

---

# **TANDBERG MPS API**

## **User Manual**

TANDBERG  
D13639 Rev 02

---

## Table Of Contents

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 .....	9
1.5 Introduction to TANDBERG XML API Service (TXAS) .....	13
1.6 Exercises.....	15
2 The XML-based Advanced Command Line Interface .....	17
2.1 XACLI.....	18
2.2 The Status-type root commands – xstatus / xhistory.....	21
2.3 The Configuration-type root commands - xconfiguration/xdirectory .....	23
2.4 The Command-type root commands - xcommand .....	25
2.5 XML Output - xgetxml .....	28
2.6 Special Commands.....	29
3 API - Configurations.....	35
3.1 configuration.xml – xconfiguration .....	36
3.2 directory.xml – xdirectory .....	41
4 API - Commands .....	42
4.1 command.xml – xcommand .....	43
5 API - Status.....	53
5.1 status.xml – xstatus .....	54
5.2 history.xml – xhistory .....	62

# 1 The TANDBERG API

---

This document is a guide to the API interface of the TANDBERG MPS 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

# 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" status="Disconnected" type="NA" protocol="NA">
    <Cause item="1">255</Cause>
  </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" status="Synced" type="Vtlph" protocol="H323">
    <CallRate item="1">768</CallRate>
    <RemoteNumber item="1">10.47.15.127</RemoteNumber>
    <Channels item="1" type="Incoming">
      <Audio item="1" status="Active">
        <Protocol item="1">G722</Protocol>
        <Rate item="1">64</Rate>
      </Audio>
      <Video item="1" status="Active">
        <Protocol item="1">H263</Protocol>
        <Resolution item="1">CIF</Resolution>
        <Rate item="1">704</Rate>
      </Video>
      <Video item="2" status="Inactive" />
      <Data item="1" status="Inactive" />
    </Channels>
    <Channels item="2" type="Outgoing">
      <Audio item="1" status="Active">
        <Protocol item="1">G722</Protocol>
        <Rate item="1">64</Rate>
      </Audio>
      <Video item="1" status="Active">
        <Protocol item="1">H264</Protocol>
        <Resolution item="1">SIF</Resolution>
        <Rate item="1">704</Rate>
      </Video>
      <Video item="2" status="Inactive" />
      <Data item="1" status="Inactive" />
    </Channels>
  </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
```

### Example 1.8

XPath also supports addressing by putting constraints on element attributes. Lets go back to the `Call` element in [Example 1.2](#). The below expression will address the `CallRate` element of all *Synced* calls in a system:

```
Status/Call[@status="Synced"]/CallRate
```

To add more constraints on element attributes, XPath supports boolean expressions. To address all *Synced H323* calls in a system, the following expression can be used:

```
Status/Call[@status="Synced" AND @protocol="H323"]/CallRate
```

## 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.

### 1.3.3 Feedback

Feedback is an extremely powerful feature where the TANDBERG system actively returns updated status and configuration information to the user/control application whenever changes occur. The user/control application can specify what parts of the status and

configuration hierarchies it wants to monitor by using XPath. The user/control application can therefore limit the amount of information it receives from the target system to only those parts being of interest for the given application.

## 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
  - b. history.xml
2. **Configuration documents (rw)**: Documents holding all system configurations.  
Supported documents:
  - a. configuration.xml
  - b. directory.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" status="Synced" type="Vt1ph" protocol="H323"
direction="Outgoing">
  <CallRate item="1">768</CallRate>
  <RemoteNumber item="1">58458</RemoteNumber>
  <Mute item="1">Off</Mute>
  <Microphone item="1">Off</Microphone>
  <Duration item="1">15</Duration>
  <Channels item="1" type="Incoming">
    <Rate item="1">768</Rate>
    <Restrict item="1">Off</Restrict>
    <Encryption item="1" status="Off" />
    <Audio item="1" status="Active">
      <Protocol item="1">G722</Protocol>
      <Rate item="1">64</Rate>
      <RemoteIPAddress item="1" />
    </Audio>
  </Channels>
</Call>
```

```

    <LocalIPAddress item="1">10.47.8.41:2326</LocalIPAddress>
    <Encryption item="1" status="On">
      <Type item="1">AES-128</Type>
    </Encryption>
    <RSVP item="1">Off</RSVP>
  </RSVPRate item="1">0</RSVPRate>
  <DynamicRate item="1">64</DynamicRate>
  <TotalPackets item="1">367</TotalPackets>
  <PacketLoss item="1">0</PacketLoss>
  <Jitter item="1">0</Jitter>
</Audio>
.
.
.
</Call>
---
<Call item="2" status="Synced" type="Vt1ph" protocol="H320"
direction="Outgoing">
  <CallRate item="1">384</CallRate>
  <Bonding item="1">On</Bonding>
  <RemoteNumber item="1">8796</RemoteNumber>
  <RemoteNumber item="2" />
  <RemoteSubAddress item="1" />
  <Mute item="1">Off</Mute>
  <Microphone item="1">Off</Microphone>
  <LogTag item="1">25</LogTag>
  <Channels item="1" type="Incoming">
    <Rate item="1">384</Rate>
    <Restrict item="1">Off</Restrict>
    <Encryption item="1" status="Off" />
    <Audio item="1" status="Active">
      <Protocol item="1">G722</Protocol>
      <Rate item="1">56</Rate>
    </Audio>
    .
    .
    .
  </Call>
---
<Call item="6" status="Disconnected" type="NA" protocol="NA"
direction="NA">
  <Cause item="1">255</Cause>
</Call>

```

In the above example we see that the `Bonding` element, `RemoteNumber[2]` and `SubAddress` is not present for H323 calls. On the other hand, for H323 calls, the `Audio` channel element holds information regarding packetloss etc., which is not present for H320 calls. If the call is disconnected, the `Call` element only contains the disconnect cause value.

### 1.4.3 Configuration documets (rw)

The structure of the Configuration documents are 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 `Dial` is defined to take five parameters, while only the `Number` parameter is required as specified by the attribute `required`. The value domain for the parameters are referenced by the attribute `valueSpaceRef`.

```
<Command>
  <Dial item="1">
    <Number item="1" required="True"
valueSpaceRef="/ValueSpace/RemoteNumber"/>
    <SubAddress item="1" required="False"
valueSpaceRef="/ValueSpace/SubAddress"/>
    <CallRate item="1" required="False"
valueSpaceRef="/ValueSpace/Bandwidth"/>
```

```

    <Restrict item="1" required="False"
valueSpaceRef="/ValueSpace/OnOff"/>
    <NetProfile item="1" required="False"
valueSpaceRef="/ValueSpace/NetprofileRef"/>
  </Dial>
</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 place a call to number 999 the user can simply post the following XML structure to the system:

```

<Command>
  <Dial item="1">
    <Number item="1">999</Number>
  </Dial>
</Command>

```

When issuing Commands, the system will return an XML structure in response. The response structure will have the same name as the command issued, but it will be postfixed with "Result". All commands will have an attribute named *status*, stating whether the command was accepted or not. If a command is not accepted, the response structure will contain a cause code. If the command is accepted, the response structure may contain information relevant for the specific command.

### Example 13

The Dial command in the above example may return the following response structure:

```

<Command>
  <DialResult item="1" status="OK">
    <CallRef item="1">1</CallRef>
    <LogTag item="1">6</LogTag>
  </DialResult>
</Command>

```

The response structure for the Dial command, *DialResult*, states that the command was accepted by the system. In addition to stating that the command was accepted, the Dial command returns the elements *CallRef* and *LogTag*. This lets the user identify/trace the call in the Status documents (*status.xml* and *history.xml*).

### Example 14

Below is an example of the Dial command not being accepted by the system:

```

<Command>
  <DialResult item="1" status="Error">
    <Cause item="1">17</Cause >
    <Description item="1">Too much bandwidth requested</Description >
  </DialResult>
</Command>

```

## 1.5 Introduction to TANDBERG XML API Service (TXAS)

---

TXAS is a service provided by TANDBERG units for transmitting and receiving (transceiving) information encoded in XML format.

The API uses HTTP(S) as the transport mechanism and connects to the normal web port (80). TXAS can be accessed in two ways; bare-bone HTTP requests where URL's uniquely identifies the request, and SOAP where a single URI is used but the request itself is encoded with XML.

### 1.5.1 Bare-bone HTTP(S) access

The bare-bone HTTP mode uses a unique URL to identify the specific request. The contents of the HTTP body will be a XML document (or part of it).

Bare-bone HTTP(S) access is accomplished by passing arguments in the query string (after '?' in URL) in a GET request, or using the "application/x-www-form-urlencoded" content-type method of POSTing form data (Each argument starts with a name '=' and a value, and every parameter separated with '&' (and opt NL).)

#### **getxml**

REQUEST:

/getxml

PARAM:

location = XPath expression

"/getxml" request returns an XML document based on the location parameter passed to the request. The elements (or complete document) matching the expression will be returned. On Incorrect XPath expression, a <Fault> element with a <XPathError> element will be returned.

#### **formputxml**

REQUEST:

/formputxml

PARAM:

xmldoc = "an XML document of Configuration, Directory or Command"

This is most useful in a POST (to extend character limit of 255 of GET urls). It posts a Configuration or Command document to set the configurations or issue a command. Like getxml, it has the data URL form-data encoded with one single parameter. The Content-Type of the document must be of type "application/x-www-form-urlencoded" and the body must be encoded accordingly (e.g. first line will be xmldoc=<then the document>).

#### **putxml**

REQUEST:

/putxml

PARAM:

HTTP BODY as argument

Putxml is like "formputxml", put uses the complete BODY as argument (i.e. the content of the xmldoc parameter). The Content-type should be "text/xml" or "application/xml" ( or "text/plain"), though no check at the moment. (Except for application/x-www-form-urlencoded which will cause a failure).

## 1.5.2 SOAP

SOAP is described in a separate section.

## 1.6 Exercises

---

The exercises in this section are based on using a TANDBERG 6000 MXP codec and Microsoft Internet Explorer. Some of the examples may however also apply to other systems and other browsers.

**NOTE!** Replace the ip address, 10.47.8.41, in the below examples with the ipaddress of your system.

### Exercise 1

The examples in this exercise shows how to read the supported XML documents from the system using a web browser.

Enter the following address in the browsers address field:

```
http://10.47.8.41/status.xml
http://10.47.8.41/history.xml
http://10.47.8.41/configuration.xml
http://10.47.8.41/directory.xml
http://10.47.8.41/command.xml
http://10.47.8.41/valuespace.xml
```

### Exercise 2

This exercise shows how to use *getxml* to read the supported XML documents from the system. Enter the following expressions in the the browsers address field (NOTE! The first letter in the document names are uppercase):

```
http://10.47.8.41/ getxml?location=Status
http://10.47.8.41/ getxml?location=History
http://10.47.8.41/ getxml?location=Configuration
http://10.47.8.41/ getxml?location=Directory
http://10.47.8.41/ getxml?location=Command
http://10.47.8.41/ getxml?location=ValueSpace
```

### Exercise 3

This exercise shows shows how to use XPath expressions to read subsets of the XML documents.

```
http://10.47.8.41/getxml?location=Status/SystemUnit
http://10.47.8.41/getxml?location=Configuration/SerialPort/BaudRate
http://10.47.8.41/getxml?location=ValueSpace/SerialPortBaudrate[@item='1']
http://10.47.8.41/getxml?location=Configuration//Mode
http://10.47.8.41/getxml?location=Command/Dial
```

### Exercise 4

The address: <http://10.47.8.41/xmlput.ssi> contains an editor where XML data can be edited and then posted to the system by pressing the save button. Below are examples of XML structures to be posted to the system:

```
<Configuration>
  <SerialPort>
    <BaudRate>19200</BaudRate>
  </SerialPort>
</Configuration>
```

---

```
<Configuration>
```

```
<SerialPort>  
  <BaudRate>2400</BaudRate>  
</SerialPort>  
<Conference>  
  <H263>Off</H263>  
  <Downspeed>Off</Downspeed>  
</Conference>  
</Configuration>
```

---

```
<Command>  
  <Dial>  
    <Number>10.47.8.42</Number>  
  </Dial>  
</Command>
```

---

```
<Command>  
  <DisconnectCall/>  
</Command>
```

## 2 The XML-based Advanced Command Line Interface

---

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, in oposite to 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
history.xml	xhistory
configuration.xml	xconfiguration
directory.xml	xdirectory
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 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  CallRate: 768
  RemoteNumber: "10.47.15.127"
  Mute: Off
  Microphone: Off
  Duration: 10
  Channels 1 (type=Incoming):
    Rate: 768
    Restrict: Off
    Encryption (status=Off): /
    Audio (status=Active):
      Protocol: G722
      Rate: 64
    Video 1 (status=Active):
      Protocol: H263
      Resolution: CIF
      Rate: 704
    Video 2 (status=Inactive): /
    Data (status=Inactive): /
  Channels 2 (type=Outgoing):
    Rate: 768
    Restrict: Off
    Encryption (status=Off): /
    Audio (status=Active):
      Protocol: G722
      Rate: 64
    Video 1 (status=Active):
      Protocol: H263+
      Resolution: ICIF
      Rate: 704
    Video 2 (status=Inactive): /
    Data (status=Inactive): /
*s/end
```

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

**xstatus call 1 -**

```
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  Channels 1 (type=Incoming):
    Encryption (status=Off): /
    Audio (status=Active):
    Video 1 (status=Active):
    Video 2 (status=Inactive): /
    Data (status=Inactive): /
```

```
Channels 2 (type=Outgoing):  
  Encryption (status=Off): /  
  Audio (status=Active):  
  Video 1 (status=Active):  
  Video 2 (status=Inactive): /  
  Data (status=Inactive): /  
*s/end
```

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

```
xstatus call 1 --
```

```
*s Call 1 (status=Synced, type=Vt1ph, protocol=H323,  
direction=Outgoing):  
*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 commands – xstatus / xhistory

The information accessible through these commands is the exact same information that is available in the corresponding XML documents.

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

### Example 2.3

```
xstatus ?
- Status -
Audio                Feedback [1..3]
BRI [1..6]           G703
Call [1..11]         H323Gatekeeper
Camera [1..5]        IP
CameraTracking       PRI
Conference            Screensaver
Ethernet              SystemUnit
ExternalNetwork      VirtualMonitor [1..4]
FarEndInformation
```

OK

To access status-type data, simply type the status-type root command (`xstatus` or `xhistory`) 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 (status=Synced, type=Vt1ph, protocol=H323,
direction=Outgoing):
  RemoteNumber: "10.47.15.127"
*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 commands - xconfiguration/xdirectory

The information accessible through these commands is the exact same information that is available in the corresponding XML documents.

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

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

### Example 2.6

```
xconfiguration ?
```

```
- User Configurations -
AlertSpeaker      G703                OptionKey
AlertTone         H320                OSD
Audio            H323                PictureProgram [1..4]
AutoAnswer       H323CallSetup      Preset [1..15]
AutoPIP          H323Gatekeeper     QoS
Bonding          H323Prefix         RTP
CallManager      HTTP               Screensaver
Camera [1..5]    HTTPS             SelfViewOnStartup
CameraSleep      IP                 SerialPort [1..2]
CameraTracking   IPMedia           SNMP
Conference       IRControl         StillImageSource
CorporateDirectory ISDN              Streaming
DoNotDisturb    Keyboard          StrictPassword
DualMonitor     LocalLayout       SystemUnit
DuoVideoSource  Logo              T1
E1              LoS               Telnet
Ethernet        MainVideoSource   TelnetChallenge
ExternalNetwork NAT                Video
FECC            NetProfile [1..6] VNC
FTP
OK
```

```
xdirectory ?
```

```
- Directory -
GlobalEntry [1..400] LocalEntry [1..200]
GroupEntry [1..50]
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>
*h xConfiguration IP Password: <S: 0, 16>
```

**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 // ?
```

or

```
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
```

**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                DuoVideoStart      MessageBoxDisplay
CallAccept          DuoVideoStop       PIPHide
CallMute            FECCFocus          PIPShow
CameraBrightness   FECCMove           PresetActivate
CameraFocus         FECCPresetActivate PresetStore
CameraHalt          FECCPresetStore    ScreensaverActivate
CameraMove          FECCRequestStill    ScreensaverDeactivate
CameraPosition      FECCSelectSource   ScreensaverReset
CameraTrackingStart FeedbackDeregister  SiteDisconnect
CameraTrackingStop FeedbackRegister     SiteView
CameraWhiteBalance FloorRelease        SiteViewEnd
ChairRelease        FloorRequest        SPIDAutoConfigure
ChairTake           FloorToSite         StillImageSend
ConferenceDisconnect FloorToSiteEnd      StreamingStart
DefaultValuesSet   GroupEntryAdd       StreamingStop
Dial                GroupEntryDelete    TextDelete
DialGlobalEntry    LocalEntryAdd       TextDisplay
DialGroupEntry     LocalEntryDelete    VirtualMonitorReset
DialLocalEntry     MessageBoxDelete    VirtualMonitorSet
DisconnectCall
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 Dial ?
```

```
*h xCommand Dial
```

```
Number(r): <S: 0, 30>
SubAddress: <S: 0, 10>
CallRate: <1xh221/2xh221/64/128/256...
Restrict: <On/Off>
NetProfile: <1..6>
```

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 dial number:666 restrict:on callrate:128 subaddress:10
```

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

#### Example 2.14

```
xcommand dial nu:666 r:on c:128 s:10
```

If there are multiple instances of a parameter, the item number is added after the tag separated with a dot:

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

#### Example 2.15

```
xcommand groupentryadd name:TANDBERG localentryid.1:15
localentryid.2:57
```

### Sequence notation

<command-type root command> <command> <value> <value>...

When using this notation the parameter values must be entered in the sequence as stated in the help text:

#### Example 2.16

```
xcommand dial 666 10 128 on
```

### Combination

A combination of markup notation and sequence are also supported. The marked parameters will be assigned the user entered values first, then the system will assign the sequence entered parameters for the parameters not yet having been assigned a value:

### Example 2.17

```
xcommand dial 666 r:on 10 128
```

#### 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 dial 10.47.15.127
```

```
*r Result (status=OK):  
  CallRef: 1  
  LogTag: 6  
*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
```

## 2.6 Special Commands

In addition to the root commands described above, XACLI support a set of root commands that only applies to the Telnet session or RS232 session from where they are issued. This lets the user/control application individually configure the session(s) in use.

Supported special commands:

- `xfeedback` (not supported on all platforms)
- `xpreferences`

### 2.6.1 `xfeedback`

The special command `xfeedback` lets the user register user defined XPath expressions (with possible *exposure options*) to monitor changes in the XML/XACLI data. Whenever there is a change in one or more elements addressed by a registered XPath expression, the part of the element structure containing these changes will be returned. The system supports a total of 20 registered expressions, with a total of 15 expressions for one session.

**`xfeedback ?`**

```
usage: xfeedback register <XPathExpression>
or:    xfeedback deregister <index>
or:    xfeedback list
```

-

(note: deregistration with index=0 will deregister all registered expressions)

OK

#### Example 2.20

User wants to monitor changes in audio protocols for all active calls:

```
xfeedback register status/call/channels/audio/protocol
```

To view registered expressions:

```
xfeedback list
```

```
*xf 1 status/call/channels/audio/protocol
```

OK

The call changes audio protocol from G722 to G728 on incoming audio channel on call 1:

```
*s Call 1 (status=Synced, type=Vt1ph, protocol=H323,
direction=Outgoing):
```

```
  Channels 1 (type=Incoming):
```

```
    Audio (status=Active):
```

```
      Protocol: G728
```

```
*s/end
```

When changing back to G722:

```
*s Call 1 (status=Synced, type=Vt1ph, protocol=H323,
direction=Outgoing):
```

```
  Channels 1 (type=Incoming):
```

```
    Audio (status=Active):
```

```
      Protocol: G722
```

```
*s/end
```

### Example 2.21

*Exposure options* are also supported together with feedback.  
User only wants to monitor call setup progression.

```
xfeedback register status/call--
```

```
OK
```

```
xcom dial 10.47.15.127
```

```
*s Call 1 (status=EstablOut, type=Vtlph, protocol=H323,  
direction=Outgoing):  
*s/end
```

```
*r Result (status=OK):  
    CallRef: 1  
    LogTag: 3  
*r/end
```

```
OK
```

```
*s Call 1 (status=Alerting, type=Vtlph, protocol=H323,  
direction=Outgoing):  
*s/end
```

```
CONNECT
```

```
*s Call 1 (status=Syncing, type=Vtlph, protocol=H323,  
direction=Outgoing):  
*s/end
```

```
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,  
direction=Outgoing):  
*s/end
```

### Example 2.22

User only wants to know when calls are connected and disconnected:

```
xfeedback register status/call[@status="Synced"]--
```

```
OK
```

```
xfeedback register status/call[@status="Disconnected"]--
```

```
OK
```

```
xcom dial 10.47.15.127
```

```
*r Result (status=OK):  
    CallRef: 1  
    LogTag: 4  
*r/end
```

```
OK
```

```
CONNECT
```

```
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,  
direction=Outgoing):  
*s/end
```

**xcom disc**

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

OK

NO CARRIER
*s Call 1 (status=Disconnected, type=NA, protocol=NA, direction=NA):
*s/end
```

When conditional XPath expressions are used, the system will provide feedback on all elements within the address the first time the condition is true.

**Example 2.23**

User wants to monitor call changes only when the call is *Synced*. By registering the below expression, the system will not provide feedback on the call before it reaches the *Synced* state. When it first enters the *Synced* state it will provide status for the complete call. After this, the system will only give feedback on elements changing values (provided that the call is still *Synced*).

```
xfeedback register status/call[@status="Synced"]
```

```
OK
```

```
xcom dial 10.47.15.127
```

```
*r Result (status=OK):
  CallRef: 1
  LogTag: 5
*r/end

OK

CONNECT
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  CallRate: 768
  RemoteNumber: "10.47.15.127"
  Mute: Off
  Microphone: Off
  Duration: 0
  Channels 1 (type=Incoming):
    Rate: 768
    Restrict: Off
    Encryption (status=Off): /
    Audio (status=Active):
      Protocol: G722
      Rate: 64
    Video 1 (status=Active):
      Protocol: H263
      Resolution: CIF
      Rate: 704
    Video 2 (status=Inactive): /
    Data (status=Inactive): /
  Channels 2 (type=Outgoing):
    Rate: 768
    Restrict: Off
    Encryption (status=Off): /
```

```

Audio (status=Active):
  Protocol: G722
  Rate: 64
Video 1 (status=Active):
  Protocol: H263+
  Resolution: ICIF
  Rate: 704
Video 2 (status=Inactive): /
Data (status=Inactive): /
*s/end

```

...suddenly there is a change in audio protocol:

```

*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  Channels 1 (type=Incoming):
    Rate: 704
  Audio (status=Inactive): /
*s/end
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  Channels 1 (type=Incoming):
    Rate: 720
  Audio (status=Active):
    Protocol: G728
    Rate: 16
*s/end

```

## 2.6.2 xpreferences

The special command *xpreferences* lets the user/control application individually configure the Telnet/RS-232 session in use.

### **xpreferences ?**

```

usage: xpreferences xpathwrite <on/off>
or:    xpreferences detaillevel <1..2>
or:    xpreferences xmlconfigfeedback <on/off>
or:    xpreferences xmlstatusfeedback <on/off>
or:    xpreferences xmlcommandresult <on/off>

```

OK

```
xpreferences xpathwrite <on/off>
```

Disables/enables the XPath engine when issuing configurations. When the XPath engine is disabled, the user/control application must supply the complete path to the configurations to be set (no “double slashes” allowed). This will improve the performance of the system when issuing many consecutive configurations.

**NOTE!** It is always recommended to supply the complete path for configurations to be set when issuing commands from an external control application.

```
xpreferences detaillevel <1..2>
```

Most information elements accessible by the status-type root commands are defined to be level 1 information. However there are some information elements defined to be level 2 information. When reading status information, only the information elements with a detail level equal to or less than the detaillevel defined for the interface will be listed.

### **Example 2.24**

```

xstat call 1 channels 1 audio

*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  Channels 1 (type=Incoming):
    Audio (status=Active):
      Protocol: G722
      Rate: 64
*s/end

OK

xpreferences detaillevel 2

OK

xstat call 1 channels 1 audio

*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing, logTag=3):
  Channels 1 (type=Incoming):
    Audio (status=Active):
      Protocol: G722
      Rate: 64
      RemoteIPAddress: ""
      LocalIPAddress: "10.47.8.28:2326"
      Encryption (status=On):
        Type: AES-128
      RSVP: Off
      RSVPRate: 0
      DynamicRate: 64
      TotalPackets: 1618
      PacketLoss: 0
      Jitter: 0
*s/end

OK

```

`xpreferences xmlconfigfeedback <on/off>`

If `xmlconfigfeedback` is set to on, feedback on configurations will be returned in XML-format instead of the standard XACLI configuration format.

### Example 2.25

XACLI-format:

```
*c xConfiguration SerialPort 1 BaudRate: 2400
```

XML-format:

```

<Configuration>
  <SerialPort item="1">
    <BaudRate item="1">2400</BaudRate>
  </SerialPort>
</Configuration>

```

`xpreferences xmlstatusfeedback <on/off>`

If `xmlstatusfeedback` is set to on, all status feedback will be returned in XML-format instead of the standard XACLI status format.

### Example 2.26

XACLI-format:

```
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing):
  Channels 1 (type=Incoming):
    Rate: 736
    Audio (status=Active):
      Protocol: G722_1
      Rate: 32
*s/end

XML-format:
<Status>
  <Call item="1" status="Synced" type="Vtlph" protocol="H323"
direction="Outgoing">
    <Channels item="1" type="Incoming">
      <Rate item="1">768</Rate>
      <Audio item="1" status="Active">
        <Protocol item="1">G722</Protocol>
        <Rate item="1">64</Rate>
      </Audio>
    </Channels>
  </Call>
</Status>
```

xpreferences xmlcommandresult <on/off>

If *xmlcommandresult* is set to on, response for commands will be returned in XML-format.

### Example 2.27

```
XACLI-format:
xcom dial 10.47.15.127

*r Result (status=OK):
  CallRef: 1
  LogTag: 8
*r/end

XML-format:
xcom dial 10.47.15.127

<Result item="1" status="OK">
  <CallRef item="1">1</CallRef>
  <LogTag item="1">8</LogTag>
</Result>
```

## 3 API - Configurations

---

This section gives an overview of the Configuration Information available in the Configuration XML documents (*configuration.xml* / *directory.xml*) and the Configuration root commands (*xconfiguration* / *xdirectory*) of the XACLI interface.

All examples are presented using the standard XACLI format.

## 3.1 configuration.xml – xconfiguration

Conference	<p>Conference [1..12] Numbers E164Alias: &lt;E164: 0, 31&gt;</p> <p>Conference [1..12] Numbers PRI: &lt;S: 0, 24&gt;</p>
ConferenceTemplate	<p>ConferenceTemplate Name: &lt;S: 0, 30&gt;</p> <p>ConferenceTemplate CallRate: &lt;1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/Tlph/H0&gt;</p> <p>ConferenceTemplate Restrict: &lt;On/Off&gt;</p> <p>ConferenceTemplate MaxVideoSites: &lt;0..48&gt;</p> <p>ConferenceTemplate MaxAudioSites: &lt;0..48&gt;</p> <p>ConferenceTemplate Password: &lt;S: 0, 8&gt;</p> <p>ConferenceTemplate Encryption: &lt;On/Off&gt;</p> <p>ConferenceTemplate EncryptionType: &lt;DES/AES-128/Auto&gt;</p> <p>ConferenceTemplate PictureMode: &lt;Auto/4Split/9Split/16Split/5+1Split/7+1Split/VS/CPAuto&gt;</p> <p>ConferenceTemplate VideoFormat: &lt;Auto/Motion/Sharpness&gt;</p> <p>ConferenceTemplate CustomFormats: &lt;On/Off&gt;</p> <p>ConferenceTemplate AGC: &lt;On/Off&gt;</p> <p>ConferenceTemplate TelephoneFilter: &lt;On/Off&gt;</p> <p>ConferenceTemplate AllowIncomingCalls: &lt;On/Off&gt;</p> <p>ConferenceTemplate Duration: &lt;0..65534&gt;</p> <p>ConferenceTemplate EntryExitTones: &lt;On/Off&gt;</p> <p>ConferenceTemplate WelcomeMessage: &lt;On/Off&gt;</p> <p>ConferenceTemplate LegacyLevel: &lt;0..15&gt;</p> <p>ConferenceTemplate DuoVideo: &lt;On/Off&gt;</p> <p>ConferenceTemplate AudioG728: &lt;On/Off&gt;</p> <p>ConferenceTemplate CascadingPreference: &lt;Auto/Master/Slave&gt;</p>

	<p>ConferenceTemplate BillingCode: &lt;S: 0, 16&gt;</p> <p>ConferenceTemplate CPAutoSwitch: &lt;0..60&gt;</p> <p>ConferenceTemplate FloorToFull: &lt;On/Off&gt;</p> <p>ConferenceTemplate WebCallListTimeout: &lt;On/Off&gt;</p> <p>ConferenceTemplate NetworkId: &lt;1..2&gt;</p>
Ethernet	Ethernet [1..2] Speed: <Auto/10half/10full/100half/100full/None>
H323CallSetup	H323CallSetup [1..2] Mode: <Direct/Gatekeeper>
H323Gatekeeper	H323Gatekeeper [1..2] Address: <IPAddr>
HTTP	HTTP Mode: <On/Off>
HTTPS	HTTPS Mode: <On/Off>
IP	<p>IP [1..2] Address: &lt;IPAddr&gt;</p> <p>IP [1..2] SubnetMask: &lt;IPAddr&gt;</p> <p>IP [1..2] Gateway: &lt;IPAddr&gt;</p>
ISDNInterfaceCard	<p>ISDNInterfaceCard [1..6] ISDN SendComplete: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN SendNumber: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN ParallelDial: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN HLC: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN SpeechTimers: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI NSFTelephony Mode: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI NSFTelephony Number: &lt;0..31&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI NSFVideoTelephony Mode: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI NSFVideoTelephony Number: &lt;0..31&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI SwitchType: &lt;NI/ATT/Euro&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI TrunkGroups: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI InitialRestart: &lt;On/Off&gt;</p>

	<p>ISDNInterfaceCard [1..6] ISDN PRI Alert: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] MaxChannels: &lt;1..30&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] HighChannel: &lt;1..31&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] LowChannel: &lt;1..31&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] Search: &lt;High/Low&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] NumberRangeStart: &lt;S: 0, 24&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] NumberRangeStop: &lt;S: 0, 24&gt;</p> <p>ISDNInterfaceCard [1..6] ISDN PRI Interface [1..8] Enable: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] E1 Port [1..8] CRC4: &lt;On/Off&gt;</p> <p>ISDNInterfaceCard [1..6] T1 Port [1..8] CableLength: &lt;Range1/Range2/Range3/Range4/Range5&gt;</p>
LOS	<p>LoS Duration Exponent: &lt;10..30&gt;</p> <p>LoS Duration Offset: &lt;0..65535&gt;</p> <p>LoS Inhibit: &lt;0..65535&gt;</p> <p>LoS Initial: &lt;0..65535&gt;</p> <p>LoS Polarity: &lt;Positive/Negative&gt;</p> <p>LoS Retry: &lt;0..65535&gt;</p>
MediaBoard	<p>MediaBoard [1..12] IP Address: &lt;IPAddr&gt;</p> <p>MediaBoard [1..12] IP SubnetMask: &lt;IPAddr&gt;</p> <p>MediaBoard [1..12] IP Gateway: &lt;IPAddr&gt;</p> <p>MediaBoard [1..12] IP NetworkId: &lt;1..2&gt;</p> <p>MediaBoard [1..12] Ethernet Speed: &lt;Auto/10half/10full/100half/100full/None&gt;</p>
NetProfile	<p>NetProfile [1..6] Name: &lt;S: 0, 8&gt;</p> <p>NetProfile [1..6] CallPrefix: &lt;S: 0, 9&gt;</p> <p>NetProfile 1 Network: &lt;Auto&gt;</p> <p>NetProfile 2 Network: &lt;H320&gt;</p>

	<p>NetProfile 3 Network: &lt;H323&gt;</p> <p>NetProfile 4 Network: &lt;H320/H323/Auto&gt;</p> <p>NetProfile 5 Network: &lt;H320/H323/Auto&gt;</p> <p>NetProfile 6 Network: &lt;H320/H323/Auto&gt;</p>
OptionKey	<p>OptionKey Ports: &lt;S: 0, 20&gt;</p> <p>OptionKey SerialPorts: &lt;S: 0, 20&gt;</p> <p>OptionKey PRIPorts: &lt;S: 0, 20&gt;</p>
QoS	<p>QoS [1..2] Precedence Telephony Audio: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Precedence Telephony Signalling: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Precedence VideoTelephony Audio: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Precedence VideoTelephony Signalling: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Precedence VideoTelephony Video: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Precedence VideoTelephony Data: &lt;0/1/2/3/4/5/6/7/Auto/Off&gt;</p> <p>QoS [1..2] Diffserv Telephony Audio: &lt;0..63&gt;</p> <p>QoS [1..2] Diffserv Telephony Signalling: &lt;0..63&gt;</p> <p>QoS [1..2] Diffserv VideoTelephony Audio: &lt;0..63&gt;</p> <p>QoS [1..2] Diffserv VideoTelephony Signalling: &lt;0..63&gt;</p> <p>QoS [1..2] Diffserv VideoTelephony Video: &lt;0..63&gt;</p> <p>QoS [1..2] Diffserv VideoTelephony Data: &lt;0..63&gt;</p> <p>QoS [1..2] Mode: &lt;Precedence/Diffserv/Off&gt;</p> <p>QoS [1..2] ToS: &lt;MinDelay/MaxThrough/MaxReliable/MinCost/Off&gt;</p>
RTP	RTP MTU: <1200..1400>
SerialInterfaceCard	SerialInterfaceCard [1..6] Port [1..32] Clocking: <Dual/Single>
SerialPort	<p>SerialPort BaudRate:</p> <p>&lt;1200/2400/4800/9600/19200/38400/57600/115200&gt;</p>

	<p><b>SerialPort Parity:</b> &lt;None/Odd/Even&gt;</p> <p><b>SerialPort DataBits:</b> &lt;7/8&gt;</p> <p><b>SerialPort StopBits:</b> &lt;1/2&gt;</p>
<b>SNMP</b>	<p><b>SNMP Mode:</b> &lt;On/Off/ReadOnly/TrapsOnly&gt;</p> <p><b>SNMP CommunityName:</b> &lt;S: 0, 16&gt;</p> <p><b>SNMP SystemContact:</b> &lt;S: 0, 70&gt;</p> <p><b>SNMP SystemLocation:</b> &lt;S: 0, 70&gt;</p> <p><b>SNMP HostIPAddr [1..3]:</b> &lt;IPAddr&gt;</p>
<b>SSH</b>	<p><b>SSH Mode:</b> &lt;On/Off&gt;</p>
<b>SystemClock</b>	<p><b>SystemClock Port:</b> &lt;0..32&gt;</p>
<b>SystemUnit</b>	<p><b>SystemUnit Name:</b> &lt;S: 0, 50&gt;</p> <p><b>SystemUnit Password:</b> &lt;S: 0, 16&gt;</p>
<b>Telnet</b>	<p><b>Telnet Mode:</b> &lt;On/Off&gt;</p>

## 3.2 directory.xml – xdirectory

<b>LocalEntry</b>	<p><b>LocalEntry [1..99] Name:</b> &lt;S: 0, 16&gt;</p> <p><b>LocalEntry [1..99] Number:</b> &lt;S: 0, 30&gt;</p> <p><b>LocalEntry [1..99] SecondNumber:</b> &lt;S: 0, 30&gt;</p> <p><b>LocalEntry [1..99] SubAddress:</b> &lt;S: 0, 10&gt;</p> <p><b>LocalEntry [1..99] CallRate:</b> &lt;1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/Tlph/H0/Max/Auto&gt;</p> <p><b>LocalEntry [1..99] Restrict:</b> &lt;On/Off&gt;</p> <p><b>LocalEntry [1..99] NetProfile:</b> &lt;1..6&gt;</p> <p><b>LocalEntry [1..99] NetworkId:</b> &lt;1..32&gt;</p> <p><b>LocalEntry [1..99] NetworkModule:</b> &lt;0..6&gt;</p>
<b>GroupEntry</b>	<p><b>GroupEntry [1..16] Name:</b> &lt;S: 0, 16&gt;</p> <p><b>GroupEntry [1..16] LocalEntryId [1..32]:</b> &lt;0..99&gt;</p>

## 4 API - Commands

---

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

## 4.1 command.xml – xcommand

<p><b>Boot</b></p>	<p>Command used to reboot the system.</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 boot</pre> <pre>*r Result (status=OK): *r/end</pre> <p>OK</p>
<p><b>CallMute</b></p>	<p>Command used to mute incoming audio from a specific call.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Call(r):</b> &lt;1..128&gt; Reference to the call to be muted or unmuted.</li> <li>• <b>Mode(r):</b> &lt;On/Off&gt; Denotes whether the call is to be muted or unmuted.</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 callmute call:2 mode:on</pre> <pre>*r Result (status=OK): *r/end</pre> <p>OK</p>
<p><b>CallMuteVideo</b></p>	<p>Command used to mute incoming video from a specific call.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Call(r):</b> &lt;1..128&gt; Reference to the call to be muted or unmuted.</li> <li>• <b>Mode(r):</b> &lt;On/Off&gt; Denotes whether video is to be muted or unmuted.</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> xcommand callmutevideo call:2 mode:on</p> <p>*r Result (status=OK): *r/end</p> <p>OK</p>
<p><b>ConferenceDisconnect</b></p>	<p>Command used to disconnect all calls in a conference.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</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> xcommand conferencedisconnect conference:1</p> <p>*r Result (status=OK): *r/end</p> <p>OK</p>
<p><b>ConferenceModify</b></p>	<p>Command used to modify conference settings.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</li> <li>• <b>PictureMode:</b> &lt;Auto/4Split/9Split/16Split/5+1Split/7+1Split/VS/CPAuto&gt;</li> <li>• <b>VideoFormat:</b> &lt;Auto/Motion/Sharpness&gt;</li> <li>• <b>CustomFormats:</b> &lt;On/Off&gt;</li> <li>• <b>AGC:</b> &lt;On/Off&gt;</li> <li>• <b>AllowIncomingCalls:</b> &lt;On/Off&gt;</li> <li>• <b>Duration:</b> &lt;0..65534&gt;</li> <li>• <b>MaxAudioSites:</b> &lt;0..64&gt;</li> <li>• <b>MaxVideoSites:</b> &lt;0..64&gt;</li> <li>• <b>EntryExitTones:</b> &lt;On/Off&gt;</li> <li>• <b>LegacyLevel:</b> &lt;0..15&gt;</li> <li>• <b>TelephoneFilter:</b> &lt;On/Off&gt;</li> <li>• <b>FloorToFull:</b> &lt;On/Off&gt;</li> <li>• <b>WebCallListTimeout:</b> &lt;On/Off&gt;</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> xcommand conferencemodify conference:1 agc:on floortofull:on</p> <p>*r Result (status=OK): *r/end</p>

	OK
<p><b>ConferenceRedefine</b></p>	<p>Command used to modify conference settings for a started conference with no active calls.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</li> <li>• <b>Name:</b> &lt;S: 0, 30&gt;</li> <li>• <b>CallRate:</b> &lt;1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/Tlph/H0&gt;</li> <li>• <b>Restrict:</b> &lt;On/Off&gt;</li> <li>• <b>Password:</b> &lt;S: 0, 8&gt;</li> <li>• <b>Encryption:</b> &lt;On/Off&gt;</li> <li>• <b>EncryptionType:</b> &lt;DES/AES-128/Auto&gt;</li> <li>• <b>WelcomeMessage:</b> &lt;On/Off&gt;</li> <li>• <b>DuoVideo:</b> &lt;On/Off&gt;</li> <li>• <b>AudioG728:</b> &lt;On/Off&gt;</li> <li>• <b>CascadingPreference:</b> &lt;Auto/Master/Slave&gt;</li> <li>• <b>BillingCode:</b> &lt;S: 0, 16&gt;</li> <li>• <b>CPAutoSwitch:</b> &lt;0..60&gt;</li> <li>• <b>NetworkId:</b> &lt;1..32&gt;</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 conferenceredefine conference:1 duovideo:on restrict:on</pre> <pre>*r Result (status=OK): *r/end</pre> <p>OK</p>
<p><b>ConferenceStart</b></p>	<p>Command used to start a new conference.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</li> <li>• <b>Name:</b> &lt;S: 0, 30&gt;</li> <li>• <b>CallRate:</b> &lt;1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/Tlph/H0&gt;</li> <li>• <b>Restrict:</b> &lt;On/Off&gt;</li> <li>• <b>Password:</b> &lt;S: 0, 8&gt;</li> <li>• <b>Encryption:</b> &lt;On/Off&gt;</li> <li>• <b>EncryptionType:</b> &lt;DES/AES-128/Auto&gt;</li> <li>• <b>WelcomeMessage:</b> &lt;On/Off&gt;</li> <li>• <b>DuoVideo:</b> &lt;On/Off&gt;</li> <li>• <b>AudioG728:</b> &lt;On/Off&gt;</li> <li>• <b>CascadingPreference:</b> &lt;Auto/Master/Slave&gt;</li> <li>• <b>BillingCode:</b> &lt;S: 0, 16&gt;</li> <li>• <b>CPAutoSwitch:</b> &lt;0..60&gt;</li> <li>• <b>NetworkId:</b> &lt;1..32&gt;</li> </ul> <p><b>OK Result parameters:</b></p>

	<p>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 conferencestart conference:1</pre> <pre>*r Result (status=OK):</pre> <pre>*r/end</pre> <hr/> <p>OK</p>
<p><b>ConferenceStop</b></p>	<p>Command used to stop a conference. All active calls must be disconnected prior to stopping the conference.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</li> </ul> <p><b>OK Result parameters:</b></p> <p>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 conferencestop conference:1</pre> <pre>*r Result (status=OK):</pre> <pre>*r/end</pre> <hr/> <p>OK</p>
<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></p> <p>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 defaultvaluesset level:2</pre> <pre>*r Result (status=OK):</pre> <pre>*r/end</pre> <hr/> <p>OK</p>
<p><b>Dial</b></p>	<p>Command used to initiate an outgoing call.</p> <p><b>Parameters:</b></p>

- **Conference(r): <1..12>**
- **Number: <S: 0, 30>** Number to dial.
- **SecondNumber: <S: 0, 30>** 2Xh221 second number
- **SubAddress: <S: 0, 10>** Sub address
- **CallRate: <Tlph/1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto>** Specifies the callrate to use
- **Restrict: <On/Off>**
- **NetProfile: <1..6>**
- **NetworkId: <1..32>**
- **NetworkModule: <1..12>**

**OK Result parameters:**

- **CallRef: <1..96>** Reference to the call. To be used as reference when monitoring the call.
- **LogTag: <1...>** Unique reference to call. Identifies the call in the call log.

**ERROR Result parameters:**

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

**Example:**

```
xcommand dial number:666 callrate:256 netprofile:3
```

```
*r Result (status=OK):
    CallRef: 26
    LogTag: 312
*r/end
```

OK

**DialGroupEntry**

Command used to dial an entry from the Group Directory. Dialling from the Group Directory makes it possible to set up a MultiSite conference in one operation.

**Parameters:**

- **Conference(r): <1..12>**
- **GroupEntryId(r): <1..16>** Reference to the directory entry to be dialed.

**OK Result parameters:**

The system will return the following elements for each call initiated.

- **CallRef: <1..96>** Reference to the call. To be used as reference when monitoring the call.
- **LogTag: <1...>** Unique reference to call. Identifies the call in the call log.

**ERROR Result parameters:**

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

**Example:**

```
xcommand dialgroupentry conference:1 groupentryid:19
```

```
*r Result (status=OK):
    CallRef: 2
    LogTag: 313
```

	<pre> CallRef: 1 LogTag: 312 CallRef: *r/end  OK </pre>
<p><b>DialLocalEntry</b></p>	<p>Command used to dial a number from the locally stored directory.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r): &lt;1..12&gt;</b></li> <li>• <b>LocalEntryId(r): &lt;1..99&gt;</b> Reference to the directory entry to be dialed.</li> </ul> <p><b>OK Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>CallRef: &lt;1..96&gt;</b> Reference to the call. To be used as reference when monitoring the call.</li> <li>• <b>LogTag: &lt;1...&gt;</b> Unique reference to call. Identifies the call in the call log.</li> </ul> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause: &lt;1...&gt;</b> Cause code specifying why the call 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 diallocalentry localentryid:15  *r Result (status=OK):   CallRef: 1   LogTag: 312 *r/end  OK </pre>
<p><b>DisconnectCall</b></p>	<p>Command used to disconnect a call.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Call(r): &lt;1..128&gt;</b> Reference to the call to be disconnected.</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: &lt;1...&gt;</b> 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 disconnectcall call:9  *r Result (status=OK): *r/end  OK </pre>
<p><b>FeedbackDeregister</b></p>	<p>Command used to deregister XML feedback over HTTP(S).</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>ID: &lt;1..3&gt;</b> ID for the registration to deregister.</li> </ul> <p><b>OK Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>ID: &lt;1..3&gt;</b></li> </ul>

	<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 feedbackderegister id:1</pre> <pre>*r Result (status=OK):   ID: 2 *r/end</pre> <p>OK</p>
<p><b>FeedbackRegister</b></p>	<p>Command used to instruct the system to return XML feedback over HTTP(S) to specific URLs. What parts of the Status and Configuration XML documents to monitor are specified by XPath expressions. The system supports issuing feedback to 3 different URLs. The system allows a total of 20 XPath expressions to be registered, with a maximum of 15 for a single URL.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>ID:</b> &lt;1..3&gt; ID for the registration. If this parameter is omitted the system uses the first vacant ID.</li> <li>• <b>URL(r):</b> &lt;S: 0, 256&gt; The URL to post feedback to.</li> <li>• <b>Expression.1..15:</b> &lt;S: 0, 256&gt; XPath expression</li> </ul> <p><b>OK Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>ID:</b> &lt;1..3&gt;</li> </ul> <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 feedbackregister url:http://10.47.14.185:8000                              expression.1:status/call                              expression.2:status/conference</pre> <pre>*r Result (status=OK):   ID: 2 *r/end</pre> <p>OK</p>
<p><b>FloorToSite</b></p>	<p>Command used to assign floor to a specific site in a conference.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r):</b> &lt;1..12&gt;</li> <li>• <b>MCUID(r):</b> &lt;1..191&gt; MCUID to the MultiSite the site is connected to.</li> <li>• <b>TerminalID(r):</b> &lt;1..191&gt; The site's terminal id, referenced to the MultiSite it is connected to.</li> </ul> <p><b>OK Result parameters:</b></p> <p>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>  xcommand floortosite conference:4 mcuid:85 terminalid:2</p> <pre>*r Result (status=OK): *r/end</pre> <p>OK</p>
<p><b>FloorToSiteEnd</b></p>	<p>Command used to end the assignment of floor to a specific site in a conference supporting. Requires that the command <i>FloorToSite</i> has been issued in advance.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Conference(r): &lt;1..12&gt;</b></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: &lt;1...&gt;</b> 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>  xcommand floortositeend</p> <pre>*r Result (status=OK): *r/end</pre> <p>OK</p>
<p><b>GroupEntryAdd</b></p>	<p>Command used to add a new Group entry to the locally stored Group Directory (or MultiSite Directory). The entry is stored in the first vacant position in the Group Directory.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Name: &lt;S: 0, 16&gt;</b> The entry's name.</li> <li>• <b>LocalEntryId.1..32: &lt;1..99&gt;</b>References to local entry ids to be included in this Group entry.</li> </ul> <p><b>OK Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>GroupEntryId: &lt;1..16&gt;</b> Reference to the Group Directory position the entry is stored.</li> </ul> <p><b>ERROR Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Cause: &lt;1...&gt;</b> 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>  xcommand groupentryadd name:"The Team"  localentryid.1:17  localentryid.2:29  localentryid.3:56</p> <pre>*r Result (status=OK):     GroupEntryId: 15 *r/end</pre> <p>OK</p>

<p><b>GroupEntryDelete</b></p>	<p>Command used to delete an entry in the locally stored Group Directory.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>GroupEntryId(r):</b> &lt;1..16&gt; Reference to the entry to delete.</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 groupentrydelete groupentryid:15  *r Result (status=OK): *r/end  OK</pre>
<p><b>LocalEntryAdd</b></p>	<p>Command used to add a new entry to the locally stored Directory. The entry is stored in the first vacant position in the Directory.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>Name:</b> &lt;S: 0, 16&gt; The entry's name.</li> <li>• <b>Number:</b> &lt;S: 0, 30&gt; The entry's number.</li> <li>• <b>SecondNumber:</b> &lt;S: 0, 30&gt; The entry's second number (2XH221 number).</li> <li>• <b>SubAddress:</b> &lt;S: 0, 10&gt; The entry's sub address.</li> <li>• <b>CallRate:</b> &lt;Tlph/1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto&gt; The callrate to use when calling this entry.</li> <li>• <b>Restrict:</b> &lt;On/Off&gt; Whether to use restrict or not when calling this entry.</li> <li>• <b>NetProfile:</b> &lt;1..6&gt; The Net Profile to use when calling this entry.</li> <li>• <b>NetworkId:</b> &lt;1..32&gt;</li> <li>• <b>NetworkModule:</b> &lt;1..6&gt;</li> </ul> <p><b>OK Result parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>LocalEntryId:</b> &lt;1..250&gt; Reference to the Directory position the entry is stored.</li> </ul> <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 localentryadd name:"John Galt" number:666  *r Result (status=OK):     LocalEntryId: 17 *r/end  OK</pre>
<p><b>LocalEntryDelete</b></p>	<p>Command used to delete an entry in the locally stored Directory.</p> <p><b>Parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>LocalEntryId(r):</b> &lt;1..99&gt; Reference to the entry to delete.</li> </ul>

**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 localentrydelete localentryid:66
```

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

```
*r/end
```

```
OK
```

## 5 API - Status

---

This section gives an overview of the Status Information available in the Status XML documents (status.xml / history.xml) and the Status root commands (xstatus / xhistory) of the XACLI interface.

All examples are presented using the standard XACLI format.

## 5.1 status.xml – xstatus

### Call [1..96]

#### Top level attributes:

- **status:**  
CallIDLE/Dialing/Alerting/Proceeding/EstabOut/EstabIn/AwaitInCnf/Connected/Disconnecting/Disconnected/Await2ndnr/ ClearOut/ClearIn/Syncing/Capex/Synced/Unframed
- **type:** Tlph/Vtlph
- **protocol:** H320/H323
- **direction:** Incoming/Outgoing
- **logTag:** 1... Unique number identifying the call. This tag can be used to track the call in the call log (*history.xml / xhistory*).
- **conferenceRef:** 1..9

#### Summary:

- Returns all currently available information for a call.

#### Examples:

```
*s Call 1 (status=Synced, type=Vtlph,
protocol=H323, direction=Outgoing, logTag=1,
conferenceRef=1):
```

```
  CallRate: 384
  RemoteNumber: "10.47.15.127"
  Mute: Off
  Microphone: Off
  Duration: 29
  Channels 1 (type=Incoming):
    Rate: 384
    Restrict: Off
    Encryption (status=Off): /
    Audio (status=Active):
      Protocol: G722
      Rate: 64
      RemoteIPAddress: ""
      LocalIPAddress: "10.47.9.106:2326"
      Encryption (status=Off): /
      RSVP: Off
      RSVPRate: 0
      DynamicRate: 64
      TotalPackets: 621
      PacketLoss: 0
      Jitter: 0
    Video 1 (status=Active):
      Protocol: H263
      Resolution: CIF
      Rate: 320
      RemoteIPAddress: ""
      LocalIPAddress: "10.47.9.106:2328"
      Encryption (status=Off): /
      RSVP: Off
      RSVPRate: 0
      DynamicRate: 37
      TotalPackets: 689
      PacketLoss: 0
      Jitter: 0
    Video 2 (status=Inactive): /
```

```

Data (status=Inactive): /
Channels 2 (type=Outgoing):
  Rate: 384
  Restrict: Off
  Encryption (status=Off): /
Audio (status=Active):
  Protocol: G722
  Rate: 64
  RemoteIPAddress: "10.47.15.127:2326"
  LocalIPAddress: "10.47.9.106:2334"
  Encryption (status=Off): /
  RSVP: Off
  RSVPRate: 0
  DynamicRate: 64
  TotalPackets: 1245
  PacketLoss: 0
  Jitter: 0
Video 1 (status=Active):
  Protocol: H264
  Resolution: CIF
  Rate: 320
  RemoteIPAddress: "10.47.15.127:2328"
  LocalIPAddress: "10.47.9.106:2336"
  Encryption (status=Off): /
  RSVP: Off
  RSVPRate: 0
  DynamicRate: 3
  TotalPackets: 506
  PacketLoss: 0
  Jitter: 36
Video 2 (status=Inactive): /
Data (status=Inactive): /
*s/end

*s Call 2 (status=Disconnected, type=NA,
protocol=NA, direction=NA, logTag=NA,
conferenceRef=NA):
  Cause: 255
*s/end

```

### Conference [1..9]

#### Top level attributes:

- **status: NotStarted/Started/Active**

#### Summary:

- Includes references to the calls being connected to the conference
- DuoVideo status
- Includes information about the pictures generated by the MultiSite
- Cascading status
- MCU Site list
- On Air information

#### Examples:

```

*s Conference 1 (status=NotStarted): /
*s/end

*s Conference 1 (status=Started):
  MCUID: 1
  Properties:

```

```

Name: ""
CallRate: 384
Restrict: Off
Password: ""
Encryption: Off
EncryptionType: Auto
WelcomeMessage: On
DuoVideo: On
AudioG728: On
CascadingPreference: Auto
BillingCode: ""
CPAutoSwitch: 0
PictureMode: Auto
VideoFormat: Auto
CustomFormats: On
AGC: On
AllowIncomingCalls: On
Duration: 0
MaxAudioSites: 8
MaxVideoSites: 8
EntryExitTones: On
LegacyLevel: 0
TelephoneFilter: On
FloorToFull: On
WebCallListTimeout: On
NetworkId: 1
*s/end

*s Conference 1 (status=Active):
Calls:
  CallRef 1: 1
  CallRef 2: 2
  CallRef 3: 3
DuoVideo (status=Off): /
Floor: None
Current:
  CallRef: 3
Previous:
  CallRef: 2
OutgoingPicture 1 (name=Current):
  Layout (type=5+1Split):
    Window 1:
      Picture: RemoteMain
      CallRef: 3
    Window 2:
      Picture: RemoteMain
      CallRef: 2
    Window 3:
      Picture: RemoteMain
      CallRef: 1
    Window 4:
      Picture: NA
      CallRef: None
    Window 5:
      Picture: NA
      CallRef: None
    Window 6:
      Picture: NA
      CallRef: None
OutgoingPicture 2 (name=Previous):
  Layout (type=5+1Split):

```

```
Window 1:  
  Picture: RemoteMain  
  CallRef: 2  
Window 2:  
  Picture: RemoteMain  
  CallRef: 3  
Window 3:  
  Picture: RemoteMain  
  CallRef: 1  
Window 4:  
  Picture: NA  
  CallRef: None  
Window 5:  
  Picture: NA  
  CallRef: None  
Window 6:  
  Picture: NA  
  CallRef: None  
OutgoingPicture 3 (name=Duo):  
  Layout (type=NA): /  
MCUID: 1  
CascadingMode: StandAlone  
MCUSiteList:  
  Site 1:  
    MCUID: 1  
    TerminalID: 2  
    Name: "System1"  
    CallRef: 1  
  Site 2:  
    MCUID: 1  
    TerminalID: 3  
    Name: "System2"  
    CallRef: 2  
  Site 3:  
    MCUID: 1  
    TerminalID: 4  
    Name: "System3"  
    CallRef: 3  
Properties:  
  Name: ""  
  CallRate: 384  
  Restrict: Off  
  Password: ""  
  Encryption: Off  
  EncryptionType: Auto  
  WelcomeMessage: On  
  DuoVideo: On  
  AudioG728: On  
  CascadingPreference: Auto  
  BillingCode: ""  
  CPAutoSwitch: 0  
  PictureMode: Auto  
  VideoFormat: Auto  
  CustomFormats: On  
  AGC: On  
  AllowIncomingCalls: On  
  Duration: 0  
  MaxAudioSites: 8  
  MaxVideoSites: 8  
  EntryExitTones: On  
  LegacyLevel: 0
```



	<pre>*s H323Gatekeeper 1 (status=Registered):   Address: "10.47.9.1"   Port: 1719 *s/end  *s H323Gatekeeper 1 (status=Rejected):   Address: "10.47.9.0"   Port: 0 *s/end</pre>
<p><b>IP [1..2]</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 1:   Address: "10.47.8.222"   SubnetMask: "255.255.248.0"   Gateway: "10.47.8.1" *s/end</pre>
<p><b>ISDNInterface [1..6]</b></p>	<p><b>Example</b></p> <pre>*s ISDNInterfaceCard 1 (status=On):   PRI 1 (ready=True):     BChannelsTotal: 30     BChannelsFree: 30     H0ChannelsFree: 5     Channels 1 (type=BChannel, status=Idle): /     Channels 2 (type=BChannel, status=Idle): /     Channels 3 (type=BChannel, status=Idle): /     Channels 4 (type=BChannel, status=Idle): /     Channels 5 (type=BChannel, status=Idle): /     Channels 6 (type=BChannel, status=Idle): /     Channels 7 (type=BChannel, status=Idle): /     Channels 8 (type=BChannel, status=Idle): /     Channels 9 (type=BChannel, status=Idle): /     Channels 10 (type=BChannel, status=Idle): /     Channels 11 (type=BChannel, status=Idle): /     Channels 12 (type=BChannel, status=Idle): /     Channels 13 (type=BChannel, status=Idle): /     Channels 14 (type=BChannel, status=Idle): /     Channels 15 (type=BChannel, status=Idle): /     Channels 16 (type=DChannel, status=NA): /     Channels 17 (type=BChannel, status=Idle): /     Channels 18 (type=BChannel, status=Idle): /     Channels 19 (type=BChannel, status=Idle): /     Channels 20 (type=BChannel, status=Idle): /     Channels 21 (type=BChannel, status=Idle): /     Channels 22 (type=BChannel, status=Idle): /     Channels 23 (type=BChannel, status=Idle): /     Channels 24 (type=BChannel, status=Idle): /     Channels 25 (type=BChannel, status=Idle): /     Channels 26 (type=BChannel, status=Idle): /     Channels 27 (type=BChannel, status=Idle): /     Channels 28 (type=BChannel, status=Idle): /</pre>

	<pre>Channels 29 (type=BChannel, status=Idle): / Channels 30 (type=BChannel, status=Idle): / Channels 31 (type=BChannel, status=Idle): / *s/end</pre>
<p><b>MediaBoard [1..3]</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 and Ethernet information for the Media Boards.</li> </ul> <p><b>Example</b></p> <pre>*s MediaBoard 1:   IP:     Address: "10.47.9.106"     SubnetMask: "255.255.248.0"     Gateway: "10.47.8.1"   Ethernet:     MacAddress: "00:50:60:00:ED:1F"     Speed: 100full *s/end</pre>
<p><b>SerialInterfaceCard [1..6]:</b></p>	<p><b>Example</b></p> <pre>*s SerialInterfaceCard 1 (status=On):   Port 1 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 2 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 3 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 4 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 5 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 6 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 7 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 8 (status=On):     ClockRate: 0     CarrierDetect: Off     DTR: Off   Port 9 (status=Off): /   Port 10 (status=Off): /</pre>

```

Port 11 (status=Off): /
Port 12 (status=Off): /
Port 13 (status=Off): /
Port 14 (status=Off): /
Port 15 (status=Off): /
Port 16 (status=Off): /
Port 17 (status=Off): /
Port 18 (status=Off): /
Port 19 (status=Off): /
Port 20 (status=Off): /
Port 21 (status=Off): /
Port 22 (status=Off): /
Port 23 (status=Off): /
Port 24 (status=Off): /
Port 25 (status=Off): /
Port 26 (status=Off): /
Port 27 (status=Off): /
Port 28 (status=Off): /
Port 29 (status=Off): /
Port 30 (status=Off): /
Port 31 (status=Off): /
Port 32 (status=Off): /
*s/end

```

### SystemUnit

#### Top level attributes:

None

#### Summary:

- Returns information about the System Unit

#### Example

```

*s SystemUnit:
  ProductType: "TANDBERG MPS-MCU"
  Uptime: 9851
  Software:
    Version: "J1.0"
    Name: "s41001"
    ReleaseDate: "2004-08-19"
  Configuration:
    Telephony: 48
    VideoTelephony: 48
    AdvancedVideoOption: True
  Hardware:
    SerialNumber: "36a00103"
    MainBoard: "113637 MPC 820 System Control"
    BootSoftware: "PPCBUG"
*s/end

```

## 5.2 history.xml – xhistory

<p><b>Call [1..96]</b></p>	<p><b>Top level attributes:</b></p> <ul style="list-style-type: none"> <li>• <b>type:</b> Tlph/Vtlph</li> <li>• <b>protocol:</b> H320/H323</li> <li>• <b>direction:</b> Incoming/Outgoing</li> </ul> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>• Returns information about disconnected calls</li> </ul> <p><b>Examples:</b></p> <pre>*1 Call 1 (type=Vtlph, protocol=H323, direction=Outgoing):   LogTag: 4   ConferenceLogTag: 2   ConferenceRef: 1   RemoteNumber: "10.47.12.242"   CallRate: 384   DisconnectCauseValue: 16   Duration: 0   UptimeAtEndOfCall: 7758   BillingCode: "" *1/end</pre>
	<p><b>Conference [1..20]</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 ended conferences</li> </ul> <p><b>Examples:</b></p> <pre>*1 Conference 1:   LogTag: 1   Name: ""   CallRate: 384   Restrict: Off   Password: ""   Encryption: Off   Duration: 0 *1/end</pre>