parameter:value>...
```

When using this notation, the sequence the parameters are entered is unessential:

#### Example 2.13

```
xcommand KeyPress Key: VolumeUp
```

Abbreviations can be used for the parameter names as long as the parameter names are unique within the command:

#### Example 2.14

```
xcom keyp k: VolumeUp
```

### Command response

When issuing a command, the system will return a set of return values, ref. the documentation of the TANDBERG XML API. The response will be on the same format as the standard XACLI Status format.

#### Example 2.18

```
xcommand KeyPress Key: VolumeUp
```

```
*r Result (status=OK): /  
*r/end
```

```
OK
```

**NOTE!** When using XACLI as a machine-machine interface it is recommended to use markup notation and always supply complete tag names.

## 2.5 XML Output - `xgetxml`

---

As an alternative to the standard XACLI output format, XML format is supported through the root command **`xgetxml`**. `xgetxml` takes an XPath expression as parameter and the elements (or complete document) matching the expression will be returned.

### Example 2.19

```
xgetxml status/ip
```

```
<Status>  
  <IP item="1">  
    <Address item="1">10.47.8.20</Address>  
    <SubnetMask item="1">255.255.248.0</SubnetMask>  
    <Gateway item="1">10.47.8.1</Gateway>  
  </IP>  
</Status>  
OK
```

## 3 API - Configurations

---

This section gives an overview of the Configuration Information available in the Configuration XML document (*configuration.xml*) and the Configuration root command (*xconfiguration*) of the XACLI interface.

All examples are presented using the standard XACLI format.

## 3.1 configuration.xml – xconfiguration

<b>AlertSpeaker</b>	<p><b>AlertSpeaker Mode: &lt;On/Off&gt;</b> Turns the internal alert speaker on or off.</p>
<b>AlertTone</b>	<p><b>AlertTone Volume: &lt;0..15&gt;</b> Sets the volume of the alert tone.</p> <p><b>AlertTone Tone: &lt;1..7&gt;</b> Sets the alert tone to use for incoming calls.</p>
<b>Audio</b>	<p><b>Audio Microphones Mode: &lt;On/Off&gt;</b> Turns Audio inputs on or off.</p> <p><b>Audio Volume: &lt;0..15&gt;</b></p> <p><b>Audio KeyTones: &lt;On/Off&gt;</b> Tones when pressing keys on remote control.</p>
<b>Camera</b>	<p><b>Camera [1..5] Brightness Mode: &lt;Manual/Auto&gt;</b> Sets whether to control the camera brightness manually or have it automatically set by the system.</p> <p><b>Camera [1..5] Brightness Level: &lt;0..16&gt;</b> Defines the brightness level to use if brightness mode is set to manual.</p>
<b>DNS</b>	<p><b>DNS Server [1..5]: &lt;IPAddr&gt;</b> Defines the address of the dns servers to use.</p> <p><b>DNS DomainName: &lt;domain name&gt;</b> Defines the domain name of the DNS server.</p>
<b>Ethernet</b>	<p><b>Ethernet Speed: &lt;Auto/10half/10full/100half/100full&gt;</b></p>
<b>IP</b>	<p><b>IP Assignment: &lt;DHCP/Static&gt;</b> Selects whether to use <i>Dynamic Host Configuration Protocol</i> or static IP address allocation for the system.</p> <p><b>IP Address: &lt;IPAddr&gt;</b> Defines the ip address for the system if static ip assignment is selected.</p> <p><b>IP SubnetMask: &lt;IPAddr&gt;</b> Defines the subnet mask for the system.</p> <p><b>IP Gateway: &lt;IPAddr&gt;</b> Defines the gateway address to use.</p>
<b>OptionKey</b>	<p><b>OptionKey Bandwidth: &lt;S: 0, 16&gt;</b></p>

	Defines the maximum bandwidth for a call.
<b>Screensaver</b>	<p><b>Screensaver Mode: &lt;On/Off&gt;</b> Enables/disables screensaver to be activated after a delay specified by the "Screensaver Delay" parameter.</p> <p><b>Screensaver Delay: &lt;1..480&gt;</b> Defines screensaver delay in minutes.</p>
<b>SerialPort</b>	<p><b>SerialPort [1..2] BaudRate: &lt;1200/2400/4800/9600/19200/38400/57600/115200&gt;</b></p> <p><b>SerialPort [1..2] Parity: &lt;None/Odd/Even&gt;</b></p> <p><b>SerialPort [1..2] DataBits: &lt;7/8&gt;</b></p> <p><b>SerialPort [1..2] StopBits: &lt;1/2&gt;</b></p> <p><b>SerialPort 1 Mode: &lt;Data/Modem/Control/T120/Keyboard&gt;</b></p> <p><b>SerialPort 2 Mode: &lt;VISCA/Auto&gt;</b></p>
<b>StrictPassword</b>	<p><b>StrictPassword: &lt;On/Off&gt;</b> When set to on the "SystemUnit Password" must be at least 6 characters in length and contain a mix of digits [0-9] and letters [a-z].</p>
<b>SystemUnit</b>	<p><b>SystemUnit Password: &lt;S: 0, 16&gt;</b> Password needed to access system via telnet. Disable password by setting StrictPassword to Off, and then set SystemUnit Password to "".</p>
<b>TFTP</b>	<p><b>TFTP Alternate: &lt;On/Off&gt;</b></p> <p><b>TFTP Server [1..2]:&lt;IPAddr&gt;</b></p>
<b>Video</b>	<p><b>Video Outputs ScreenFormatTV: &lt;4:3/16:9&gt;</b> Screen format of main (PAL/NTSC) monitor.</p> <p><b>Video Outputs ScreenFormatPC: &lt;4:3/16:9&gt;</b> Screen format of DVI/VGA monitor</p> <p><b>Video Outputs FormatPCWideScreen: &lt;Normal/Wide&gt;</b></p> <p><b>Video Outputs TV 1 OSD: &lt;On/Off&gt;</b> Switch menu on main monitor on/off.</p> <p><b>Video Outputs DVI OSD: &lt;On/Off&gt;</b> Switch menu on DVI/VGA monitor on/off.</p>
<b>WLAN</b>	<p><b>xConfiguration WLAN Mode: &lt;Managed/Adhoc&gt;</b> Use Adhoc when not communicating with an access point. Use Managed when communication is made through access point.</p> <p><b>xConfiguration WLAN SSID: &lt;S: 0, 32&gt;</b> Example "WLANNETWORK". Defines a local network id for this</p>

wireless region. It must be the same for all end points and the access point.

**xConfiguration WLAN Community: <S: 0, 32>**

Optional.

Community can be used when connecting to an access point where the SSID is the same. Example "Unit2".

**xConfiguration WLAN Key [1..4]: <S: 0, 26>**

The 64-bit keys can consist of a leading star (\*) and 5 characters.

The 128-bit key can consist of a leading star (\*) and 13 characters.

Start with a \* and then the text. Example: 128 bit key: \*secretkeyhome.

Encryption using Hex numbers The 64-bit keys can consist of 10 hexadecimal digits.

Example:

"de01ad4dbe". The 128-bit key can consist of 26 hex numbers.

**xConfiguration WLAN UseKey: <1..4>**

Select which of the keys shown below you want to use.

**xConfiguration WLAN Encryption: <Off/64/128>**

Select if you want to encrypt your Wireless LAN connection. Increased encryption level will decrease performance.

**xConfiguration WLAN Enable: <On/Off>**

## 4 API - Commands

---

This section gives an overview of the supported system Commands.  
All examples are presented using the standard XACLI format.

## 4.1 commands.xml - xcommands

<p><b>Boot</b></p>	<p>Command used to reboot the system.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>ParameterRestore:</b> &lt;On/Off&gt; When rebooting the system after software upgrade, all configurations will be restored. By setting ParameterRestore to off, the system configurations prior to software upgrade will be lost.</li> </ul> <p><b>OK Result parameters:</b> None</p> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause:</b> &lt;1...&gt; Cause code specifying why the command was not accepted by the system</li> <li>• <b>Description</b> Textual description of the cause code.</li> </ul> <p><b>Example:</b></p> <pre>xcommand boot  *r Result (status=OK): *r/end  OK</pre>
<p><b>DefaultValuesSet</b></p>	<p>Command used to reset configurations to default values.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Level:</b> &lt;1..3&gt; Configurations are divided into three different storage classes. The level parameter denotes that configurations on this level and all levels below are to be reset.</li> </ul> <p><b>OK Result parameters:</b> None</p> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause:</b> &lt;1...&gt; Cause code specifying why the command was not accepted by the system</li> <li>• <b>Description</b> Textual description of the cause code.</li> </ul> <p><b>Example:</b></p> <pre>xcommand defaultvalueset level:2  *r Result (status=OK): *r/end  OK</pre>
<p><b>KeyPress</b></p>	<p>Command used to emulate a keypress.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>0..9</b></li> <li>• <b>Softkey1..Softkey5</b></li> <li>• <b>Line</b></li> <li>• <b>Layout</b></li> <li>• <b>Brightness</b></li> <li>• <b>Up, Down, Left, Right</b></li> </ul>

	<ul style="list-style-type: none"> <li>• <b>Selfview</b></li> <li>• <b>PIP</b></li> <li>• <b>Help</b></li> <li>• <b>VoiceMail</b></li> <li>• <b>PhoneBook</b></li> <li>• <b>Services</b></li> <li>• <b>Settings</b></li> <li>• <b>Cancel</b></li> <li>• <b>MicrophoneOff</b></li> <li>• <b>VideoOff</b></li> <li>• <b>Star</b></li> <li>• <b>Square</b></li> <li>• <b>Speaker</b></li> <li>• <b>Headset</b></li> <li>• <b>VolumeUp</b></li> <li>• <b>VolumeDown</b></li> <li>• <b>OK</b></li> </ul>
<b>KeyRelease</b>	Command used to emulate key release. Used together with KeyPress.
<b>ScreensaverActivate</b>	<p>Command used to activate screensaver.</p> <p><b>Parameters:</b> None</p> <p><b>OK Result parameters:</b> None</p> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause:</b> &lt;1...&gt; Cause code specifying why the command was not accepted by the system</li> <li>• <b>Description</b> Textual description of the cause code.</li> </ul> <p><b>Example:</b></p> <pre>xcommand screensaveractivate  *r Result (status=OK): *r/end  OK</pre>
<b>ScreensaverDeactivate</b>	<p>Command used to deactivate screensaver.</p> <p><b>Parameters:</b> None</p> <p><b>OK Result parameters:</b> None</p> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause:</b> &lt;1...&gt; Cause code specifying why the command was not accepted by the system</li> <li>• <b>Description</b> Textual description of the cause code.</li> </ul> <p><b>Example:</b></p> <pre>xcommand screensaverdeactivate  *r Result (status=OK): *r/end  OK</pre>

## ScreensaverReset

Command used to reset the screensaver timer.

### Parameters:

- **Delay(r):** <1..480> Specifies the screensaver delay in minutes.

### OK Result parameters:

None

### ERROR Result parameters:

- **Cause:** <1...> Cause code specifying why the command was not accepted by the system
- **Description** Textual description of the cause code.

### Example:

```
xcommand screensaverreset delay:90
```

```
*r Result (status=OK):  
*r/end
```

OK

## 5 API - Status

---

This section gives an overview of the Status Information available in the Status XML document (status.xml) and the Status root command (xstatus) of the XACLI interface.

All examples are presented using the standard XACLI format.

## 5.1 status.xml – xstatus

### Call [1..20]

#### Top level attributes:

- **state:** OnHook/OffHook/RingOut/RingIn/Connected/Busy/Congestion/Hold/CallWaiting/CallTransfer/CallPark/Proceed/CallRemoteMultiline/InvalidNumber
- **direction:** Incoming/Outgoing

#### Summary:

- Returns all currently available information for a call.

#### Examples:

```
*s Call 1 (direction=Incoming):
  State: Connected
  RemoteNumber: "9798906"
  Duration: 14
  Channel 1 (type=Outgoing):
    Rate: 768
    Audio:
      Protocol: G722
      Rate: 64
    Video:
      Protocol: H264
      Resolution: CIF
      Rate: 291
  Channel 2 (type=Incoming):
    Rate: 768
    Audio:
      Protocol: G722
      Rate: 64
    Video:
      Protocol: H264
      Resolution: SIF
      Rate: 159
*s/end

*s Call 2 (direction=Outgoing):
  State: Hold
  RemoteNumber: "1028"
  Duration: 25
  Channel 1 (type=Outgoing):
    Rate: 0
    Audio:
      Protocol: Off
      Rate: 0
    Video:
      Protocol: Off
      Resolution: Off
      Rate: 0
  Channel 2 (type=Incoming):
    Rate: 0
    Audio:
      Protocol: Off
      Rate: 0
    Video:
```

	<pre> Protocol: Off Resolution: Off Rate: 0 *s/end  *s Call 3: State: OnHook *s/end  *s Call 4: State: OnHook *s/end </pre>
<p><b>CallManager [1..7]</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns name of CallManager</li> <li>• Returns CallManager state</li> </ul> <p><b>Example</b></p> <pre> xstatus callmanager  *s CallManager 1: Name: "RD-CCM1" State: Active *s/end  *s CallManager 2: Name: "RD-CCM2" State: Standby *s/end </pre>
<p><b>DNS</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns ip address of dns server 1..5.</li> <li>• Returns dns domain name</li> </ul> <p><b>Example</b></p> <pre> xstatus dns  *s DNS: Server 1: Address: "10.0.0.10" Server 2: Address: "10.0.0.2" Server 3: Address: "0.0.0.0" Server 4: Address: "0.0.0.0" Server 5: Address: "0.0.0.0" DomainName: "lysaker.tandberg.net" *s/end  OK </pre>

<p><b>Ethernet</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns MAC Address</li> <li>• Returns Ethernet speed</li> </ul> <p><b>Example</b></p> <pre>*s Ethernet:   MacAddress: "00:50:60:01:A4:EB"   Speed: 100full *s/end</pre>
<p><b>IP</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns current IP address, Subnet Mask and Gateway address</li> </ul> <p><b>Example</b></p> <pre>*s IP:   Address: "10.47.8.26"   SubnetMask: "255.255.248.0"   Gateway: "10.47.8.1" *s/end</pre>
<p><b>Screen saver</b></p>	<p><b>Top level attributes:</b></p> <ul style="list-style-type: none"> <li>• <b>status: On/Off</b> Indicates wheter Screensaver is active or not</li> </ul> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns current Screensaver status and the current timer value (time left before the screensaver is activated if not already active)</li> </ul> <p><b>Examples:</b></p> <pre>*s Screensaver (status=On): / *s/end  *s Screensaver (status=Off):   Timer: 9 *s/end</pre>
<p><b>SystemUnit</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns information about the System Unit</li> </ul> <p><b>Example</b></p> <pre>*s SystemUnit:   ProductType: "TANDBERG Codec"   ProductID: "TANDBERG 1000MXP"   Uptime: 87522   Software:     Version: "M1.3"     Name: "test"     ReleaseDate: "2005-04-15 "   Configuration:     LANBandwidth: 768</pre>

	<pre>Hardware:   SerialNumber: "13A20019"   MainBoard: "101110 rev. 03"   BootSoftware: "Rev. 1.7, 2004-12-15"   Configuration:     TV-Standard: PAL   TemperatureCelcius: 40   TemperatureFahrenheit: 104 *s/end</pre>
<p><b>TFTP</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>Returns ip address of TFTP servers 1 and 2</li> </ul> <p><b>Example</b> xstatus tftp</p> <pre>*s TFTP:   Server 1:     Address: "10.47.9.17"   Server 2:     Address: "0.0.0.0" *s/end</pre> <p>OK</p>
<p><b>URL</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>List the URL's that point to Directories, Services, Messages and Information.</li> </ul> <p><b>Example</b> xstatus url</p> <pre>*s URL:   Info: "http://RD-CCM1/CCMCIP/         GetTelecasterHelpText.asp"   Directory: "http://RD-CCM1/CCMCIP/              xmldirectory.asp"   Message: ""   Services: "http://RD-CCM1/CCMCIP/             getservicesmenu.asp" *s/end</pre> <p>OK</p>
<p><b>Video</b></p>	<p><b>Top level attributes:</b> None</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>Gives information about video input format (PAL or NTSC)</li> </ul> <p><b>Example</b> xstatus video</p>

```
*s Video:  
  Input:  
  Format: NTSC  
*s/end
```