TANDBERG MXP API

User Manual

TANDBERG
D13638 Rev 03

The TANDBERG API

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

Table of Contents

1.1 Introduction to XML 1
1.2 Introduction to XML Path Language (XPath) 4
1.3 The TANDBERG XML Engine 5
1.3.1 Structuring of Information 5
1.3.2 Addressing using XPath 5
1.3.3 Feedback 5
1.4 The XML Documents 7
1.4.1 Documents 7
1.4.2 Status Documents (r) 7
1.4.3 Configuration documents (rw) 8
1.4.4 Command documents (w) 9
1.5 Introduction to TANDBERG XML API Service (TXAS) 11
1.5.1 Bare-bone HTTP(S) access 11
1.5.2 SOAP 12
1.6 Exercises 13
2 The XML-based Advanced Command Line Interface 15
2.1 XACLI 16
2.1.1 Accessing XACLI 16
2.1.2 Root commands 16
2.1.3 Addressing 16
2.1.4 Exposure options 16
2.1.5 Misc 18
2.2 The Status-type root commands – xstatus / xhistory 19
2.2.1 Format 19
2.3 The Configuration-type root commands - xconfiguration/xdirectory 21
2.3.1 Configuration help 21
2.3.2 Configuration read 22
2.3.3 Configuration set (write) 22
2.4 The Command-type root commands - xcommand 23
2.4.1 Command help 23
2.4.2 Issuing a command 24
2.5 XML Output - xgetxml 26
2.6 Special Commands 27
2.6.1 xfeedback 27
2.6.2 xpreferences 30
3 API - Configurations 33
3.1 configuration.xml – xconfiguration 34
3.2 directory.xml – xdirectory 49
4 API - Commands 50
4.1 commands.xml - xcommands 51
5 API - Status 74
5.1 status.xml – xstatus 75
5.2 history.xml – xhistory 86

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 beeing 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="Vtlph" 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" />
```

```
      <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="Vtlph" 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 documents (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 responsse 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 a 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 mehtod of POSTing form data (Each argument starts with a name '=' and a value, and every parameter seperated with '&' (and opt NL).)

1.5.1.1 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.

1.5.1.2 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>).

1.5.1.3 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 InternetExplorer. 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>

2.1.4.1 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=Vtlph, 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 availible in the corresponding XML documents.

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 -
 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=Vtlph, 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 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          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.

2.4.2.1 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
```

2.4.2.2 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
```

2.4.2.3 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
```

2.4.2.4 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=Vtlph, 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=Vtlph, 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 availble 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

AlertSpeaker	**AlertSpeaker Mode: <On/Off>** Turns the internal alert speaker on or off.
AlertTone	**AlertTone Volume: <0..15>** Sets the volume of the alert tone. **AlertTone VideoTelephony: <1..10>** Sets the alert tone to use for incoming telephone calls. **AlertTone Telephony: <1..10>** Sets the alert tone to use for incoming videotelephone calls.
Audio	**Audio Microphones Mode: <On/Off>** Turns all microphones on or off. This is the configuration that is tied to the “Mic off” key on the TANDBERG MXP remote control. **Audio MicrophoneMixer Mode: <Fixed/Auto>** When set to Auto the adjustment of each microphone signal is done automatically to obtain the best possible audio and minimize the background noise. When set to Fixed the system will maintain a constant weighting of all microphones. **Audio AutoMute: <On/Off>** When automute is set to on, the mic will be turned off automatically at boot and at end of calls. **Audio AudioModule: <NAMII-6000/NAMII-7000/NAMII-8000/DNAM/None>** Select Audio Module according to the type of Audio Module installed if this is not automatically detected. The Audio Module setting will only take effect if the audio module of the system is unidentified. **Audio Volume: <0..15>** Sets the volume level output. **Audio AGC Microphones: <On/Off>** Turns the AGC connected to the microphone mixer on or off. On a TANDBERG 6000 MXP, Line input 1 (Audio in 4) is also connected to this AGC. **Audio AGC AUX: <On/Off>** Turns the AGC connected to the AUX input on or off. NOTE! The AUX input is also sometimes referred to as Line Input 2 or Audio in 5. **Audio AGC VCR: <On/Off>** Turns the AGC connected to the VCR input on or off. NOTE! The VCR input is also sometimes referred to as Line Input 3 or Audio in 6.

Audio AGC Received: <On/Off>
Turns AGC for received audio on or off.

Audio EchoControl [1..4]: <On/Off/NoiseReduction>
Turns echo control on or off for the various microphone inputs. Line input 1 (Audio in 4) is connected to EchoControl number 4.

Audio Stereo: <On/Off>
Turns stereo on or off.

Audio StereoSpeakers: <On/Off>

Audio VCRDucking: <On/Off>
If VCR Ducking is activated, the VCR audio level will be attenuated if someone talks into the microphone or at the far end.

Audio KeyTones: <On/Off>
If set to On there will be a sound indicator when pressing keys on the remote control.

Audio Inputs Microphone [1..3] Level: <1..16>
Defines the input levels for the microphones.

Audio Inputs Microphone 1 Mode: <On/Off>
Used to enable/disable Microphone input 1.

Audio Inputs Microphone 2 Mode: <On/Off>
Used to enable/disable Microphone input 2.

Audio Inputs Microphone 3 Mode: <On/Off/Line>
Used to enable/disable Microphone input 3.
IMPORTANT! If set to *Line,* Microphone input 3 will become a balanced line level input. The gain will be reduced accordingly.

Audio Inputs Line [1..3] Level: <1..16>
Defines the input levels for the line inputs.

Audio Inputs Line 1 Mode: <On/Off>
Used to enable/disable Line input 1.

Audio Inputs Line 2 Mode: <On/Off>
Used to enable/disable Line input 2.

Audio Inputs Line 3 Mode: <On/Off/Auto>
Used to enable/disable Line input 3. If set to *Auto* the input will be turned off until VCR is selected as either main video source or duo video source.

Audio Outputs Line [1..3] Mode: <On/Off>
Turns *On* or *Off* the supported audio outputs.

Audio Outputs Line [1..3] Level: <1..16>
Defines the output levels for the audio outputs.

Audio Outputs Line 1 Type: <Analog/SPDIF/Auto>
If Type is set to Auto, the system will select analog or digital (SPDIF) mode dependent on the detected Audio Module. If a TANDBERG Digital NAM is detected, SPDIF mode will be selected; otherwise

	analog mode will be selected. Setting Type to either Analog or SPDIF will override the auto-detect mode.
AutoAnswer	**AutoAnswer Mode: <On/Off/Mute>** Turns autoanswer on or off. If set to Mute, incoming calls will be answered automatically, but the microphones will be turned off. **AutoAnswer Delay: <1..50>** Defines how long an incoming call has to wait before it is answered automatically by the system (autoanswer must be enabled).
AutoLayout	**AutoLayout Mode: <On/Off>** If set to *On* the system will change the local picture layout automatically.
AutoPIP	**AutoPIP Mode: <On/Off/Auto>** If set to *On* the system will bring up the PIP automatically when certain actions are performed.
Bonding	**Bonding Timer: <Normal/Relaxed>** Relaxed bonding timing should be used with applications where the B-channels use some additional time before they become transparent, like external encryption devices etc.
CallManager	**CallManager Address: <IPAddr>** Sets the ip address of the callmanager to use (if any).
Camera	**Camera [1..5] Brightness Mode: <Manual/Auto>** Sets whether to control camera brightness manually or have it automatically set by the system. **Camera [1..5] Brightness Level: <0..16>** Defines the brightness level to use if brightness mode is set to manual.
CameraSleep	**CameraSleep Mode: <On/Off>** If turned on, the main camera will go into sleep position (maximum right panning) when screensaver is activayed. In addition, video source 1 (maincam) will be selected (if not allready active).
CameraTracking	**CameraTracking Speed: <Slow/Normal/Fast>** Sets the camera tracking speed. Camera tracking is enabled/disabled by the commands *CameraTrackingStart* and *CameraTrackingStop.*
Conference	**Conference DefaultCall CallRate: <Tlph/1xh221/2xh221/64/128/192/ 256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto>** Defines the default callrate to use when placing calls from the system. **Conference DefaultCall Restrict: <On/Off>** Defines whether to use restrict or not as default when placing calls from the system.

Conference DefaultCall NetProfile: <1..7>
Defines the default Net Profile to use when when placing calls from the system.

Conference H323Alias E164: <E164: 0, 49>
Defines the H323 E164 alias for the system.

Conference H323Alias ID: <S: 0, 30>
Sets the H323 alias id to be registered at the Gatekeeper. When registered, the unit can be reached by using the H323 alias id as an alphanumeric dial string.
NOTE! If the H323 alias id is not defined, the system will use the system unit name as H323 alias id.

Conference SIP URI: <S: 0, 60>

Conference H263: <On/Off>
Enables/disables H.263 for both transmit and receive.

Conference H264: <On/Off>
Enables/disables H.264 for both transmit and receive.

Conference HD: <On/Off>
Enables/disables HD for both transmit and receive.

Conference G722: <On/Off>
Enables/disables G.722 for both transmit and receive.

Conference G722.1: <On/Off>
Enables/disables G.722.1 for both transmit and receive.

Conference G728: <On/Off>
Enables/disables G.728 for both transmit and receive.

Conference AAC-LD: <On/Off>
Enables/disables AAC-LD for both transmit and receive.

Conference H331: <On/Off>
Turns broadcast mode on or off. When set to on it's possible to make an outgoing call without any capability exchange.

Conference H239: <On/Off>
Enables/disables H.239 for both transmit and receive.

Conference AAC-LD-128-Threshold: <384/512/768/1152/1472/1920/2560/3072/4096>
The system will not transmit AAC-LD-128 unless the callrate is equal to or above the specified threshold.

Conference AAC-LD-128-Mono: <On/Off>
If set to *Off* and the system is transmitting mono, the system will not use AAC-LD-128 even though the callrate is above the AAC-LD-128 Threshold.

Conference NaturalVideo: <Off/Auto/384/512/768/1152/1472/1920>
Choosing Natural Video will enable 60 (50 on PAL) fields per second true interlaced picture for high motion video. The use of Natural Video requires the H.263+ and H.263++ video protocols. Natural video will be disabled in H.323 MultiSite calls and in H.320 Continous Presence

MultiSite calls.
When set to *Off,* Natural Video will be disabled for both transmit and receive.

When set to *Auto,* the system will enable transmission of Natural Video from 768 kbps and above. Reception of Natural Video is in this case always enabled.
Natural Video can also be configured to be enabled for callrates above a specified threshold value.

Conference PictureMode: <4Split/5+1Split/VS/Auto>
Sets the picture layout to be used in a MultiSite conference. If set to *Auto* the system will change the picture layout depending on the number of participants in the conference.

Conference VideoQualityCP: <Motion/Sharpness/Auto>
Sets the video quality to be used in continous presence mode (*4Split / 5+1Split*).

Conference FloorToFull: <On/Off>
If *picturemode* is set to *5+1Split*, and one of the participants in a MultiSite conference is granted floor this setting determines whether to display the participant having floor in full screen or just in the main window of the *5+1Split*.

Conference MaxCallLength: <0..999>
Sets the maximum allowed call length in minutes (0 = Off).

Conference AllowIncomingTlphCall: <On/Off>
If set to *Off*, the system will not accept incoming telephone calls.

Conference AllowIncomingMSCall: <On/Off>
If set to *Off*, the system will not accept incoming calls to an already active conference.

Conference Downspeed: <On/Off>
Determines whether to allow downspeeding or not.

Conference FallbackToTelephony: <On/Off>
If set to *On*, the system will try to establish a telephone call if a video telephone call could not be made.

Conference Encryption Mode: <On/Off/Auto>
If set to on, the system will not allow unencrypted calls. If set to auto, the system will use encryption whenever possible. If set to off, encryption will not be used.

Conference Encryption Type: <Auto/DES/AES-128>
Defines the encryption algorithm to use.

Conference AIM: <On/Off>
Enables/disables *Auto Indicate Mute*. If disabled, the system will not signal to the other side that the microphone is turned off.

Conference IPLR Transmit: <On/Off>
Enables/disables *Intelligent Packetloss Recovery*.

Conference WebSnapshots: <On/Off>
If set to on, snapshots of the conference can be viewed from the

	systems web interface. **Conference BillingCode: <On/Off>**
CorporateDirectory	**CorporateDirectory Mode: <On/Off>** **CorporateDirectory Address: <IPAddr>** **CorporateDirectory Path: <S: 0, 255>**
DoNotDisturb	**DoNotDisturb Mode: <On/Off>** When do not disturb is set to *On*, the codec will not alert the user to incoming calls. The calling side will receive a busy signal when trying to call the codec. Do not disturb will be turned off if the codec receives any IR signal from the handheld remote control.
DualMonitor	**DualMonitor Mode: <On/Off>** Sets the codec's monitor mode. It allows the user to set up the codec so it can utilize two displays.
DuoVideoSource	**DuoVideoSource: <None/1/2/3/4/5/6>** Defines which video input to be used as the default duo video source.
E1	**E1 Interface CRC4: <On/Off>**
Ethernet	**Ethernet Speed: <Auto/10half/10full/100half/100full>** Sets the Ethernet speed.
ExternalManager	**ExternalManager Address: <IPAddr>** **ExternalManager Path: <S: 0, 255>**
ExternalNetwork	**ExternalNetwork Clocking: <Dual/Single>** **ExternalNetwork Callcontrol: <RS366/RS366AdtranIMUX/LeasedLine/Manual>** **ExternalNetwork DTRPulse: <On/Off>**
FECC	**FECC Mode: <On/Off>** Enables/disables *Far End Camera Control*. If disabled the camera can not be controlled from a remote site.
FTP	**FTP Mode: <On/Off>** Enables/disables the FTP server.
G703	**G703 PhysicalLayer: <E1/T1>** **G703 Linecoding: <b8zsRestrict/b8zsNoRestrict>** **G703 Callcontrol: <Manual/Auto>**

	G703 Interface StartChannel: <1..31> **G703 Interface MaxChannels: <1..30>**
H320	**H320 NetType: <BRI/PRI/External/G703/None>** Defines the network type to use for H.320 calls.
H323Gatekeeper	**H323Gatekeeper Discovery: <Manual/Auto>** Determines whether the system shall register to a specific Gatekeeper, or automatically search for a Gatekeeper on the network. **H323Gatekeeper Address: <IPAddr>** The ip address of the Gatekeeper to register to, if *Discovery* is set to *Manual.* **H323Gatekeeper Authentication Mode: <Auto/Off>** **H323Gatekeeper Authentication ID: <S: 0, 49>** **H323Gatekeeper Authentication Password: <S: 0, 49>**
H323CallSetup	**H323CallSetup Mode: <Direct/Gatekeeper/CallManager>** Defines how to establish H.323 calls.
H323	**H323 Mode: <On/Off>** Enables/disables H.323.
H323Prefix	**H323Prefix: <S: 0, 4>** When dialing a number prefixed with digits specified by H.323 Prefix, and with Net: Auto, an H.323 call will be placed.
HTTP	**HTTP Mode: <On/Off>** Enables/disables HTTP.
HTTPS	**HTTPS Mode: <On/Off>** Enables/disables HTTPS.
IP	**IP Assignment: <DHCP/Static>** Selects whether to use *Dynamic Host Configuration Protocol* or static IP address allocation for the system. **IP Address: <IPAddr>** Defines the ip address for the system if static ip assignment is selected. **IP SubnetMask: <IPAddr>** Defines the subnet mask for the system. **IP Gateway: <IPAddr>** Defines the gateway address to use.

	IP DNS Server [1..5] Address: <IPAddr> **IP DNS Domain Name: <S: 0, 64>**
IPMedia	**IPMedia MaxVideoTXRate: <64..4096>**
IRControl	**IRControl Mode: <On/Off>** Enables/disables the systems IR sensor. If disabled, the system can not be controlled by the remote control.
ISDN	**ISDN SendComplete: <On/Off>** Enables/disables sending of the ISDN message information element Sending Complete. **ISDN SendNumber: <On/Off>** Enables/disables sending of own number when placing ISDN calls. NOTE! Some networks will however still send own number even though SendNumber is set to *Off.* **ISDN ParallelDial: <On/Off>** If set to *On,* channels will be dialed and connected in parallel when setting up a BONDING call. If set to *Off,* Channels will be dialed one by one, which may increase the dialing time. **ISDN HLC: <On/Off>** Turns sending of HLC information element in setup message *on* or *off* (video calls only)**.** **ISDN SpeechTimers: <On/Off>** **ISDN MSN: <On/Off>** Enables/disables the use of MSN (Multiple Subscriber Number). **ISDN SubAddress: <S: 0, 20>** Specifies an ISDN subaddress for the system. **ISDN PRI NSFTelephony Mode: <On/Off>** Turns Network Service Facilities for telephony on or off. **ISDN PRI NSFTelephony Number: <0..31>** Specifies the Network Service Facility to use for telephony if enabled. **ISDN PRI NSFVideoTelephony Mode: <On/Off>** Turns Network Service Facilities for videotelephony on or off. **ISDN PRI NSFVideoTelephony Number: <0..31>** Specifies the Network Service Facility to use for videotelephony if enabled. **ISDN PRI SwitchType: <NI/ATT/Euro/Japan>** Selects PRI network type. **ISDN PRI InitialRestart: <On/Off>** Enables/disables initial restart procedures. **ISDN PRI Alert: <On/Off>** If set to *On,* the ISDN protocol responds with an alert message to all

	incomming setup messages.(unless the call is rejected). Includes setups for additional channels. If set to *Off*, the ISDN protocol does only respond with an alert message to the incoming setup message related to the initial channel. **ISDN PRI ChanId: <On/Off>** **ISDN PRI L2WindowSize: <1..7>** **ISDN PRI T301Timer Mode: <Auto/On>** **ISDN PRI T301Timer Value: <1..500>** **ISDN PRI Interface MaxChannels: <1..30>** Used to define channel hunting strategy for the PRI interface. **ISDN PRI Interface HighChannel: <1..31>** Used to define channel hunting strategy for the PRI interface. **ISDN PRI Interface LowChannel: <1..31>** Used to define channel hunting strategy for the PRI interface. **ISDN PRI Interface Search: <High/Low>** Used to define channel hunting strategy for the PRI interface. **ISDN PRI Interface NumberRangeStart: <S: 0, 24>** Used to define the number range for the PRI interface. **ISDN PRI Interface NumberRangeStop: <S: 0, 24>** Used to define the number range for the PRI interface. **ISDN BRI SwitchType: <NI/ATT/Euro/1TR6/Japan/Australia/FETEX>** Selects BRI network type. **ISDN BRI AutoActivation: <Off/Selected/All>** **ISDN BRI MaxDeactiveTime: <1..60>** **ISDN BRI Alert: <On/Off>** If set to *On,* the ISDN protocol responds with an alert message to all incoming setup messages.(unless the call is rejected). Includes setups for additional channels. If set to *Off*, the ISDN protocol does only respond with an alert message to the incoming setup message related to the initial channel. **ISDN BRI ChanId: <On/Off>** **ISDN BRI Interface [1..6] Mode: <On/Off>** Enables/disables the various BRI interfaces. **ISDN BRI Interface [1..6] DirectoryNumber [1..2]: <S: 0, 24>** Specifies the Directory Numbers for the various BRI interfaces. **ISDN BRI Interface [1..6] SPID [1..2]: <S: 0, 20>** Specifies the SPID numbers for the various BRI interfaces.
Keyboard	**Keyboard Layout: <English/US/Norwegian/Swedish/German/French/User>**

	Defines the layout of the Keyboard if connected.
LocalLayout	**LocalLayout Mode: <Full/2Split/POP/POPWide>** Defines the picture layout to use on the local main monitor. *POP* and *POPWide* should only be used on wide screen monitors. **LocalLayout Toggle: <PIP/POP>** Defines the behavior of the *Layout* button on the remote control. If set to *POP* the system will toggle through the different *LocalLayout* modes when pressing the *Layout* button. If set to *PIP*, the system will display a *PIP* when pressing the *Layout* button.
Logo	**Logo: <On/Off>** If set to *On* the system will display a company logo will appear in the background of the welcome menu. NOTE! The TANDBERG logo will be displayed if no other company logo is uploaded.
LoS	**LoS Duration Exponent: <10..30>** **LoS Duration Offset: <0..65535>** **LoS Inhibit: <0..65535>** **LoS Initial: <0..65535>** **LoS Polarity: <Positive/Negative>** **LoS Retry: <0..65535>**
MainVideoSource	**MainVideoSource: <1..6>** Defines which video input to be used as the main video source.
NAT	Configuration of NAT (Network Address Translation) router. **NAT Mode: <On/Off/Auto>** **NAT Address: <IPAddr>**
NetProfile	**NetProfile [1..7] Name: <S: 0, 7>** Defines the name of a specific netprofile. **NetProfile [1..7] CallPrefix: <S: 0, 9>** Assigns a call prefix to a specific netprofile. **NetProfile 1 Network: <Auto>** Netprofile 1 is preprogrammed to *Auto* and can not be changed. **NetProfile 2 Network: <H320>** Netprofile 2 is preprogrammed to *H320* and can not be changed. **NetProfile 3 Network: <H323>** Netprofile 3 is preprogrammed to *H323* and can not be changed. **NetProfile 4 Network: <H320/H323/SIP/Auto>**

	Specifies the network to be associated with a netprofile. **NetProfile 5 Network: <H320/H323/SIP/Auto>** Specifies the network to be associated with a netprofile. **NetProfile 6 Network: <H320/H323/SIP/Auto>** Specifies the network to be associated with a netprofile. **NetProfile 7 Network: <SIP>** Netprofile 7 is preprogrammed to *SIP* and can not be changed.
OptionKey	**OptionKey Features: <S: 0, 16>** **OptionKey Bandwidth: <S: 0, 16>**
OSD	On Screen Display configurations. **OSD Mode: <On/Off>** Enables/disables *On Screen Display.* **OSD Menu Password: <S: 0, 5>** **OSD Menu Language: <English/German/Norwegian/French/Swedish/ Italian/Portuguese/Japanese/Chinese/TraditionalChinese/Russian/ Spanish/Korean/Finnish/Thai>** **OSD Menu WelcomeMenu: <On/Off>** **OSD Menu DisplayWelcomeText: <On/Off>** **OSD Menu WelcomeText: <S: 0, 30>** **OSD Menu DisableTimeout: <On/Off>** **OSD Menu BallonHelp: <On/Off>** **OSD MCUStatusLine Mode: <On/Off/Auto>**
PictureProgram	The PictureProgram configurations allow the user/control system to compose as many as four pictures using the supported layouts in the system and the basic pictures. A Picture Program can be set on a Virtual Monitor by the command *VirtualMonitorSet*. What Virtual Monitor to display on a local video output is configured from the output itself (ref. Video Output configurations). **PictureProgram [1..4] Layout: <Full/2Split/4Split/2+1Split/3+1Split/5+1Split>** Defines the layout to use for a specific Picture Program. **PictureProgram [1..4] Window [1..6] Picture: <LocalMain/LocalDuo/** **RemoteMain/RemoteDuo/JPEG/TandbergMonitor1/ TandbergMonitor2/None>** Defines what basic pictures to use in the sub windows of the layout.

	PictureProgram [1..4] Window [1..6] Call: <1..11> If remote pictures (RemoteMain/RemoteDuo) are defined for any of the sub windows of the layout, a reference to the given call must be supplied.
Preset	The Preset configurations should never be manually modified. To store presets use the commad *PresetStore*. **Preset [1..15] Name: <S: 0, 20>** **Preset [1..15] Audio Inputs Microphone [1..3] Mode: <On/Off>** **Preset [1..15] Audio Inputs Line [1..3] Mode: <On/Off>** **Preset [1..15] MainVideoSource: <1..6>** **Preset [1..15] DuoVideoSource: <0..6>** **Preset [1..15] Camera Pan: <0..65534>** **Preset [1..15] Camera Tilt: <0..65534>** **Preset [1..15] Camera Zoom: <0..65534>** **Preset [1..15] Camera Focus: <0..65534>** **Preset [1..15] Camera Brightness Mode: <Manual/Auto>** **Preset [1..15] Camera Brightness Level: <0..16>**
QoS	Defines Quality of Service settings. **QoS Precedence Telephony Audio: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Precedence Telephony Signalling: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Precedence VideoTelephony Audio: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Precedence VideoTelephony Signalling: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Precedence VideoTelephony Video: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Precedence VideoTelephony Data: <0/1/2/3/4/5/6/7/Auto/Off>** **QoS Diffserv Telephony Audio: <0..63>** **QoS Diffserv Telephony Signalling: <0..63>** **QoS Diffserv VideoTelephony Audio: <0..63>** **QoS Diffserv VideoTelephony Signalling: <0..63>** **QoS Diffserv VideoTelephony Video: <0..63>**

	QoS Diffserv VideoTelephony Data: <0..63> **QoS Mode: <Precedence/Diffserv/Off>** **QoS ToS: <MinDelay/MaxThrough/MaxReliable/MinCost/Off>** **QoS RSVP: <Auto/Off>**
RTP	**RTP Ports: <Static/Dynamic>** **RTP MTU: <1200..1400>**
Screensaver	**Screensaver Mode: <On/Off>** Enables/disables if screensaver is to be activated if there is now system activity. **Screensaver Delay: <1..480>** Specifes how long the system shall wait before activating screensaver when there is now system activity.
SelfViewOnStartup	**SelfViewOnStartup: <On/Off>** Specifies whether to display selfview on startup.
SerialPort	Defines various serial port settings. **SerialPort [1..2] BaudRate: <1200/2400/4800/9600/19200/38400/57600/115200>** **SerialPort [1..2] Parity: <None/Odd/Even>** **SerialPort [1..2] DataBits: <7/8>** **SerialPort [1..2] StopBits: <1/2>** **SerialPort 1 Mode: <Control >** **SerialPort 2 Mode: <VISCA/Auto>**
SIP	**SIP Mode: <On/Off>** **SIP Server Discovery: <Manual/Auto>** **SIP Server Address: <S: 0, 255>** **SIP Authentication UserName: <S: 0, 80>** **SIP Authentication Password: <S: 0, 60>** **SIP Transport Default: <TCP/UDP>**
SNMP	**SNMP Mode: <On/Off/ReadOnly/TrapsOnly>** **SNMP CommunityName: <S: 0, 16>**

	SNMP SystemContact: <S: 0, 70> **SNMP SystemLocation: <S: 0, 70>** **SNMP HostIPAddr [1..3]: <IPAddr>**
StillImageSource	**StillImageSource: <0/1/2/3/4/5/6>**
Streaming	**Streaming Port: <0..65535>** **Streaming Hops: <0..255>** **Streaming Address: <IPAddr>** **Streaming VideoRate: <16/32/64/128/192/256/320>** **Streaming Announcements: <On/Off>** **Streaming Source: <Local/Remote/Auto>** **Streaming Password: <S: 0, 16>** **Streaming AllowRemoteStart: <On/Off>**
StrictPassword	**StrictPassword: <On/Off>** If set to *On* the system requires that the system unit password has to have a minimum length of 8 characters and include both characters and numbers.
SystemUnit	**SystemUnit Name: <S: 0, 49>** Defines the system unit name. **SystemUnit DisplayName: <S: 0, 50>** **SystemUnit Password: <S: 0, 16>** Defines the password needed to access the system through IP.
T1	**T1 Interface CableLength: <Range1/Range2/Range3/Range4/Range5>**
TelnetChallenge	**TelnetChallenge Mode: <On/Off>** **TelnetChallenge Port: <23/57>**
Video	**Video Inputs Source [1..6] Name: <S: 0, 10>** All input sources can be assigned a name. **Video Inputs Source [1..6] Quality: <Motion/Sharpness>** Input sources configured to*Sharpness* will be transmitted with higher resolution, but with lower framrate. Input sources configured to *Motion* will be transmitted with higher framerate and lower resolution. **Video Outputs ScreenFormatTV: <4:3/16:9>** Specifies whether the monitors connected to the S-video/Composite

	outputs of the system are 4:3 monitors or 16:9 monitors. **Video Outputs ScreenFormatPC: <4:3/16:9>** Specifies whether the monitors connected to the DVI outputs of the system are 4:3 monitors or 16:9 monitors. **Video Outputs FormatPCWideScreen: <Normal/Wide>** This setting only takes effect when ScreenFormatTV or ScreenFormatPC is set to Wide. Use this setting to determine if you want your PC presentations to be shown stretched in full screen, or with correct aspect ratio using part of the widescreen display. **Video Outputs DVIResolution: <Auto/SVGA/XGA>** Specifies the resolution of the DVI outputs. **Video Outputs AllowHD720p: <On/Off>** **Video Outputs AllowWXGA: <On/Off>** **Video Outputs TV [1..2] VirtualMonitor: <1..4>** Defines what *VirtualMonitors* the S-Video/Composite outputs should display. Virtual monitor 1 is by default the main monitor, while Virtual monitor 2 is the secondary monitor. **Video Outputs TV [1..2] OSD: <On/Off>** Specifies whether or not to use On Screen Display on the S-video/Composite video outputs. **Video Outputs DVI [1..2] VirtualMonitor: <1..4>** Defines what *VirtualMonitors* the DVI outputs should display. Virtual monitor 1 is by default the main monitor, while Virtual monitor 2 is the secondary monitor. **Video Outputs DVI [1..2] OSD: <On/Off>** Specifies whether or not to use On Screen Display on the DVI video outputs.
VNC	**VNC IPAddress: <IPAddr>** **VNC DisplayNumber: <S: 0, 5>** **VNC Password: <S: 0, 8>**

3.2 directory.xml – xdirectory

LocalEntry	**LocalEntry [1..200] Name: <S: 0, 16>** **LocalEntry [1..200] Number: <S: 0, 30>** **LocalEntry [1..200] SecondNumber: <S: 0, 30>** **LocalEntry [1..200] SubAddress: <S: 0, 10>** **LocalEntry [1..200] CallRate: <Tlph/1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto>** **LocalEntry [1..200] Restrict: <On/Off>** **LocalEntry [1..200] NetProfile: <1..6>**
GroupEntry	**GroupEntry [1..50] Name: <S: 0, 16>** **GroupEntry [1..50] LocalEntryId [1..10]: <0..200>**
GlobalEntry	**GlobalEntry [1..400] Name: <S: 0, 16>** **GlobalEntry [1..400] Number: <S: 0, 30>** **GlobalEntry [1..400] SecondNumber: <S: 0, 30>** **GlobalEntry [1..400] SubAddress: <S: 0, 10>** **GlobalEntry [1..400] CallRate: <Tlph/1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto>** **GlobalEntry [1..400] Restrict: <On/Off>** **GlobalEntry [1..400] NetProfile: <1..6>**

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

Boot	Command used to reboot the system.
	Parameters:
	• **ParameterRestore: <On/Off>** 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.
	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 boot`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CallAccept	Command used to answer an incoming call if autoanswer is disabled.
	Parameters:
	None
	OK Result parameters:
	None
	ERROR Result parameters:
	• **Cause: <1…>** Cause code specifying why the command was not accepted.
	• **Description** Textual description of the cause code.
	Example:
	`xcommand callaccept`
	
	`*r Result (status=OK): /`
	`*r/end`
	
	`OK`

CallMute	Command used to mute incoming audio from a specific call in a MultiSite conference. **Parameters:** • **Call(r): <1..11>** Reference to the call to be muted or unmuted. • **Mode(r): <On/Off>** Denotes whether the call is to be muted or unmuted. **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 callmute call:2 mode:on`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraFocus	Command used to change focus of a specific camera.
	Parameters:
	• **Camera(r): <1..5>** Addresses the specific camera.
	• **Value(r): <Auto/Manual/+/->** Specifies the wanted operation.
	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 camerafocus camera:1 value:+`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraHalt	Command used to stop moving a specific camera.
	Parameters:
	None
	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 camerahalt`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraMove	Command used to instruct the camera to move in a specified direction. The camera will continue moving until the *CameraHalt* command is issued. **Parameters:** • **Camera(r): <1..5>** Addresses the camera to move. • **Direction(r): <Up/Down/Right/Left/In/Out>** Specifies the direction to move. **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 cameramove camera:1 direction:right`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraPosition	Command used to instruct the camera to move to a specific postion.
	Parameters:
	• **Camera(r): <1..5>** Addresses the camera to position.
	• **Pan: <0..65534>** Pan value.
	• **Tilt: <0..65534>** Tilt value.
	• **Zoom: <0..65534>** Zoom value.
	• **Focus: <0..65534>** Focus value.
	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 cameraposition camera:1 pan:1700 tilt:1700`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraTrackingStart	Command used to turn cameratracking on.
	Parameters:
	None
	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 cameratrackingstart`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraTrackingStop	Command used to turn cameratracking off. **Parameters:** None **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 cameratrackingstop`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CameraWhiteBalance	Command used to change whitebalance of a specific camera.
	Parameters:
	• **Camera(r): <1..5>** Addresses the specific camera.
	• **Value(r): <+/->** Specifies whether to increase or decrease the whitebalance.
	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 camerawhitebalance camera:1 value:+`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

ChairRelease	Command used to release chair in a conference supporting chair control.
	Parameters:
	None
	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 chairrelease`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

ChairTake	Command used to take chair in a conference supporting chair control. **Parameters:** None **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 chairtake`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

ConferenceDisconnect	Command used to disconnect all calls in a conference.
	Parameters:
	None
	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:
	`ConferenceDisconnect`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CorpDirGetNext	**Parameters:**
	• **Path: <S: 0, 256>**
	• **Query: <S: 0, 81>**
	• **StartsWith: <S: 0, 81>**
	• **Hits: <1..40>**
	• **ID: <S: 0, 21>**
	• **SubFolders: <On/Off>**
	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:
	`xcommmand corpdirgetnext query:jon id:34`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

CorpDirGetPrevious	**Parameters:**

	• **Path: <S: 0, 256>**Number to dial.
	• **Query: <S: 0, 81>** 2Xh221 second number
	• **StartsWith: <S: 0, 81>**Sub address
	• **Hits: <1..40>**Specifies the callrate to use
	• **ID: <S: 0, 21>**
	• **SubFolders: <On/Off>**
	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:
	`xcommmand corpdirgetprevious query:jon id:34`
	`*r Result (status=OK):`
	`*r/end`
	`OK`

CorpDirSearch	**Parameters:**
	• **Path: <S: 0, 256>**Number to dial.
	• **Query: <S: 0, 81>** 2Xh221 second number
	• **StartsWith: <S: 0, 81>**Sub address
	• **Hits: <1..40>**Specifies the callrate to use
	• **SubFolders: <On/Off>**
	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:
	`xcommmand corpdirsearch query:test`
	`*r Result (status=OK):`
	`*r/end`
	`OK`

DefaultValuesSet	Command used to reset configurations to default values. **Parameters:** • **Level: <1..3>** 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. **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 defaultvaluesset level:2

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

OK
```

Dial

Command used to initiate an outgoing call.

Parameters:

- **Number: <S: 0, 60>** Number to dial.
- **SecondNumber: <S: 0, 60>** 2Xh221 second number
- **SubAddress: <S: 0, 20>** 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..7>**
- **BillingCode: <S: 0, 16>**

OK Result parameters:

- **CallRef: <1..11>** 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: 1
     LogTag: 312
*r/end

OK
```

DialGlobalEntry

Command used to dial a number from the global directory (the Global Directory is downloaded to the system by an external application).

Parameters:

- **GlobalEntryId(r): <1..400>** Reference to the directory entry to be dialed.

OK Result parameters:

- **CallRef: <1..11>** 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 dialglobalentry globalentryid:19`
	
	`*r Result (status=OK):`
	`  CallRef: 1`
	`  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:
	• **GroupEntryId(r): <1..50 >** Reference to the directory entry to be dialed.
	
	OK Result parameters:
	The system will return the following elements for each call initiated.
	• **CallRef: <1..11>** 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 groupentryid:19`
	
	`*r Result (status=OK):`
	`  CallRef: 2`
	`  LogTag: 313`
	`  CallRef: 1`
	`  LogTag: 312`
	`  CallRef:`
	`*r/end`
	
	`OK`

DialLocalEntry	Command used to dial a number from the locally stored directory.
	
	Parameters:
	• **LocalEntryId(r): <1..200>** Reference to the directory entry to be dialed.
	
	OK Result parameters:
	• **CallRef: <1..11>** 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 diallocalentry localentryid:15`
	
	`*r Result (status=OK):`

	`CallRef: 1`
	`LogTag: 312`
	`*r/end`
	
	`OK`

DisconnectCall	Command used to disconnect a call.
	
	Parameters:
	• **Call: <1..11>** Reference to the call to be disconnected. If this parameter is omitted, all active calls in the system will be disconnected.
	
	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 disconnectcall call:9`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

DuoVideoStart	Command used to initiate DuoVideo from the system.
	
	Parameters:
	• **VideoSource: <1..6>** Specifies which video source to be used for the additional video stream. If this parameter is omitted, the system will use the default DuoVideo source configured for the system.
	
	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 duovideostart videosource:5`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

DuoVideoStop	Command used to stop DuoVideo. **Parameters:** None **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 duovideostop`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

DTMFSend	Command used to send DTMF tones to the far end.
	Parameters:
	• **Value(r): <E164: 1, 1>** The DTMF tone to send.
	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 dtmfsend value:5`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

FECCFocus	Command used to change focus of a farend camera.
	Parameters:
	• **Value(r): <+/->** Specifies whether to increase or decrease focus.
	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 feccfocus value:+`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

FECCMove	Command used to issue a Far End Camera Control - Move command. **Parameters:** • **Direction(r): <Up/Down/Right/Left/In/Out>** Specifies the direction to move. **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 feccmove direction:right`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

FECCPresetActivate	Command used to activate a farend preset.
	Parameters:
	• **Number(r): <0..15>** The preset number to activate.
	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 feccpresetactivate number:4`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

FECCPresetStore	Command used to store a farend preset.
	Parameters:
	• **Number(r): <0..15>** The preset number to activate.
	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 feccpresetstore number:4`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

FECCRequestStill	Command used to request a still image from a specific source on the farend side. **Parameters:** • **Source(r): <0..15>** The farend source to select. **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 requeststill source:4
*r Result (status=OK):
*r/end

OK
```

FECCSelectSource

Command used to select a farend source.

Parameters:
- **Source(r): <0..15>** The farend source to select.

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 feccselectsource source:4
*r Result (status=OK):
*r/end

OK
```

FeedbackDeregister

Command used to deregister XML feedback over HTTP(S).

Parameters:
- **ID: <1..3>** ID for the registration to deregister.

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 feedbackderegister id:1

*r Result (status=OK):
  ID: 2
*r/end

OK
```

FeedbackRegister

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.

Parameters:
- **ID: <1..3>** ID for the registration. If this parameter is omitted the system uses the first vacant ID.
- **URL(r): <S: 0, 256>** The URL to post feedback to.
- **Expression.1..15: <S: 0, 256>** XPath expression

OK Result parameters:
- **ID: <1..3>**

	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 feedbackregister url:http://10.47.14.185:8000 expression.1:status/call expression.2:status/conference`
	`*r Result (status=OK):`
	`ID: 2`
	`*r/end`
	`OK`

FloorRelease	Command used to release floor in a MultiSite conference.
	Parameters:
	None
	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 floorrelease`
	`*r Result (status=OK):`
	`*r/end`
	OK

FloorRequest	Command used to request floor in a MultiSite conference.
	Parameters:
	None
	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 floorrequest`
	`*r Result (status=OK):`
	`*r/end`
	OK

FloorToSite	Command used to assign floor to a specific site in a MultiSite conference supporting H.243. **Parameters:** • **MCUID(r): <1..191>** MCUID to the MultiSite the site is

	connected to.
	• **TerminalID(r): <1..191>** The site's terimnal id, referenced to the MultiSite it is connected to.
	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 floortosite mcuid:85 terminalid:2`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

FloorToSiteEnd	Command used to end the assignment of floor to a specific site in a MultiSite conference supporting H.243. Requires that the command *FloorToSite* has been issued in advance.
	Parameters:
	None
	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 floortositeend`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

GroupEntryAdd	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. **Parameters:** • **Name: <S: 0, 48>** The entry's name. • **LocalEntryId.1..10: <1..200>** References to local entry ids to be included in this Group entry. **OK Result parameters:** • **GroupEntryId: <1..50>** Reference to the Group Directory position the entry is stored. **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 groupentryadd name:"The Team" localentryid.1:17 localentryid.2:29`

	`localentryid.3:56`
	
	`*r Result (status=OK):`
	`  GroupEntryId: 17`
	`*r/end`
	
	OK

GroupEntryDelete	Command used to delete an entry in the locally stored Group Directory.
	
	Parameters:
	• **GroupEntryId(r): <1..50>** Reference to the entry to delete.
	
	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 groupentrydelete groupentryid:30`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

LocalEntryAdd	Command used to add a new entry to the locally stored Directory. The entry is stored in the first vacant position in the Directory.
	
	Parameters:
	• **Name: <S: 0, 48>** The entry's name.
	• **Number: <S: 0, 60>** The entry's number.
	• **SecondNumber: <S: 0, 60>** The entry's second number (2XH221 number).
	• **SubAddress: <S: 0, 10>** The entry's sub address.
	• **CallRate: <Tlph/1xh221/2xh221/64/128/192/256/320/384/512/768/1152/1472/1920/2560/3072/4096/H0/Max/Auto>** The callrate to use when calling this entry.
	• **Restrict: <On/Off>** Whether to use restrict or not when calling this entry.
	• **NetProfile: <1..7>** The Net Profile to use when calling this entry.
	
	OK Result parameters:
	• **LocalEntryId: <1..200>** Reference to the Directory position the entry is stored.
	
	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 localentryadd name:"John Galt" number:666`
	
	`*r Result (status=OK):`
	`  LocalEntryId: 17`
	`*r/end`
	
	OK

LocalEntryDelete	Command used to delete an entry in the locally stored Directory.
	Parameters:
	• **LocalEntryId(r): <1..200>** Reference to the entry to delete.
	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

MessageBoxDelete	Command used to delete a message box on the screen.
	Parameters:
	None
	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 messageboxdelete`
	`*r Result (status=OK):`
	`*r/end`
	
	OK

MessageBoxDisplay	Command used add a graphical message box on the screen.
	Parameters:
	• **Title(r): <S: 0, 40>** Message box title.
	• **Line.1..3: <S: 0, 40>** Text to be displayed on the lines within the box.
	• **Key.1..3: <S: 0, 15>** Text to be displayed on the keys.
	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 messageboxdisplay title:Welcome`
	`line.1:”How are you?”`
	`key.1:Good`
	`key.2:Bad`
	`*r Result (status=OK):`

	`*r/end` OK

PIPHide	Command used to hide a PIP on a specific VirtualMonitor.
	Parameters:
	• **VirtualMonitor(r): <1..4>** Addresses which VirtualMonitor to apply the command.
	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 piphide virtualmonitor:1`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

PIPShow	Command used to display a specific picture in a PIP on a selected VirtualMonitor.
	Parameters:
	• **VirtualMonitor(r): <1..4>** Addresses which VirtualMonitor to apply the command.
	• **Picture(r): <LocalMain/LocalDuo/RemoteMain/RemoteDuo/ JPEG/TandbergMonitor1/TandbergMonitor2/None>** Specifies which of the supported pictures to display in the PIP on the addressed VirtualMonitor.
	• **Call: <1..11>** If RemoteMain or RemoteDuo is selected, this parameter must be supplied to select the correct remote call.
	• **Position: <BottomLeft/BottomRight/TopLeft/TopRight>** Specifies where to postion the PIP.
	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 pipshow virtualmonitor:1 picture:remoteduo call:5`
	`postion:topright`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

PresetActivate	Command used activate a stored preset. **Parameters:** • **Number(r): <0..14>** The preset to activate.

	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 presetactivate number:4`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

PresetStore	Command used to store a preset.
	Parameters:
	• **Number(r): <0..14>** The number where to store the preset.
	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 presetstore number:4`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

ScreensaverActivate	Command used to activate screensaver.
	Parameters:
	None
	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 screensaveractivate`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

ScreensaverDeactivate	Command used to deactivate screensaver. **Parameters:** None **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 screensaverdeactivate`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

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

SiteDisconnect	Command used to disconnect a specific site from a MultiSite conference supporting H.243.
	Parameters:
	• **MCUID(r): <1..191>** MCUID to the MultiSite the site is connected to.
	• **TerminalID(r): <1..191>** The site's terimnal id, referenced to the MultiSite it is connected to.
	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 sitedisconnect mcuid:85 terminalid:2`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

SiteView	Command used to requset view of a specific site in a MultiSite conference supporting H.243.

	Parameters:
	• **MCUID(r): <1..191>** MCUID to the MultiSite the site is connected to.
	• **TerminalID(r): <1..191>** The site's terimnal id, referenced to the MultiSite it is connected to.
	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 siteview mcuid:85 terminalid:2`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

SiteViewEnd	Command used to end viewing of a specific site in a MultiSite conference supporting H.243. Requires that the *SiteView* command has been issued in advanced.
	Parameters:
	None
	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 floortositeend`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

SPIDAutoConfigure	Command used to initiate automatic configuration of SPIDs.
	Parameters:
	None
	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 spidautoconfigure`
	`*r Result (status=OK):`
	`*r/end`

	OK

StillImageSend	Command used to send a still image.
	Parameters:
	• **VideoSource: <1..6>** Specifies from which video source to send a still image. If this parameter is omitted, the system will use the default still image source configured for the system.
	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 stillimagesend videosource:5`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

StreamingStart	Command used to start streaming from the system.
	Parameters:
	None
	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 streamingstart`
	
	`*r Result (status=OK):`
	`*r/end`
	
	OK

StreamingStop	Command used to stop streaming from the system.
	Parameters:
	None
	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 streamingstop`
	
	`*r Result (status=OK):`
	`*r/end`

	OK

TextDelete	Command used to delete a text line added by the *TextDisplay* command.
	Parameters:
	• **Layer(r): <1..3>** The layer to delete.
	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 textdelete layer:1`
	`*r Result (status=OK):`
	`*r/end`
	`OK`

TextDisplay	Command used add a text line on screen.
	Parameters:
	• **Layer(r): <1..3>** Defines the lines position.
	• **Text: <S: 0, 38>** The text to display.
	• **TimeOut: <0..999>** Sets the timeout value for the text line.
	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 textdisplay layer:1 text:TANDBERG timeout:100`
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

VirtualMonitorReset	Command used to reset a VirtualMonitor. By resetting a VirtualMonitor the system itself retakes control over what to be displayed on the VirtualMonitor. **Parameters:** • **VirtualMonitor(r): <1..4>** Addresses which VirtualMonitor to apply the command. **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 virtualmonitorreset virtualmonitor:2`

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

VirtualMonitorSet	Command used to instruct the system to display a specific picture on a specific VirtualMonitor. A VirtualMonitor can be displayed on one or more of the local video outputs (which VirtualMonitor a specific video output are to display is configurable).
	When this command is issued for a specific VirtualMonitor the picture displayed on this monitor will not change until the VirtualMonitor is reset or set to display another picture.
	
	Parameters:
	• **VirtualMonitor(r): <1..4>** Addresses which VirtualMonitor to apply the command.
	• **Picture(r): <LocalMain/LocalDuo/Still/RemoteMain/RemoteDuo/ JPEG/TandbergMonitor1/TandbergMonitor2/PictureProgram1/ PictureProgram2/PictureProgram3/PictureProgram4/None>** Specifies which of the supported pictures to display on the addressed VirtualMonitor.
	• **Call: <1..11>** If RemoteMain or RemoteDuo is selected, this parameter must be supplied to select the correct remote call.
	
	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 virtualmonitorset virtualmonitor:2 picture:still`
	
	`*r Result (status=OK):`
	`*r/end`
	
	`OK`

5 API - Status

This section gives an overview of the Status Information availble 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

Audio

Top level attributes:
None

Summary:
- Gives information about what Audio Module currently in use
- Gives information about what audio inputs that are active, i.e. receives audio above a given level

Example

```
*s Audio:
     AudioModule: None
     Inputs:
       Microphone 1:
         Active: False
       Microphone 2:
         Active: False
       Microphone 3:
         Active: False
       Line 1:
         Active: False
       Line 2:
         Active: False
       Line 3:
         Active: False
*s/end
```

BRI [1..6]

Top level attributes:
- ready: True/False/Unknown Indicates whether the given interface is ready.

The attribute will always report *Unknown* if the configuration *ISDN BRI AutoActivation* is set to*Off*.

Summary:
- Gives ISDN B-channel information for the BRI interfaces

Examples

```
*s BRI 1 (ready=True):
     Channel 1 (type=BChannel, status=Connect):
       CallingNumber: "8774"
       ConnectionTime: 11
     Channel 2 (type=BChannel, status=Idle): /
*s/end

*s BRI 1 (ready=True):
     Channel 1 (type=BChannel, status=Disconnected):
       CallingNumber: "8774"
       CauseLocation: 0
       ChannelCause: 16
       ConnectionTime: 20
     Channel 2 (type=BChannel, status=Idle): /
*s/end

*s BRI 1 (ready=Unknown):
```

```
     Channel 1 (type=BChannel, status=Idle): /
     Channel 2 (type=BChannel, status=Idle): /
*s/end

*s BRI 1 (ready=False):
     Layer1Alarm: On
     Layer2Alarm: Off
*s/end
```

Call [1..11]

Top level attributes:

- **status: CallIDLE/Dialing/Alerting/Proceeding/EstablOut/ EstablIn/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*).

Summary:

- Returns all currently availible information for a call.

Examples:

```
*s Call 1 (status=Synced, type=Vtlph, protocol=H323,
direction=Outgoing, logTag=1):
     CallRate: 768
     RemoteNumber: "10.47.15.127"
     Mute: Off
     Microphone: Off
     Duration: 719
     Channels 1 (type=Incoming):
       Rate: 768
       Restrict: Off
       Encryption (status=Off): /
       Audio (status=Active):
         Protocol: G722
         Rate: 64
         RemoteIPAddress: ""
         LocalIPAddress: "10.47.8.37:2326"
         Encryption (status=Off): /
         RSVP: Off
         RSVPRate: 0
         DynamicRate: 64
         TotalPackets: 17868
         PacketLoss: 0
         Jitter: 0
       Video 1 (status=Active):
         Protocol: H263
         Resolution: CIF
         Rate: 704
         RemoteIPAddress: ""
         LocalIPAddress: "10.47.8.37:2328"
         Encryption (status=Off): /
         RSVP: Off
         RSVPRate: 0
         DynamicRate: 24
         TotalPackets: 18013
         PacketLoss: 0
         Jitter: 1
       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
          RemoteIPAddress: "10.47.15.127:2326"
          LocalIPAddress: "10.47.8.37:2334"
          Encryption (status=Off): /
          RSVP: Off
          RSVPRate: 0
          DynamicRate: 64
          TotalPackets: 35729
          PacketLoss: 0
          Jitter: 0
        Video 1 (status=Active):
          Protocol: H264
          Resolution: CIF
          Rate: 704
          RemoteIPAddress: "10.47.15.127:2328"
          LocalIPAddress: "10.47.8.37:2336"
          Encryption (status=Off): /
          RSVP: Off
          RSVPRate: 0
          DynamicRate: 355
          TotalPackets: 50051
          PacketLoss: 0
          Jitter: 17
        Video 2 (status=Inactive): /
        Data (status=Inactive): /
*s/end

*s Call 2 (status=Synced, type=Vtlph, protocol=H320,
direction=Outgoing, logTag=2):
      CallRate: 384
      Bonding: On
      RemoteNumber 1: "8774"
      RemoteNumber 2: ""
      RemoteSubAddress: ""
      Mute: Off
      Microphone: On
      Duration: 235
      Channels 1 (type=Incoming):
        Rate: 384
        Restrict: Off
        Encryption (status=Off): /
        Audio (status=Active):
          Protocol: G722
          Rate: 56
        Video 1 (status=Active):
          Protocol: H264
          Resolution: SIF
          Rate: 326
        Video 2 (status=Inactive): /
        Data (status=Inactive): /
      Channels 2 (type=Outgoing):
        Rate: 384
        Restrict: Off
        Encryption (status=Off): /
```

```
       Audio (status=Active):
         Protocol: G722
         Rate: 56
       Video 1 (status=Active):
         Protocol: H264
         Resolution: CIF
         Rate: 326
       Video 2 (status=Inactive): /
       Data (status=Inactive): /
*s/end

*s Call 3 (status=Synced, type=Tlph, protocol=ISDN,
direction=Outgoing, logTag=5):
     CallRate: Tlph
     RemoteNumber 1: "8774"
     RemoteNumber 2: ""
     RemoteSubAddress: ""
     Mute: Off
     Microphone: On
     Duration: 4
     Channels 1 (type=Incoming):
       Rate: 64
       Restrict: Off
       Audio (status=Active):
         Protocol: G711
         Rate: 64
     Channels 2 (type=Outgoing):
       Rate: 64
       Restrict: Off
       Audio (status=Active):
         Protocol: G711
         Rate: 64
*s/end

*s Call 3 (status=Disconnected, type=NA, protocol=NA,
direction=NA, logTag=NA):
     Cause: 255
*s/end
```

Camera [1..5]

Top level attributes:

- **connected: True/False** Indicates whether the given camera is connected or not

Summary:

- Returns Camera ID and current position

Examples:

```
*s Camera 1 (connected=True):
     ID: "TT000003"
     Pan: 916
     Tilt: 66
     Zoom: 0
     Focus: 24576
*s/end

*s Camera 2 (connected=False): /
*s/end
```

Camera

Top level attributes:

Tracking

- **status: On/Off** Indicates whether Camera Tracking is active or not

Summary:

- Returns current Camera Tracking status

Examples:

```
*s CameraTracking (status=On): /
*s/end

*s CameraTracking (status=Off): /
*s/end
```

Conference

Top level attributes:

- **type: PointToPoint/PointToMultiSite/MultiSite**

Summary:

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

Examples:

NOTE! CallRef = 0 indicates the local participant.

```
*s Conference (type=PointToPoint):
     Calls:
       CallRef 1: 1
     DuoVideo (status=Ready): /
     LoudestParticipant:
       CallRef: 1
*s/end

*s Conference (type=PointToMultisite):
     Calls:
       CallRef 1: 1
     DuoVideo (status=None): /
     LoudestParticipant:
       CallRef: 1
     MCUSiteList:
       Site 1:
         MCUID: 85
         TerminalID: 1
         Name: "System035-NOR"
         CallRef: None
       Site 2:
         MCUID: 85
         TerminalID: 2
         Name: "System037-NOR"
         CallRef: 0
       Site 3:
         MCUID: 85
         TerminalID: 3
         Name: "System036-NOR"
         CallRef: None
     OnAir:
       MCUID: 85
       TerminalID: 3
```

```
*s/end

*s Conference (type=Multisite):
     Calls:
       CallRef 1: 1
       CallRef 2: 2
     DuoVideo (status=Ready): /
     LoudestParticipant:
       CallRef: 2
     Floor: None
     Current:
       CallRef: None
     Previous:
       CallRef: None
     OutgoingPicture 1 (name=Current):
       Layout (type=4Split):
         Window 1:
           Picture: LocalMain
           CallRef: 0
         Window 2:
           Picture: RemoteMain
           CallRef: 1
         Window 3:
           Picture: RemoteMain
           CallRef: 2
         Window 4:
           Picture: NA
           CallRef: None
     OutgoingPicture 2 (name=Previous):
       Layout (type=Full):
         Window 1:
           Picture: NA
           CallRef: None
     OutgoingPicture 3 (name=Duo):
       Layout (type=NA): /
     MCUID: 1
     CascadingMode: StandAlone
     MCUSiteList:
       Site 1:
         MCUID: 1
         TerminalID: 2
         Name: " System035-NOR "
         CallRef: 1
       Site 2:
         MCUID: 1
         TerminalID: 3
         Name: "System036-NOR"
         CallRef: 2
       Site 3:
         MCUID: 1
         TerminalID: 4
         Name: "System037-NOR"
         CallRef: 0
     OnAir:
       MCUID: 1
       TerminalID: 0
*s/end
```

Ethernet	**Top level attributes:** None

Summary:
- Returns MAC Address
- Returns Ethernet speed

Example

```
*s Ethernet:
     MacAddress: "00:50:60:7F:FA:FB"
     Speed: 10full
*s/end
```

FarEnd Information

Top level attributes:
None

Summary:
- Returns Far End Capabillities

Example

```
*s FarEndInformation:
     FECC (status=On):
       NumberOfPresets: 15
       NumberOfSources: 5
       Source 1:
         Name: "main cam"
         Capabilities: "ptzfms"
       Source 2:
         Name: "aux"
         Capabilities: "ms"
       Source 3:
         Name: "doc cam"
         Capabilities: "ms"
       Source 4:
         Name: "vcr"
         Capabilities: "ms"
       Source 5:
         Name: "pc"
         Capabilities: "ms"
       CurrentSource: 1
       BroadcastSwitch: On
     T140: On
     SString: On
*s/end
```

Feedback [1..3]

Top level attributes:
- **status: on/off** Indicates if there are HTTP feedback registered for a given Feedback ID, ref. command *FeedbackRegister*.

Summary:
- Lists the URL and feedback expressions registered for the given Feedback ID

Examples:

```
*s Feedback 1 (status=Off): /
*s/end

*s Feedback 1 (status=On):
     URL: "http://10.47.14.185:8000/"
     Expression: "status/call[@status="Synced"]"
     Expression: ""
     Expression: ""
     Expression: ""
```

	Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" Expression: "" *s/end
H323 Gatekeeper	**Top level attributes:** • **status: Required/Discovering/Discovered/Authenticating/ Authenticated/Registering/Registered /Rejected/Inactive** **Summary:** • Returns H323Gatekeeper status **Examples:** *s H323Gatekeeper (status=Inactive): / *s/end *s H323Gatekeeper (status=Registered): Address: "10.47.9.1" Port: 1719 *s/end *s H323Gatekeeper (status=Rejected): Address: "10.47.9.0" Port: 0 *s/end
IP	**Top level attributes:** None **Summary:** • Returns current IP address, Subnet Mask and Gateway address **Example** *s IP: Address: "10.47.8.26" SubnetMask: "255.255.248.0" Gateway: "10.47.8.1" *s/end
PRI	**Top level attributes:** • **ready: True/False** Indicates wheter the interface is ready or not **Summary:** • Returns information about the PRI interface and the ISDN B-channels **Examples:** *s PRI (ready=True): BChannelsTotal: 30 BChannelsFree: 30 H0ChannelsFree: 5 Channel 1 (type=BChannel, status=Idle): /

```
     Channel 2 (type=BChannel, status=Idle): /
     Channel 3 (type=BChannel, status=Idle): /
     Channel 4 (type=BChannel, status=Idle): /
     Channel 5 (type=BChannel, status=Idle): /
     Channel 6 (type=BChannel, status=Idle): /
     Channel 7 (type=BChannel, status=Idle): /
     Channel 8 (type=BChannel, status=Idle): /
     Channel 9 (type=BChannel, status=Idle): /
     Channel 10 (type=BChannel, status=Idle): /
     Channel 11 (type=BChannel, status=Idle): /
     Channel 12 (type=BChannel, status=Idle): /
     Channel 13 (type=BChannel, status=Idle): /
     Channel 14 (type=BChannel, status=Idle): /
     Channel 15 (type=BChannel, status=Idle): /
     Channel 16 (type=DChannel, status=NA): /
     Channel 17 (type=BChannel, status=Idle): /
     Channel 18 (type=BChannel, status=Idle): /
     Channel 19 (type=BChannel, status=Idle): /
     Channel 20 (type=BChannel, status=Idle): /
     Channel 21 (type=BChannel, status=Idle): /
     Channel 22 (type=BChannel, status=Idle): /
     Channel 23 (type=BChannel, status=Idle): /
     Channel 24 (type=BChannel, status=Idle): /
     Channel 25 (type=BChannel, status=Idle): /
     Channel 26 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 394
     Channel 27 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 394
     Channel 28 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 394
     Channel 29 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 394
     Channel 30 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 395
     Channel 31 (type=BChannel, status=Disconnected):
       CallingNumber: "6830"
       CauseLocation: 2
       ChannelCause: 16
       ConnectionTime: 395
*s/end

*s PRI (ready=False):
     Layer1Alarm: RedAlarm
*s/end
```

Screen saver

Top level attributes:

- **status: On/Off** Indicates wheter Screensaver is active or not

Summary:

- Returns current Screensaver status and the current timer value (time left before the screensaver is activated if not allready active)

Examples:

```
*s Screensaver (status=On): /
*s/end

*s Screensaver (status=Off):
     Timer: 9
*s/end
```

SystemUnit

Top level attributes:
None

Summary:

- Returns information about the System Unit

Example

```
*s SystemUnit:
     ProductType: "TANDBERG Codec"
     ProductId: "TANDBERG CODEC 6000MXP"
     Uptime: 1706
     Software:
       Version: "F1.1 PAL"
       Name: "s50000"
       ReleaseDate: "2004-09-01"
       Configuration:
         Telephony: 5
         VideoTelephony: 5
         TotalBandwidth: 6144
         ISDNBandwidth: 1920
         LANBandwidth: 4096
         PresenterOption: True
         MultisiteOption: True
         StreamingSupport: True
         Encryption: True
     Hardware:
       SerialNumber: "25A00307"
       MainBoard: "100670 rev. 05"
       AdditionalBoard: ": 1107"
       BootSoftware: "Rev. 1.2, 2004-06-14"
       Configuration:
         PRI: 1
         BRI: 6
         ExternalNetwork: 1
         VGA: 2
         DataPorts: 2
         AudioInputs: 6
         Settop: False
         TV-Standard: PAL
     TemperatureCelcius: 41
     TemperatureFahrenheit: 105
*s/end
```

VirtualMonitor

Top level attributes:

[1..4]	None
	Summary:
	• Returns information about pictures currently dispalyed on the Virtual Monitors.
	Example
	`*s VirtualMonitor 1:`
	`  Layout (type=5+1Split, persistent=On):`
	`  Window 1:`
	`  Picture: LocalMain`
	`  CallRef: 0`
	`  Window 2:`
	`  Picture: RemoteMain`
	`  CallRef: 1`
	`  Window 3:`
	`  Picture: RemoteMain`
	`  CallRef: 3`
	`  Window 4:`
	`  Picture: RemoteMain`
	`  CallRef: 5`
	`  Window 5:`
	`  Picture: RemoteMain`
	`  CallRef: 4`
	`  Window 6:`
	`  Picture: RemoteMain`
	`  CallRef: 7`
	`  PIP (status=On):`
	`  Position: TopRight`
	`  Picture: LocalDuo`
	`  CallRef: 0`
	`*s/end`
	
	`*s VirtualMonitor 2:`
	`  Layout (type=Full, persistent=Off):`
	`  Window 1:`
	`  Picture: LocalMain`
	`  CallRef: 0`
	`  PIP (status=Off): /`
	`*s/end`

5.2 history.xml – xhistory

Call [1..20]	**Top level attributes:**
	• **type: Tlph/Vtlph**
	• **protocol: H320/H323**
	• **direction: Incoming/Outgoing**
	Summary:
	• Returns information about disconnected calls
	Examples:
	`*l Call 1 (type=Vtlph, protocol=H323, direction=Outgoing, logTag=2):`
	`LogTag: 2`
	`RemoteNumber: "10.47.15.127"`
	`CallRate: 768`
	`DisconnectCauseValue: 16`
	`Duration: 5`
	`UptimeAtEndOfCall: 4657`
	`BillingCode: ""`
	`*l/end`
	
	`*l Call 2 (type=Vtlph, protocol=H320, direction=Outgoing, logTag=1):`
	`LogTag: 1`
	`RemoteNumber: "4566"`
	`CallRate: 128`
	`Bonding: Off`
	`DisconnectCauseValue: 16`
	`Duration: 14`
	`UptimeAtEndOfCall: 64204`
	`BillingCode: ""`
	`*l/end`