Virtual Auditorium Gateway Application Programming Interface, GAPI

Vicente Sirvent (UPM), Santiago Pavón (UPM), Mónica Cortés (UPM)

Date: 01/10/2008
Version: 1.1

Table of Contents

Introduction

This document describes the Virtual Auditorium Gateway Application Programming Interface, GAPI, developed to enable a standard API interface to Isabel. This GAPI is needed in order to enable the development of gateways for other video-conferencing protocols such as H323, SIP, sccp, etc signalling. This is required as Isabel uses a proprietary Application Layer Multicast protocol (ALM) and a proprietary control protocol to manage sessions and members. This GAPI will simplify the development of interworking gateways as described in WP4.

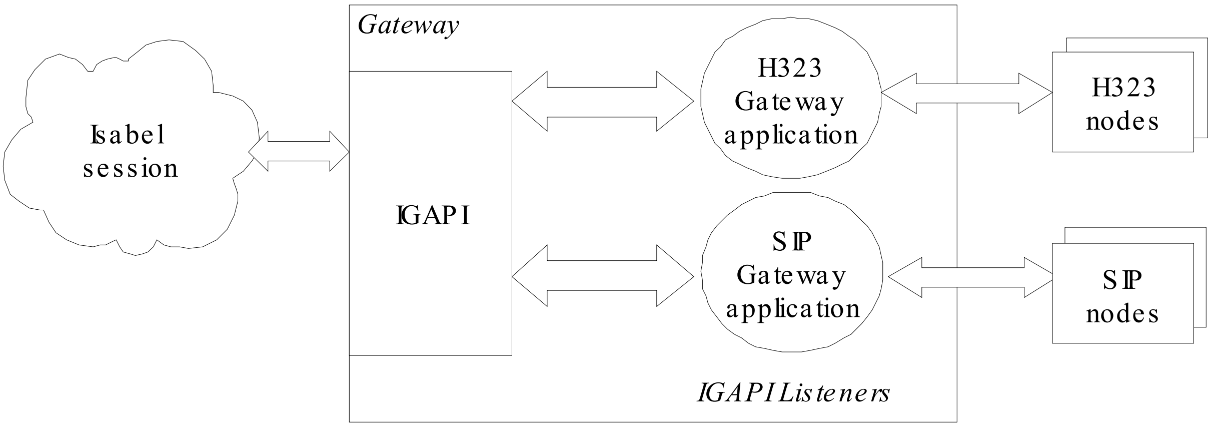
In order to develop this document, we gathered the requirements for Isabel's control protocol, which include member control (active members, join/leaves of the session), data flows (current and change of codec's, formats, audio mix, data flows, video composition).

A very simple API to Isabel already exists, but does not account for all the control data needed. This API is written in Java. The more complete GAPI will initially be prototyped in Java too, though further implementations will allow smarter communication channels (XmlRPC, REST, …).

global-project.eu

GAPI Description

Two Java interfaces will be coded: "IGAPI" interface offers the GAPI commands; "IGAPIListener" is a listener interface that must be implemented in the Gateway application.



In the following sections we will describe the classes, interfaces and enumerators defined in the GAPI package. The next sections deal with control management: membership management, flow management. Message management, QoS management and security management will be provided in future GAPI versions.

1- Classes and interfaces

1.1- IGAPI

This class offers the Gateway API commands to the Gateway application, which will be able to configure any kind of conference scheme by means of this API. Below we list the interface public methods:

- CResult Connect(string login, string password, CMemberInfo info)

Description: Connects an external node to the session.

Parameters:

○ login	user name.
○ password	user password.
○ info	user information data structure.

- *CResult Disconnect(int ID)*

Description: Disconnects an external node from the session.

Parameters:

ID	user identifier.

- *CResult SetupVideo(int ID, EvideoMode video_mode, int PT, int BW, FR, Q, Width, Height, SSRC, EGridMode grid_mode)*

Description: This *optional* command is used to configure video data flows. If not used, the video packets are sent without modifications, the gateway will act as a common network router.

Parameters:

ID	User identifier.
video_mode	Video mode enumerator.
PT	Video codec payload (-1 to keep unchanged)
BW	Maximum bandwidth (-1 to keep unchanged)
FR	Frames per second (-1 to keep unchanged)
Q	Quality factor (-1 to keep unchanged)
Width	Image width (-1 to keep unchanged)
Height	Image height (-1 to keep unchanged)
SSRC	RTP SSRC channel for the output flow (-1 to set random SSRC)
grid_mode	Mosaic mode enumerator (only if video_mode = GRID_MODE).

- *CResult SetupAudio(int ID, EAudioMode audio_mode,int PT,int SL,int SSRC)*

Description: This command is used to configure audio flows. As the “SetupVideo” command, this command is also optional. If not used, audio data packets will be sent without any modification.

Parameters:

ID	User identifier.
audio_mode	Audio mode enumerator.
PT	Audio codec payload (-1 to keep unchanged)
SL	Silence detection level (-1 to disable silence detection feature)
SSRC	RTP SSRC channel of the mixed flow (-1 to set random SSRC)

- *CResult EnableFlow(int PT, int OrigID,int DestID)*

Description: Enable flow routing between “OrigID” and “DestID”.

Parameters:

○ PT	Payloadtype, identifies the codec.
○ origID	Source node user identifier.
○ destID	Destination node user identifier.

- *CResult DisableFlow(int PT, int OrigID, int DestID)*

Description: Disable flow routing between “OrigID” and “DestID”.

Parameters:

○ PT	Payloadtype, identifies the codec.
○ origID	Source node user identifier.
○ destID	Destination node user identifier.

- *CResult AddListener (IGAPIListener listener)*

Description: Adds a listener to the gateway.

Parameters:

○ listener	Reference to the listener object.

- *CResult RemoveListener(IGAPIListener listener)*

Description: Removes a listener from the gateway.

Parameters:

○ listener	Reference to the listener object.

1.2- IGAPIListener

This interface must be implemented by the gateway application and is used by the gateway to send events to the application. The events that must be implemented are:

- CResult AddMember(CMemberIndo info)

Description: This event is received each time a member joins a session.

Parameters:

- info user information data.

- CResult DelMember(CMemberIndo info)

Description: This event is received when a member leaves a session.

Parameters:

- info user information data.

- CResult ActiveFlows(hashDict<int,Set<CFlowInfo>> flows)

Description: This event informs the gateway application about the active flows in the session. This event will be used by the gateway application to configure the external nodes to send only the required flows to the session and optimize network bandwidth usage.

Parameters:

- flows hash table with user identifier key and flow info value.

1.3- CMemberInfo

This class will store member information and is composed of following data types. We have only defined the class members:

int id	Member identifier.
String name	Member name.
String ip	Member IP address.
Set<CFlowInfo> flowList	Member flow information vector.

1.4- CFlowInfo

This class stores data flow related information for each session member. We have only defined the class members:

int pt	RTP payload identifier.

int localPort	local port where gateway will receive the data flow.
int remotePort	remote port where external node will receive the data flow.
int bw	Bandwidth in bps. (a 0 value means unknown bandwidth).

1.5- CResult

This class is used to manage command return values.

-Bool Succedded()

Description: Returns true if the operation was completed successfully, otherwise returns false.

-String GetResultString()

Description: Returns text format result code string (OK, Login error, ...).

-String GetResultDescription()

Description: Returns result description string.

2- Enumerators

We define 3 types of enumerator fields:

2.1- EVideoMode

Description: Enumerates the available video mixing modes.

Values:

- SWITCH_MODE: Each video is sent via single video channel.
- GRID_MODE: Videos are mixed in a single video channel.
- COMMON_GRID_MODE: All videos are mixed in a single common video channel.

2.2- EAudioMode

Description: Enumerates the available audio mixing modes.

Values:

- SWITCH_MODE: Each audio is sent via single audio channel.

- MIX_MODE: All audio channels are mixed in a single audio channel.

2.3- EGridMode

Description: Enumerates the available mosaic video types when mixing video channels.

Values:

- GRID_AUTO: Automatic mosaic.
- GRID_BORDER: Border mosaic.
- GRID_CIRCLE: Circle shape mosaic.
- GRID_PIP: “Picture in Picture” mosaic.