

Codian MCU 4200 Series

Remote Management API

Version 2.1.1



Contents

1	Introduction.....	5
1.1	HTTP	5
1.2	XML-RPC	5
2	Authentication	5
2.1	Message Flow	5
3	Messages	8
3.1	Common Message Elements.....	8
3.1.1	Authentication.....	8
3.1.2	Participant identification parameters.....	8
3.2	device.query	9
3.3	device.network.query	9
3.4	device.health.query	10
3.5	device.restartlog.query	11
3.6	gatekeeper.query	11
3.7	conference.create.....	12
3.8	conference.modify	14
3.9	conference.destroy	15
3.10	conference.end.....	15
3.11	conference.enumerate	16
3.12	conference.streaming.query	18
3.13	conference.paneplacement.query	19
3.14	conference.paneplacement.modify	20
3.15	participant.add	21
3.16	participant.remove	23
3.17	participant.modify.....	23
3.18	participant.connect.....	24
3.19	participant.disconnect	25
3.20	participant.move.....	25
3.21	participant.enumerate	25
3.22	participant.fecc.....	31
3.23	participant.message	32
3.24	participant.diagnostics	32
3.25	autoAttendant.enumerate.....	33
3.26	autoAttendant.destroy.....	34
4	Deprecated messages	35
4.1	system.query	35
4.2	conference.query	35
4.3	conference.participant.modify	36
4.4	conference.participant.remove	36
4.5	conference.participant.add	36
5	Related information sources.....	37
5.1	system.xml	37
6	Required user privileges.....	38
7	Fault Codes	39
8	Participant disconnect reasons	40
9	References	42

Appendix A - Conference Layouts	43
Appendix B - Linking conferences across MCUs	46
B.1 Example message 1 - creating conference "linked1" on "mcu1"	46
B.2 Example message 2 - creating conference "linked2" on "mcu2"	47
B.3 Example message 3 - calling into "linked2" from "linked1"	47
B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout	48
B.5 Message responses	49

Revision History

Version	Date	Author	Comments
1.0	20 Apr 2005	MKL, AP	Initial release detailing the API support in version 1.2(1) of the MCU 4200 series software.
2.0	6 Feb 2006	MKL, AP, AB	Draft including the API enhancements to be released in version 1.4(1) of the MCU 4200 series software.
2.0.1	4 Apr 2006	AP, MKL	Corrections and additional information. Updated Appendix A, removed outdated Appendix B
2.0.1+	6 Apr 2006	AP, MKL	Clarified the meanings of the participantType enumerations
2.0.1+	13 Jul 2006	NM	General corrections and clarifications, and additional enhancements for version 2.1 of the API, released in version 1.5(1) of the MCU software.
2.1	31 Jul 2006	SPH,NM	Formatting changes and editing only
2.1.1	30 Oct 2006	NM	Updates to correspond to MCU version 1.5(1.14)

1 Introduction

This document contains the specification for version 2.1 of the Codian MCU 4200 Series Remote Management API, by which the Codian MCU 4200 series products are controllable. This is accomplished via messages sent using the XML-RPC protocol.

Note: The Management API option key must be loaded before an MCU 4200 series device will respond to API commands.

XML-RPC is a simple protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, whilst allowing complex data structures to be transmitted, processed and returned. XML-RPC has no platform or software dependence. XML-RPC was chosen over SOAP because of its simplicity.

The interface is stateless. Currently there is no mechanism for the Codian MCU to call back the controlling application and therefore the controlling application must poll the MCU for status, as required. A future enhancement *may* provide a mechanism for signaling MCU status changes to the controlling application.

1.1 HTTP

The Codian MCU expects to receive HTTP communication over TCP/IP connections to port 80. The HTTP messages should be “POST”s to the URL “/RPC2”.

The MCU implements HTTP/1.1 as defined by RFC 2616 [2].

1.2 XML-RPC

For the background and details of XML-RPC, please refer to the [specification](#) [1].

In this implementation, all parameters and return values are part of a <struct> and are all explicitly named. For example, the “device.query” method returns the current time value as a structure member named ‘currentTime’ rather than as a single value of type <dateTime.iso8601>.

2 Authentication

In order to manage the MCU, the controlling application must authenticate itself as a user with relevant privileges. Accordingly, each message contains a user name and password; see section 3.1 for details of the format. It is worth noting that authentication information is sent using plain text and should only be sent over a trusted network.

2.1 Message Flow

An application can create and manage conferences by sending command messages to the MCU. For each command sent (provided the message is correctly formatted according to the XML-RPC spec), the MCU responds with a message indicating success or failure. The response message may also contain data requested.

Command messages are sent in XML format. For example, the following message schedules a conference to begin at 10:45 on 18 February 2005 and last for one hour:

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: 10.2.1.100
```

```
Content-Type: text/xml
Content-length: 713

<?xml version="1.0"?>
<methodCall>
<methodName>conference.create</methodName>
<params>
<param>
<value>
<struct>
<member>
<name>authenticationUser</name>
<value>
<string>api_test</string>
</value>
</member>
<member>
<name>authenticationPassword</name>
<value>
<string>123456</string>
</value>
</member>
<member>
<name>conferenceName</name>
<value>
<string>Meeting 1</string>
</value>
</member>
<member>
<name>startTime</name>
<value>
<dateTime.iso8601>20050218T10:45:00</dateTime.iso8601>
</value>
</member>
<member>
<name>durationSeconds</name>
<value>
<int>3600</int>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

If the command was successful, the MCU sends a success response. For example, in response to a successful conference.create message, the MCU returns:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 240

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
```

```
<value>
<struct>
<member>
<name>status</name>
<value>
<string>operation successful</string>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

If the command fails, the MCU sends a fault response. For example, in response to a “conference.create” message where the conference name is not unique, the MCU returns:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 411

<?xml version="1.0"?>
<methodResponse>
<fault>
  <value>
    <struct>
      <member>
        <name>faultCode</name>
        <value>
          <int>2</int>
        </value>
      </member>
      <member>
        <name>faultString</name>
        <value>
          <string>duplicate conference name</string>
        </value>
      </member>
    </struct>
  </value>
</fault>
</methodResponse>
```

The complete list of command messages, their required and optional parameters, and the expected responses are detailed in section **Error! Reference source not found.** The possible fault codes are listed in section 7. Appendix B contains examples of some messages and their corresponding responses.

3 Messages

3.1 Common Message Elements

3.1.1 Authentication

All messages must contain a user name and password as follows:

Parameter	Type	Comments
authenticationUser	string	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive
authenticationPassword	string	The corresponding user's password. This parameter is ignored if the user has no password set - note that this differs from the web interface where a blank password must be blank.

3.1.2 Participant identification parameters

The following parameters appear in the majority of messages, and identify a specific participant on which operations are to be performed.

Parameter	Type	Description
participantName	string	This is an "internal" name, and therefore is not necessarily related to any name configured on an endpoint. Within the scope of a particular conference or auto attendant, the combination of "participantType", "participantProtocol" and "participantName" is always unique
participantProtocol	string	Used in conjunction with "participantName" to uniquely identify a participant within a connection. Typically, these parameters should be treated as opaque values, but the current possibilities are: for "participantProtocol": h323 – an endpoint using the H.323 protocol vnc – a VNC connection (e.g. remote desktop) sip – an endpoint using the SIP protocol. for "participantType": ad_hoc – this participant called into the MCU or was dialed out via the web GUI and is not in the MCU's endpoint list by_address – fully-specified participant added through the API by_name – MCU-configured endpoint irrespective of whether the endpoint dialed in or the MCU dialed out API-created participants (i.e. those originated by "participant.add") will be of type "by_address"
participantType	string	

Parameter	Type	Description
conferenceName	string	Unique conference name – the conference name space is shared between API-generated conferences and all other ad hoc and scheduled MCU conferences
autoAttendantUniqueId	string	If the participant in question is connected to an auto attendant rather than a conference, this field contains a unique identifier for that auto attendant

When modifying or querying parameters for a specific endpoint, “participantName”, “participantProtocol” and “participantType” parameters are supplied, along with either a “conferenceName” or an “autoAttendantUniqueId”. However, as the API can only create H.323 participants of type “by_address”, if the protocol and type fields are absent, then they are assumed to be “h323” and “by_address”. The only safe way to find these values is to use the fields returned from participant.enumerate.

3.2 device.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system’s current time
restartTime	dateTime.iso8601	The date and time at which the system was last restarted
serial	string	The serial number of the MCU
softwareVersion	string	The software version of the running software
buildVersion	string	The build version of the running software
activatedFeatures	array of structs	Currently, only contains a string "feature" with a short description of the feature
totalVideoPorts	int	The total number of video ports on the MCU
totalAudioOnlyPorts	int	The total number of additional audio-only ports on the MCU
portReservationMode	string	"enabled" or "disabled", depending on the Media Port Reservation configuration setting
maxVideoResolution	string	One of "cif" or "4cif"
model	string	The model of this MCU
apiVersion	string	The version number of the API implemented by this MCU

3.3 device.network.query

This call takes no parameters. The response returns the following:

Parameter	Type	Comments
portA	struct (see below)	Contains the configuration and status for port A
portB	struct (see below)	Contains the configuration and status for port B

The format for the two structures above is:

Field	Type	Comments	
enabled	boolean	true if the port is enabled, otherwise false	
hostName (optional)	string	The host name of this port	
dhcp (optional)	boolean	true if configured by DHCP, otherwise false	
ipAddress (optional)	string	a.b.c.d format	
subnetMask (optional)	string	a.b.c.d format	
defaultGateway (optional)	string	a.b.c.d format	
domainName (optional)	string	The domain name of this port	
nameServer (optional)	string	a.b.c.d format	
nameServerSecondary (optional)	string	a.b.c.d format	
linkStatus	boolean	true if the link is up, false if the link is down	
speed	int	one of 10, 100 or 1000, in Mbps	
fullDuplex	boolean	true if full duplex enabled, false if half	
macAddress	string	a 12 character string, no separators	
packetsSent	int	Stats from the web interface. It is worth noting that all these values are 32 bit signed integers, and thus may wrap.	
packetsReceived	int		
multicastPacketsSent	int		
multicastPacketsReceived	int		
bytesSent	int		
bytesReceived	int		
queueDrops	int		
collisions	int		
transmitErrors	int		
receiveErrors	int		
bytesSent64	string		64 bit versions of the above stats, using a string rather than an int.
bytesReceived64	string		

All fields above marked optional will only be returned if the interface has been enabled and has been configured.

3.4 device.health.query

Returns the current status of the MCU, such as health monitors and CPU load.

Response	Type	Comments
cpuLoad	int	The CPU load, as a percentage
mediaLoad	int	Loads for the media processors (total, and split

Response	Type	Comments
audioLoad	int	between audio and video) as percentage values
videoLoad	int	
fanStatus	string	One of "ok", "outOfSpec" and "critical"
fanStatusWorst	string	
temperatureStatus	string	
temperatureStatusWorst	string	
rtcBatteryStatus	string	
rtcBatteryStatusWorst	string	
voltagesStatus	string	
voltagesStatusWorst	string	
operationalStatus	string	"active", "shuttingDown" or "shutDown"

3.5 device.restartlog.query

Used to return the restart log (also known as the system log on the MCU web interface).

Response	Type	Comments
log	array	Contains the restart log in structures as described below

The "log" array consists of structures which contain the following fields.

Field	Type	Comments
time	dateTime.iso8601	The time of the last reboot
reason	string	The reason for the reboot ("unknown", "User requested shutdown" or "User requested upgrade")

3.6 gatekeeper.query

Retrieves the gatekeeper settings and current status. Takes no parameters.

Response	Type	Comments
gatekeeperUsage	string	One of "disabled", "enabled" or "required"
<i>The following parameters are all optional and will only be present if gatekeeperUsage is not "disabled"</i>		
address	string	The address of the gatekeeper
dnsStatus	string	The status of the DNS resolution: one of "inProgress", "resolved" or "failed"
ip	string	If the dnsStatus is resolved, contains the IP address of the gatekeeper
activeRegistrations	int	The number of active registration

Response	Type	Comments
pendingRegistrations	int	The number of registrations in progress.
registrationPrefix	string	The registration prefix used by the MCU
h323ID	string	The h323 id used by the MCU
mcuServicePrefix	string	The service prefix used by the MCU
h323IDStatus	string	The current status of the ID/service prefix registration process. One of "idle", "registering", "registered", "deregistering", "pendingReregistration", "waitingRetry", "noID" or "idTooLong"
mcuServicePrefixStatus	string	

3.7 conference.create

Parameter	Type	Comments
conferenceName	string (<32 chars)	Name of the conference to be created. The conference name must be unique
numericId (optional)	string (<32 chars)	Numeric identifier of the conference
conferenceId (deprecated)	string (< 32 chars)	Deprecated alternative for "numericId"
registerWithGatekeeper (optional)	boolean	Register the conference's "numericId" with the gatekeeper
startTime (optional)	dateTime.iso8601	If you do not specify a time, the conference starts immediately
durationSeconds (optional)	int	The length of each repeating conference instance, in seconds. If this parameter is absent, or set to "0", the conference is permanent
endTime (optional) (deprecated)	dateTime.iso8601	If you do not specify an end time, then the conference will be permanent (until it is explicitly deleted). This parameter is deprecated and present for backward compatibility reasons only. Application code should use "durationSeconds" instead
pin (optional)	string (< 32 chars)	If present, this is the string of numeric digits that people need to enter to join the conference
description(optional)	string (< 32 chars)	
multicastStreamingEnabled(optional)	boolean	
unicastStreamingEnabled(optional)	boolean	
h239Enabled(optional)	boolean	

Parameter	Type	Comments
maximumAudioPorts (optional)	int	These fields set the limit on number of audio (audio only) and video (video + audio) ports for the conference. The “reserved” values are for port reservation mode, whereas the “maximum” figures apply to non-reserved mode (and will be absent if no limits have been configured)
maximumVideoPorts (optional)	int	
reservedAudioPorts (optional)	int	
reservedVideoPorts (optional)	int	
repetition(optional)	string	One of: none daily weekly everyTwoWeeks monthly
weekDay(optional)	string	Must be present if repetition is “monthly”. One of: monday tuesday wednesday thursday friday saturday sunday Note that if repetition is "not weekly" or "everyTwoWeeks", the "weekDays" parameter should be used.
whichWeek(optional)	string	Must be present if repetition is “monthly”. One of: first = first X of the month (where X is the day specified by weekday) second third fourth last = last X of the month

Parameter	Type	Comments
weekDays(optional)	string	Must be present if repetition is “weekly” or “everyTwoWeeks”. A comma separated string of weekdays from the following list: monday tuesday wednesday thursday friday saturday sunday e.g. “monday,Wednesday,friday”
terminationType (optional)	string	One of: noTermination afterNRepeats endOnGivenDate
terminationDate (optional)	dateTime.iso8601	If terminationType is endOnGivenDate, this is the day that the conference repetition will end on.
numberOfRepeats (optional)	int	If terminationType is afterNRepeats, this is the number of repeats to end after

Conferences created through the management API will appear in the list of conferences accessible via the Web interface, and vice versa.

3.8 conference.modify

Parameter	Type	Comments
conferenceName	string (< 32 chars)	Name of the conference to modify
newConferenceName (optional)	string (< 32 chars)	If present, the conference will be renamed to specified value
oldConferenceName (deprecated)	string (<32 chars)	Deprecated conference renaming scheme – new code should use conferenceName and newConferenceName as above
conferenceName (deprecated)	string (<32 chars)	
numericId	string (< 32 chars)	Optional fields as per “conference.create” described above
conferenceId (deprecated)	string	
registerWithGatekeeper	boolean	
startTime	dateTime.iso8601	
durationSeconds	int	

Parameter	Type	Comments
endTime (deprecated)	dateTime.iso8601	
pin	string	
description	string	
multicastStreamingEnabled	boolean	
unicastStreamingEnabled	boolean	
h239Enabled	boolean	
reservedVideoPorts	int	
reservedAudioPorts	int	
maximumVideoPorts	int	
maximumAudioPorts	int	
repetition	string	
weekDay	string	
whichWeek	string	
weekDays	string	
terminationType	string	
terminationDate	dateTime.iso8601	
numberOfRepeats	int	

Conferences created through the management API will appear in the list of conferences accessible via the web interface. Therefore, the API can be used to modify conferences scheduled via the web interface, and vice versa.

3.9 conference.destroy

Parameter	Type	Comments
conferenceName	string	Name of the conference to be destroyed

A conference can be destroyed at any time; that is, before the conference has begun, during the conference or after the conference has ended. Destroyed conferences are removed entirely from the system; this includes all future repetitions of the conference.

3.10 conference.end

Parameter	Type	Comments
conferenceName	string	Name of the conference to be ended

A conference remains in the list of conferences even after the conference has ended — until `conference.destroy` is called. In particular, this can be used to end an instance of a conference without deleting all future repetitions.

3.11 conference.enumerate

See participant.enumerate for notes upon how to use enumerateID values.

Parameter	Type	Comments
enumerateID (optional)	string	The value returned by the last enumeration call. If it is omitted, a new enumeration is started.

This method returns:

Response	Type	Comments
enumerateID (optional)	string	The value which should be used in the next call to get the next set of data. If this is omitted, no further data is available from the MCU
conferences	array of structs	See below for details

The array “conferences” contains structs with the following fields

Field	Type	Comments
conferenceName	string	
conferenceType	string	One of: scheduled ad_hoc
conferenceActive	boolean	Indicates whether conference is currently active
description	string	Extra user-specified information about the conference
pin	string	The security PIN
numericId	string	
registerWithGatekeeper	boolean	
multicastStreamingEnabled	boolean	
unicastStreamingEnabled	boolean	
h239Enabled	boolean	
maximumAudioPorts	int	These fields set the limit on number of audio (audio only) and video (video + audio) ports for the conference. The “reserved” values are for port reservation mode, whereas the “maximum” figures apply to non-reserved mode (and will be absent if no limits have been configured)
maximumVideoPorts	int	
reservedAudioPorts	int	
reservedVideoPorts	int	

Field	Type	Comments
customLayoutEnabled	boolean	True if a custom layout has been enabled for this conference
customLayout (optional)	int	The index (from appendix A) of the custom layout. This is only present if the custom layout is enabled
The following timing fields will be present for scheduled conferences only		
startTime	dateTime.iso8601	The time at which the conference started at or will start at
durationSeconds	int	How long each repeating instance of the conference should last for. If absent, the conference is permanent
repetition(optional)	string	One of: none daily weekly everyTwoWeeks monthly
weekDay(optional)	string	Must be present if repetition is monthly. One of: monday tuesday wednesday thursday friday saturday Sunday
whichWeek (optional)	string	Must be present if repetition is monthly. One of: first = first X of the month (where X is the day specified by weekday) second third fourth last = last X of the month

Field	Type	Comments
weekDays (optional)	string	A comma separated string of weekdays from the following list: monday tuesday wednesday thursday friday saturday sunday e.g. "monday,wednesday,friday" This field should be provided when repetition is weekly or everyTwoWeeks
terminationType(optional)	string	One of: noTermination afterNRepeats endOnGivenDate
terminationDate(optional)	dateTime.iso8601	If terminationType is endOnGivenDay, this is the day that the conference repetition will end on.
The following timing values will be present for active conferences only		
activeStartTime	dateTime.iso8601	If the conference is currently active, these fields show the time span of the current activation. If the conference is permanent then "activeEndTime" will be absent
activeEndTime	dateTime.iso8601	
activeConferenceId	string	A unique ID for the active instance of this conference; this conference will have this ID even if, for example, the conference is renamed while active, but each scheduled instance of this conference will have a different activeConferenceId

3.12 conference.streaming.query

This returns some details on the current state of streaming viewers for a conference.

Parameter	Type	Comments
conferenceName	string	Name of the conference from which streaming information is required

This will return a structure with the following fields:

Response	Type	Comments
unicastViewers	int	The number of unicast streaming viewers
multicastViewers	int	The number of multicast streaming viewers
audioStreams (optional)	array	An array of stream structs (defined below). These are only present if there are any streams of either type currently in use
videoStreams (optional)	array	
audioRTCPReceiverReports	int	The number of RTCP receiver reports for the audio streams seen by the MCU
audioRTCPSenderReports	int	The number of RTCP sender reports for the audio streams seen by the MCU
audioRTCPOther	int	The number of other RTCP packets seen for the audio streams
audioRTCPPacketsSent	int	The number of RTCP packets send by the MCU
videoRTCPReceiverReports	int	As for the audio equivalents
videoRTCPSenderReports	int	
videoRTCPOther	int	
videoRTCPPacketsSent	int	

The stream structures used in the audioStreams/videoStreams responses above have the following fields:

Field	Type	Comments
codec	string	The codec in use, or "other" for undefined codecs
count	int	The number of users of this codec
bitRate (optional)	int	The bit rate of this stream in bits/second. This is only present for video streams with a defined codec
width (optional)	int	The maximum width and height of this stream. Only present for defined video streams.
height (optional)	int	

This method will return a fault code of "no such conference" if there is no *active* conference with the given name, regardless of the presence a configured but inactive conference of that name.

3.13 conference.paneplacement.query

Queries the current pane placement configuration.

Parameter	Type	Comments
conferenceName	string	The name of the conference to be queried.

This returns a struct containing the following response fields:

Response	Type	Comments
enabled	boolean	true if pane placement is enabled and in use, false otherwise.
panes (optional)	array of structs	This is only present if enabled above is true. The struct definition is as below.

The panes array contains structures with the following format:

Field	Type	Comments
type	string	Any one of: default - the default behaviour blank - a blank window loudest - the current loudest speaker h239 - the H.239 content channel participant - a participant as identified below.
index	int	The index of this pane.
participantType (optional)	string	Participant identification, only present if this pane contains a specific participant.
participantProtocol (optional)	string	
participantName (optional)	string	

3.14 conference.paneplacement.modify

Modifies the pane placement configuration of a particular conference.

Parameter	Type	Comments
conferenceName	string	Name of the conference to be queried
enabled (optional)	boolean	true to enable pane placement, false to disable
panes (optional)	array of structs	see below for format of the structures

The panes array contains a structures which define a specific pane and its contents. If a pane index is not present in the array, then that pane will remain unchanged. Participant identification is as returned in participant.enumerate.

Field	Type	Comments
index	int	The index of the pane to be changed.
type	string	Any one of: default - the default behaviour blank - a blank window loudest - the current loudest speaker h239 - the H.239 content channel participant - a participant as described in the three optional fields.
participantType (optional)	string	Participant identification, only required if type is participant, these identify a specific participant.
participantProtocol (optional)	string	
participantName (optional)	string	

Since not all panes are guaranteed to be changed, this call returns the following structure:

Response	Type	Comments
panesModified	int	The number of panes successfully modified. This will be the number of elements in the panes array on complete success, and zero if there is no panes array.

3.15 participant.add

Parameter	Type	Comments
conferenceName	string	The name of the conference to which to add the participant
participantName	string (<32 chars)	The name of the participant to be added. This must be a unique value, i.e. not the same as any existing participant
participantProtocol (optional)	string	If present, must be "h323" – this is the only protocol that the API can currently use
participantType (optional)	string	If present, must be "by_address"
address	string (< 32 chars)	The participant's E.164 directory number, hostname or IP address
gatewayAddress (optional)	string (< 32 chars)	IP address or hostname of an H.323 gateway
deferConnection (optional)	boolean	If true, means don't call out to this participant immediately, but wait for a "participant.connect" command
All of the following parameters are optional, and control the conferencing behaviour of the MCU		

Parameter	Type	Comments
		with respect to the endpoint in question, for example the maximum resolution of the video streams used, or whether the participant is able to control their conference view layout.
maxBitRateToMCU	int	The maximum bit rate to the MCU specified as kBit/s
maxBitRateFromMCU	int	The maximum bit rate from the MCU specified as kBit/s
motionSharpnessTradeoff	string	One of "default" (to use the global default setting), "preferMotion", "preferSharpness" and "balanced"
displayNameOverrideStatus	Boolean	If true, means use the specified "displayNameOverrideValue" text as the participant's display name during the conference
displayNameOverrideValue	string (< 32 chars)	Value to use as the participant's display name (if "displayNameOverrideStatus" set to true)
cpLayout	string	This sets the initial conference view layout for video sent to this participant. Refer to Appendix A for the full list of available layouts
layoutControlEnabled	boolean	Controls whether this participant is able to change the conference view layout that they see; 1 (true) means that the participant can change the layout using FECC or DTMF, 0 (false) means that they cannot
audioRxMuted	boolean	1 (true) means that audio from this participant will not be heard by other conference participants
audioRxGainMode	string	One of: none – no extra gain applied automatic – automatic gain control applied fixed – fixed number of dBs of gain applied
audioRxGainMillidB	int	If audio gain mode "fixed", this is the number of decibels of gain applied, multiplied by 1000, and can be a negative value
videoRxMuted	boolean	true means that video from this participant will not be seen by other conference participants
videoTxWidescreen	boolean	If true, means that video sent to this participant will be in a form suitable for a widescreen (16:9) display
videoTxMaxResolution	string	One of: cif 4cif max
videoRx+MaxResolution	string	Same as above

All participants in a conference must have a "participantName" that is unique to the conference but it need not be unique across all conferences.

Participants can be added before or during a conference. A participant which is added at any time via the API will be added to the configured list of participants, and thus will be called at the start of the conference by the MCU for any conference which has any sort of repetition; to avoid this, a participant must be removed directly using `participant.remove`.

Note: If a “participantName” matches the name of an endpoint in the list of configured endpoints (via the web interface, go to **Endpoints**) the two are treated as unrelated. This is because web interface named, configured, endpoints have the “participantType” value “by_name”, whereas API participants are of type “by_address”.

3.16 *participant.remove*

This call removes a participant from the database of configured participants, and also removes this participant from any conferences. It will also remove all records of this participant's presence in a conference.

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

3.17 *participant.modify*

Depending on the `operationScope` parameter below, this call modifies the configuration of a participant, or the active state of a participant in a conference.

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
operationScope	string	One of "activeState" or "configuredState". This parameter specifies the scope of the changes to be made, be they to the configured state of an endpoint, or to the active state of a participant in a conference
address	string	All these parameters are optional. They should not be present if operationScope is "activeState". These parameters override the configured values
gatewayAddress	string	
deferConnection	boolean	
maxBitRateToMCU	int	
maxBitRateFromMCU	int	
motionSharpnessTradeoff	string	One of "default" (if set to the global default

Parameter	Type	Comments
(optional)		setting), "preferMotion", "preferSharpness" and "balanced". Optional, and valid for both configuredState and activeState scopes
displayNameOverrideStatus	boolean	All of these parameters are optional, and override / change the values provided in the "participant.add" call. Depending on the value of the operationScope parameter, these either, if "configuredState", change the stored configuration of a participant, or, if "activeState", change the active participant state, resulting in real-time changes to that participant
displayNameOverrideValue	string	
cpLayout	string	
layoutControlEnabled	boolean	
audioRxMuted	boolean	
audioRxGainMode	string	
audioRxGainMillidB	int	
videoRxMuted	boolean	
videoTxWidescreen	boolean	
important (optional)	boolean	This setting should not be present if operationScope is "configuredState". Specifies whether this participant is important
borderWidth (optional)	int	This setting should not be present if operationScope is "configuredState". 0 (no border), or 1, 2, or 3 for +1/+2/+3

If there is no operationScope parameter, the MCU will attempt to change both active and configured states. This is deprecated behaviour, and should not be relied upon.

3.18 participant.connect

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

Used primarily for API-generated participants added with "deferConnection" set to true; this command causes the MCU to call out to such participants.

3.19 participant.disconnect

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

This call causes the MCU to tear down its connection to the specified participant, if such a connection exists. This is different from “participant.remove” above because,

- ▶ in the case of configured participants, it does not remove the configuration (thus allowing later re-connection with “participant.connect”), and
- ▶ in the case of ad hoc participants, it does not remove the record of the previous connection

3.20 participant.move

This method is used to move a participant from one conference to another.

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
newConferenceName	string	The name of the conference to move the participant to

This will only move an active participant. Even if this participant is preconfigured, the configuration is unchanged.

A fault code of "no such participant" is returned if the participant is not moved or does not exist.

3.21 participant.enumerate

This method is used to return data about participants in conferences on the MCU. Several calls may be required to receive data about all participants; see the notes on enumerateID below.

Parameter	Type	Comments
enumerateID(optional)	string	The value returned by the last enumeration call. If this parameter is omitted, a new enumeration is started.
operationScope	array of strings	This should contain none, either or both of "currentState" or "configuredState". If "currentState" is present, the active configuration of each participant is returned by the MCU in the currentState structure. If "configuredState" is present, the stored configuration is returned in the configuredState structure

Notes on enumerateID: The use of this parameter is as follows:

1. The client computer sends a participant.enumerate call with the desired operationScope and no enumerateID parameter.
2. The MCU returns with an array containing the requested data, and sometimes a new enumerateID.
3. If there is an enumerateID, the client should call participant.enumerate again, with the desired operationScope and the enumerateID returned by the MCU from the previous call as parameters. This should be repeated while the MCU continues to provide new enumerateID values in responses.
4. After all data is returned, the MCU will reply with all remaining participants, but no enumerateID.

This method should only be called using enumerateID values as provided by the MCU.

This method returns:

Response	Type	Comments
enumerateID (optional)	string	The value which should be used in the next call to get the next set of data. If this is not present, there is no further data available from the MCU
participants	array of structs	See below for contents

and an array called "participants" of structs which contain:

Field	Type	Comments
participantName	string	Participant identification as described above
participantProtocol	string	
participantType	string	
conferenceName	string	
autoAttendantUniqueId	string	
currentState (optional)	struct	The current state of the participant as used by the MCU. Details of this struct are below. This is only present if requested in the operationScope
configuredState (optional)	struct	The stored configuration of the participant, if any. Details of this struct are below. This is only present if requested in the operationScope

If the endpoint is not configured, the configuredState structure is empty; otherwise the configuredState structure contains the following entries:

Field	Type	Comments
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known, or if the participant is configured via the API (which requires the address to be specified when added)
gatewayAddress	string	
deferConnection	boolean	
displayNameOverrideStatus	boolean	Indicates whether the displayName value is the result of being overridden
maxBitRateToMCU	int	As for "participant.add"; in kbps
maxBitRateFromMCU	int	
motionSharpnessTradeoff	string	One of "default" (if set to the global default setting), "preferMotion", "preferSharpness" and "balanced"
audioRxMuted	boolean	
audioRxGainMode	string	One of "fixed", "none" or "automatic"
audioRxGainMillidB	int	Only returned if audioRxGainMode is "fixed"
videoRxMuted	boolean	
videoTxWidescreen	boolean	
layoutControlEnabled	boolean	
cpLayout	string	The configured layout behavior for this participant - see appendix A

The currentState structure contains the following responses, if present.

Field	Type	Comments
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known
gatewayAddress	string	
ipAddress	string	The IP address to which the MCU is connected for this endpoint - this will usually be the endpoint itself, but may be a gatekeeper or gateway. Only present for active participants.
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only available after the participant has connected
displayNameOverrideStatus	boolean	Indicates whether the displayName value is the result of being overridden

Field	Type	Comments
maxBitRateToMCU	int	As for "participant.add"; in kbps
maxBitRateFromMCU	int	
motionSharpnessTradeoff	string	One of "default" (if set to the global default setting), "preferMotion", "preferSharpness" and "balanced"
callState	string	One of: 'dormant' 'alerting' 'connected' 'disconnected'
connectTime	dateTime.iso8601	Only returned after the participant is connected. This value is always present if the call state is "connected". It may or may not be defined for participants in the "disconnected" state, depending on whether they were ever connected
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	string	Only returned after the participant has disconnected; this contains one of the disconnect reason strings given in section 8
connectPending	boolean	true if sending a "participant.connect" command for this participant will cause either the initial connection to that endpoint (in the event that it was configured with "deferConnection" set) or a re-connection to that endpoint (in the event that it has disconnected)
audioRxCodec	string	
audioRxLost	int	
audioRxReceived	int	
audioTxCodec	string	
audioTxReportedLost	int	
audioTxSent	int	
audioRxMuted	boolean	
audioRxGainMode	string	One of "fixed", "none" or "automatic"
audioRxGainMillidB	int	Only present if the audioRxGainMode is fixed
videoRxCodec	string	
videoRxLost	int	
videoRxReceived	int	
videoTxCodec	string	
videoTxReportedLost	int	

Field	Type	Comments
videoTxSent	int	
videoRxMuted	boolean	
videoTxWidescreen	boolean	
important	boolean	
activeSpeaker	boolean	true if this participant is currently the active speaker in the conference
layoutControlEnabled	boolean	
activeConferenceId	string	The active conference ID of the current conference - see conference.enumerate for details of this field. This field is only present if the participant is currently in an active conference
currentLayout	int	Actual layout in use for the video stream being sent by the MCU to this participant. This parameter will not be present if the participant is in an auto attendant rather than a conference, or if the MCU is not currently transmitting video to the participant in question. The values for this are those described in appendix A
layoutSource	string	This will be one of family<x>, conferenceCustom, participantCustom, and describes the reason for the current layout. This parameter is only present if the currentLayout parameter is also present, i.e. if the participant is in an active conference
callDirection	string	One of: incoming outgoing
previewURL	string	The location of the preview image; this is not a complete URL, and requires a prefix of http://<hostname> (where hostname is the hostname of this MCU) before it is used

Note: This participant information is returned for all participants added to the conference using the participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

Deprecated behavior:

If there is no operationScope variable, the MCU will revert to previous behavior, and return a participant structure with the following members:

Response	Type	Comments
participantName	string	Participant identification as described above
participantProtocol	string	

Response	Type	Comments
participantType	string	
conferenceName	string	
autoAttendantUniqueId	string	
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known, or if the participant is configured via the API (which requires the address to be specified when added)
gatewayAddress	string	
deferConnection	boolean	
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only available after the participant has connected
displayNameOverrideStatus	boolean	Indicates whether the displayName value is the result of being overridden
maxBitRateToMCU	int	As for “participant.add”; in kbps
maxBitRateFromMCU	int	
callState	string	One of: ‘dormant’ ‘alerting’ ‘connected’ ‘disconnected’
connectTime	dateTime.iso8601	Only returned after the participant is connected. This value is always present if the call state is “connected”; it may or may not be defined for participants in the “disconnected” state, depending on whether they were ever connected
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	string	Only returned after the participant has disconnected, one of the disconnect reason values given in table 5
connectPending	boolean	true if sending a “participant.connect” command for this participant. This parameter will cause either the initial connection to that endpoint (in the event that it was configured with “deferConnection” set) or a re-connection to that endpoint (in the event that it has disconnected)
audioRxCodec	string	
audioRxLost	int	
audioRxReceived	int	

Response	Type	Comments
audioTxCodec	string	
audioTxReportedLost	int	
audioTxSent	int	
audioRxMuted	boolean	
audioRxGainMode	string	
audioRxGainMillidB	int	
videoRxCodec	string	
videoRxLost	int	
videoRxReceived	int	
videoTxCodec	string	
videoTxReportedLost	int	
videoTxSent	int	
videoRxMuted	boolean	
videoTxWidescreen	boolean	
important	boolean	
activeSpeaker	boolean	true if this participant is currently the active speaker in the conference
layoutControlEnabled	boolean	
cpLayout	string	The configured layout behavior for this participant - see appendix A. This parameter will be present only for participants configured via the API
currentLayout	int	Actual layout in use for the video stream being sent by the MCU to this participant. This parameter will not be present if the participant is in an auto attendant rather than a conference, nor if the MCU is not currently transmitting video to the participant in question. The values for this are those described in appendix A
callDirection	string	One of: incoming outgoing

3.22 participant.fecc

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueid	string	
participantName	string	
participantProtocol	string	

Parameter	Type	Comments
participantType	string	
direction	string	One of: up down left right zoomIn zoomOut

3.23 participant.message

Puts a message on the display of a given participant.

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
message	string	The string to send to the participant.
durationSeconds (optional)	int	The length of time, in seconds, to display the message. This defaults to 30 seconds.

3.24 participant.diagnostics

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

The method response returns the following:

Response	Type	Comments
videoTxFrameRate	int	
videoRxFrameRate	int	
videoRxFramesReceived	int	
videoTxChannelBitRate	int	
videoTxSelectedBitRate	int	

Response	Type	Comments
videoTxActualBitRate	int	
videoTxLimitReason	string	One of: notLimited viewedSize quality aggregateBandwidth flowControl endpointLimitation
videoRxChannelBitRate	int	
videoRxSelectedBitRate	int	
videoRxActualBitRate	int	
videoRxLimitReason	string	One of: notLimited viewedSize quality aggregateBandwidth flowControl endpointLimitation
videoTxWidth	int	
videoTxHeight	int	
videoTxInterlaced	boolean	
videoRxWidth	int	
videoRxHeight	Int	
videoRxInterlaced	boolean	

3.25 autoAttendant.enumerate

See participant.enumerate for notes upon how to use enumerateID values.

Parameter	Type	Comments
enumerateID(optional)	string	The value returned by the last enumeration call, if omitted a new enumeration is started

This method returns:

Response	Type	Comments
enumerateID(optional)	string	The value which should be used in the next call to get the next set of data. If this is omitted, then no further data is available from the MCU
autoAttendants	array of structs	See below for contents

and an array called “autoAttendants” of structs which contain:

Response	Type	Comments
autoAttendantUniqueID	string	
startTime	dateTime.iso8601	The time at which the auto attendant was created

3.26 autoAttendant.destroy

Parameter	Type	Comments
autoAttendantUniqueID	string	Identifier for the auto attendant to be destroyed

4 Deprecated messages

These messages were supported in version 1.0 of the MCU 4200 Series Management API, but have since been superseded in API Version 2.0.

4.1 *system.query*

This method is deprecated in favour of “*device.query*” and “*conference.enumerate*”.

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system’s current time
restartTime	dateTime.iso8601	The date and time at which the system was last restarted

The method also returns a ‘conferences’ <array> of <struct> for each conference, where each <struct> contains the following parameters:

Parameter	Type	Comments
conferenceName	string	The name of the conference
numJoined	int	The number of participants that have ever joined the conference
numLeft	int	The number of participants that have ever left the conference. The difference between numJoined and numLeft gives the number of current participants

Note that until *conference.destroy* is called for a particular conference, the conference will remain in the list of conferences even after the conference has ended.

4.2 *conference.query*

This method is deprecated in favour of “*conference.enumerate*” and “*participant.enumerate*”.

Parameter	Type	Comments
conferenceName	string	The name of the conference of interest

The method response returns the following:

Parameter	Type	Comments
startTime	dateTime.iso8601	The time at which the conference started at or will start at
endTime	dateTime.iso8601	The time at which the conference will end. If the conference is permanent then this parameter is absent
pin	string	The PIN

The method also returns a 'participants' <array> of <struct> for each conference, where each <struct> contains the following parameters:

Response	Type	Comments
participantName	string	The participant name supplied in the participant.add message
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only returned when the participant is connected
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known
callState	string	One of: 'dormant' 'alerting' 'connected' 'disconnected'
connectTime	dateTime.iso8601	Only returned after the conference has begun
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	string	Only returned after the participant has disconnected

Note: This participant information is returned for all participants added to the conference using the participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

4.3 conference.participant.modify

This method has been deprecated in favour of participant.modify.

See participant.modify for details of this function.

4.4 conference.participant.remove

This method has been deprecated in favour of participant.remove.

See participant.remove for details of this function.

4.5 conference.participant.add

This method has been deprecated in favour of participant.add.

See participant.add for details of this function.

5 Related information sources

5.1 system.xml

While not strictly part of the XMLRPC API, some information can be retrieved from the system.xml file. This can be downloaded via HTTP as the file system.xml in the root of the MCU, i.e.

<http://TestMCU/system.xml>

An example is:

```
<system>
  <manufacturer>Codian</manufacturer>
  <model>MCU 4210</model>
  <serial>MRV1001SM0004B8</serial>
  <softwareVersion>1.4(1.2)</softwareVersion>
  <buildVersion>B.6.6(1.2)</buildVersion>
  <hostName>TestMCU</hostName>
  <totalVideoPorts> 20</totalVideoPorts>
  <totalAudioOnlyPorts>20</totalAudioOnlyPorts>
  <portReservationMode>disabled</portReservationMode>
  <maxVideoResolution>cif</maxVideoResolution>
</system>
```

The meaning of the fields is:

Field	Comments
manufacturer	The manufacturer of this MCU, i.e. Codian
model	The model of this particular MCU, e.g. MCU 4210
serial	The serial number of this MCU
softwareVersion	The software version currently running
buildVersion	The build version of the software currently running
hostName	The host name of the system
totalVideoPorts	The total number of video ports on the MCU
totalAudioOnlyPorts (optional)	The total number of additional audio only ports on the MCU. Only present if there are any audio-only ports
portReservationMode	"enabled" or "disabled", depending on the Media Port Reservation configuration setting
maxVideoResolution	The maximum video resolution for the MCU; either "cif" or "max" if larger video is enabled

6 Required user privileges

The following table summarises which users are permitted to perform which remote management operations.

Method	Valid users
device.query	Any user with administrator privileges
device.network.query	Any user with administrator privileges
device.health.query	Any user with administrator privileges
gatekeeper.query	Any user with administrator privileges
conference.enumerate	Any user with administrator privileges
participant.enumerate	Any user with administrator privileges
conference.create	Any user with conference creation abilities
conference.destroy	The owner of the conference, or a user with “conference creation and full control” or “administrator” privilege
conference.modify	
conference.end	
conference.streaming.query	The owner of the conference, or a user with "conference creation and limited control" or higher privileges
participant.add	Any user with “administrator” privilege, the owner of the conference, or a user with “conference creation and full control” rights
autoAttendant.enumerate	Any user with administrator privileges
autoAttendant.destroy	
participant.remove	If the participant is connected to an auto attendant, administrator privileges are required. If the participant is in a conference, the user must be the owner of the conference or a user with “conference creation and full control” or “administrator” privilege
participant.modify	
participant.move	
participant.diagnostics	
participant.fecc	
participant.message	
participant.connect	
participant.disconnect	
participant.disconnect	
Deprecated methods	
system.query	Any user with administrator privileges
conference.query	Any user with privilege “conference detail” or higher
conference.participant.modify	As above for “participant.modify”
conference.participant.remove	As above for “participant.remove”
conference.participant.add	As above for “participant.add”

7 Fault Codes

Fault Code	Description
1	Method not supported
2	Duplicate conference name
3	Duplicate participant name
4	No such conference or auto attendant
5	No such participant
6	Too many conferences
7	Too many participants
8	No conference name or auto attendant id supplied
9	No participant name supplied
10	No participant address supplied
11	Invalid start time specified
12	Invalid end time specified
13	Invalid PIN specified
14	Unauthorized – invalid user id / password
15	Insufficient privileges – specified user id / password not valid for attempted operation
16	Invalid “enumerateID” value passed in “...enumerate” method invocation
17	Port reservation failure – “reservedAudioPorts” or “reservedVideoPorts” value too high
18	Duplicate numeric ID

8 Participant disconnect reasons

These are the possible values of the “disconnectReason” field in participant information responses:

Value	Description
unspecified	
unspecifiedError	
remoteTeardown	
localTeardown	
noAnswer	
moved	
rejected	
rejectedImmediately	
busy	
timeout	
gatekeeperError	
networkError	
protocolError	
dnsFailed	
destinationUnreachable	
gatekeeperEnded	
videoPortAllocationExceeded	
portAllocationExceeded	
disconnectAll	
incompatibleVncVersion	
failedToConnectToServer	
authenticationFailed	
serviceUnavailable	
capabilityNegotiationError	
messageQueueOverflow	
gatekeeperRequiredButAbsent	
noGatekeeperForDN	
localGatekeeperRefused	
remoteGatekeeperRefused	
remoteGatekeeperUnreachable	
remoteGatewayResources	

Value	Description
gatekeeperForced	
h225SocketError	
h225ProtocolError	
h225DecodeError	
h245SocketError	
h245ProtocolError	
h245DecodeError	
q931DecodeError	
q931ProtocolError	

9 References

[1]	XML-RPC, http://www.xmlrpc.com/
[2]	RFC 2616, http://www.faqs.org/rfcs/rfc2616.html

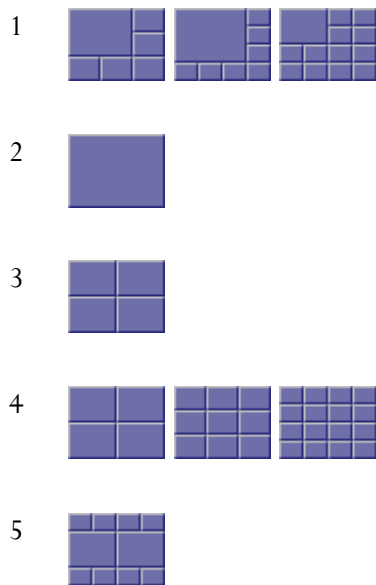
Appendix A - Conference Layouts

The `participant.add` and `participant.modify` methods allow a particular layout to be specified for video sent to that participant via the “`cpLayout`” parameter.

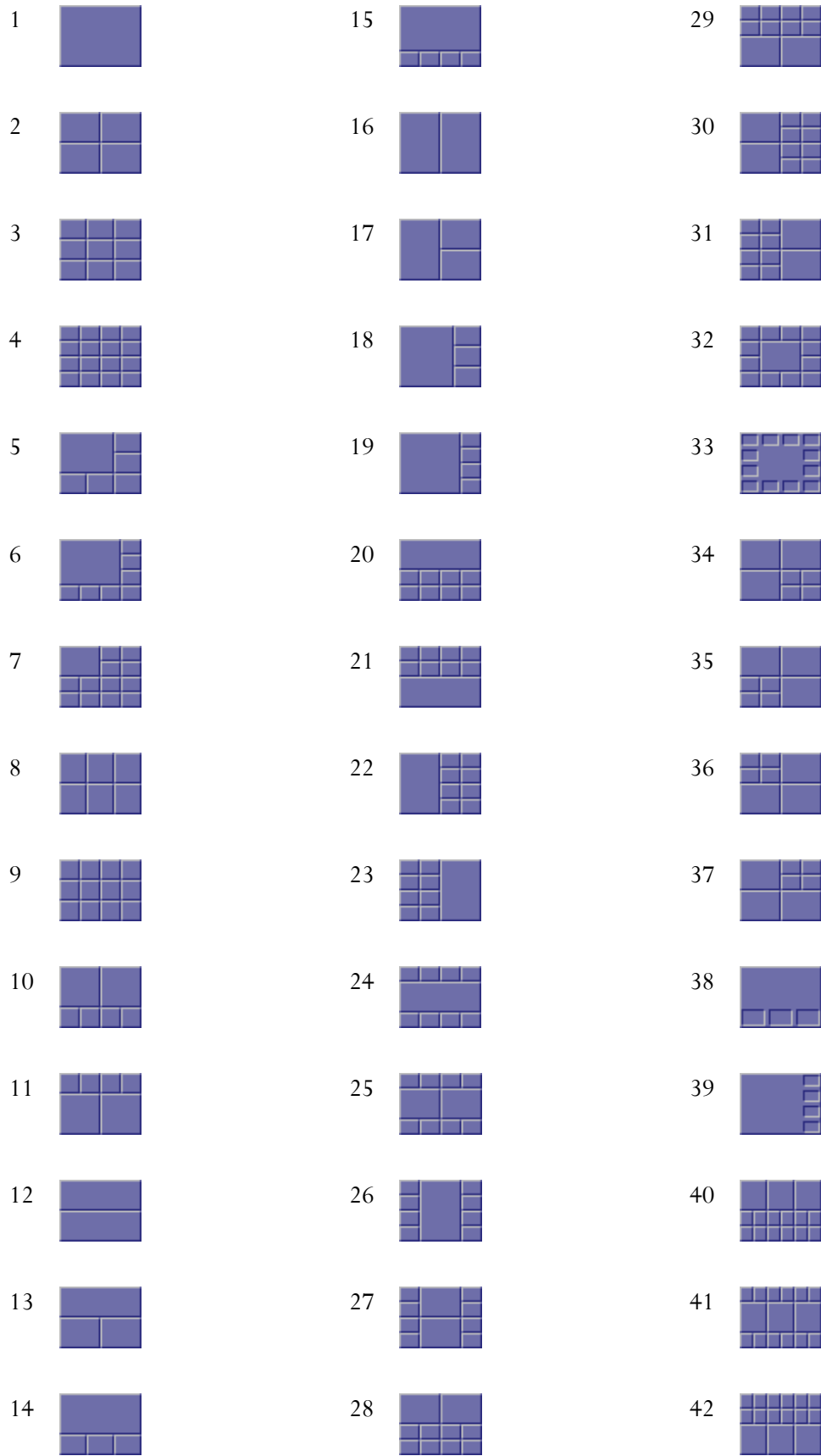
The “`cpLayout`” string parameter can take the following values:

- **default** - use the MCU’s default view family
- **family<index>** - use the specified layout family; see below
- **layout<index>** - use a specific layout; see below

The “<index>” values for “family<index>” correspond to the pane arrangements shown below:



The “<index>” values for “layout<index>” correspond to the pane arrangements shown below. It is worth noting that these indices are also used for the `currentLayout` parameter from the `participant.enumerate` call.





Appendix B - Linking conferences across MCUs

For the purposes of this description, two conferences are said to be "linked" if there is a bi-directional H.323 connection between them and each MCU is sending a video channel to the other, showing the active speaker full screen. The audio communicated between the MCUs will be the usual mix of active speakers. For clarification, the linked conferences are given different names ("linked1" and "linked2") in the explanation, but they can have the same name.

The first step is to set up the two conferences. It is important to ensure that the conferences have a numeric id set (the "conferenceID" field in "conference.create"), because, without this configured field, it is not possible to call in directly to a conference. In this example both conferences are given a numeric id, though strictly it is only necessary on the *target* MCU (i.e. the one that is called rather than the one calling).

In this specific example, "linked1" is set up on "mcu1" and "linked2" set up on "mcu2". The creation of "linked1" is shown in **example message 1**, and it is configured with numeric id "1234"; the creation of "linked2" is shown in **example message 2**, and this conference is given the numeric id "5678".

Next, a participant needs to be added to the "linked1" conference and connected to "linked2" on the target MCU. The most reliable way to accomplish this, which does not rely on the target MCU's gatekeeper usage, is to call from "mcu1" into the target conference using "mcu2" as a gateway and the target conference's numeric id as the remote address. The participant addition is shown in **example message 3** - as well as the address and gateway. It also configures the view layout to be full screen (by setting "cpLayout" to "layout1"), to make sure that just the active speaker from "linked1" is sent to "linked2".

The final step is slightly more complex — it involves modifying the new "linked2" participant on "mcu2" which was the result of the call from "mcu1". The modification required is to change the view layout setting (for the video sent from "linked2" to "linked1") to full screen so that a view of the "linked2" active speaker is sent.

The complication here is that the "linked2" participant in question is not a participant created via the API, and so the API does not know the name in advance. Therefore, it is necessary to

- ▶ poll membership of "linked2" after the connection from "linked1" has been made,
- ▶ identify the participant corresponding to the call, and
- ▶ use its name in a "participant.modify" call to set the view layout.

The simplest way to identify the participant is to look for an absence of the "address" field in a "conference.query" response: for incoming, non-API, connections this will not be present. **Example message 4** shows such a "participant.modify" call; in this case the participant name needed was "1_Codian MCU 4210".

B.1 Example message 1 - creating conference "linked1" on "mcu1"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
```

```

        <string>admin</string>
    </value>
</member>
<member>
    <name>conferenceName</name>
    <value>
        <string>linked1</string>
    </value>
</member>
<member>
    <name>conferenceID</name>
    <value>
        <string>1234</string>
    </value>
</member>
</struct>
</value>
</param>
</params>
<methodCall>

```

B.2 Example message 2 - creating conference "linked2" on "mcu2"

```

<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked2</string>
            </value>
          </member>
          <member>
            <name>conferenceID</name>
            <value>
              <string>5678</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>

```

B.3 Example message 3 - calling into "linked2" from "linked1"

```

<?xml version="1.0"?>

```

```

<methodCall>
  <methodName>participant.add</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked1</string>
            </value>
          </member>
          <member>
            <name>participantName</name>
            <value>
              <string>remote_mcu</string>
            </value>
          </member>
          <member>
            <name>address</name>
            <value>
              <string>5678</string>
            </value>
          </member>
          <member>
            <name>gatewayAddress</name>
            <value>
              <string>10.2.1.27</string>
            </value>
          </member>
          <member>
            <name>cplayout</name>
            <value>
              <string>layout1</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>

```

B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout

```

<?xml version="1.0"?>
<methodCall>
  <methodName>participant.modify</methodName>
  <params>
    <param>
      <value>
        <struct>

```

```

    <member>
      <name>authenticationUser</name>
      <value>
        <string>admin</string>
      </value>
    </member>
    <member>
      <name>conferenceName</name>
      <value>
        <string>linked2</string>
      </value>
    </member>
    <member>
      <name>participantName</name>
      <value>
        <string>l_Codian MCU 4210</string>
      </value>
    </member>
    <member>
      <name>operationScope</name>
      <value>
        <string>active</string>
      </value>
    </member>
    <member>
      <name>cpLayout</name>
      <value>
        <string>layout1</string>
      </value>
    </member>
  </struct>
</value>
</param>
</params>
</methodCall>

```

B.5 Message responses

The response to each of the above method invocations should be the same normal success indication:

```

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value>
              <string>operation successful</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>

```