TB640 H.223 User's Guide

Document number
9010-00082-01

November 2005

This page in intentionally left blank

1 TABLE OF CONTENT

2 LIST OF FIGURES

3 LIST OF TABLES

This page in intentionally left blank

The information/code contained in this document/product is based on the best information we have available. Although it has been tested successfully with other piece of equipment, we cannot guarantee that it will conform to the usage of any particular switch in the field.

This page in intentionally left blank

4 H.324M

4.1 Overview

The H.324M standard defines the method used to send and receive streaming video over error-prone channels, as found in mobile applications. It is composed of three major components:

- The H.223 multiplexing/demultiplexing protocol.
- The H.245 protocol for capabilities exchange and channel management.
- Codecs to encode/decode audio and video streams (Media server).

4.1.1 Summary

This document focuses on **TelcoB**ridges H.223 solution. To achieve a complete H.324M system, the customer shall provide required upper parts, H.245 stack and media server.

4.1.2 Architecture

Figure 1 presents the architecture for a typical video playback application using **TelcoB**ridges products.

- H.223 solution on TB640 blade.

- Customer application with TBX host library to communicate with TB640 blade. A H.245 protocol stack for control and communication with peer phone and an optional media server interface to communicate with media server. Media server application can also be included inside customer application.

- Media server application to generate audio and video streams. All streams are sent to TB640 blade via RTP packets. Media server must know audio and video codecs to achieve RTP segmentation and/or video conversion from one phone to one other phone type (transcoding).

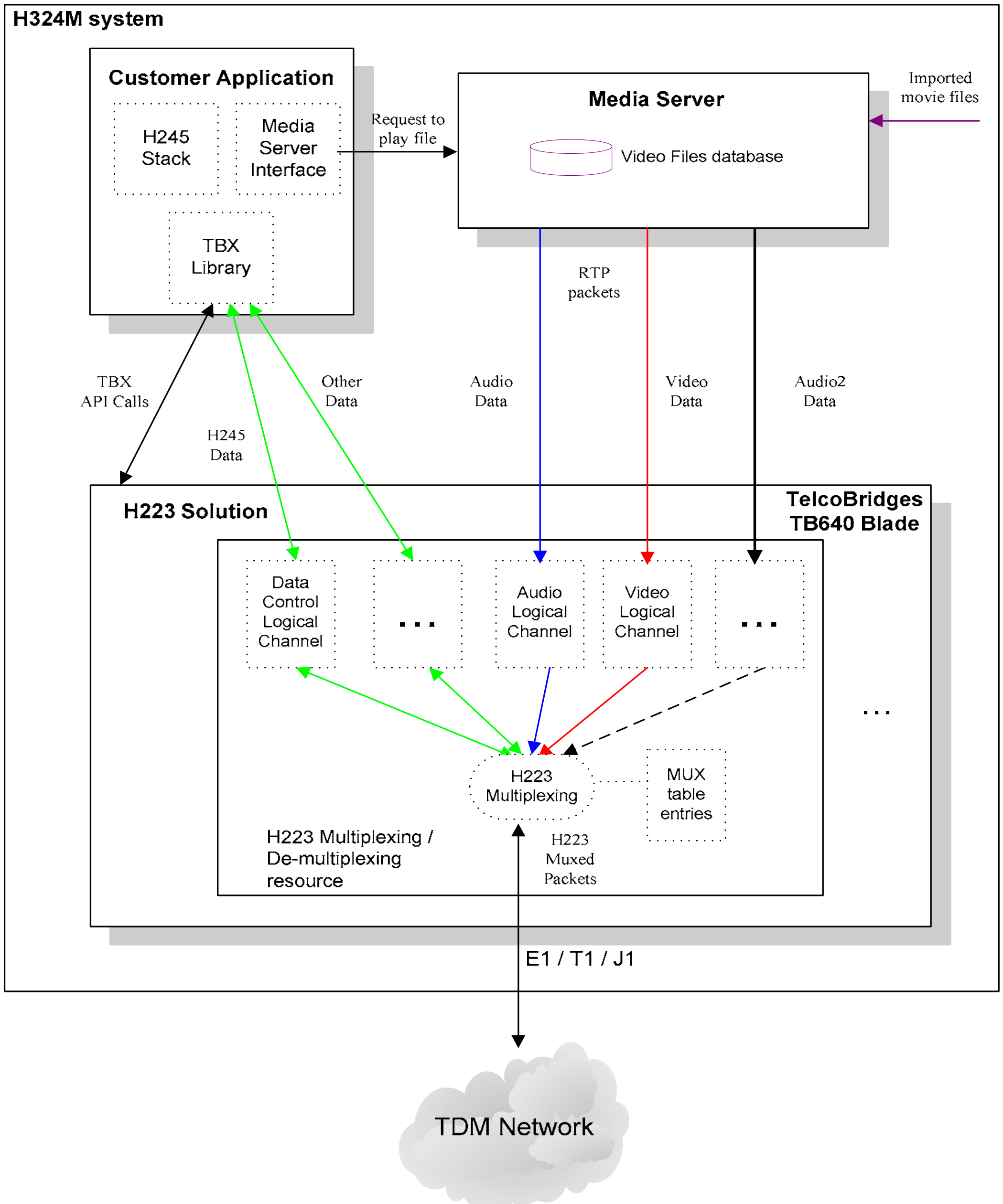

Figure 1 - H324-M Architecture for typical video playback application

5 H.223

5.1 Overview

This section gives a description of the TB640 H.223 architecture and usage. See ITU-T Recommendation H.223, Multiplexing protocol for low bit rate multimedia communication for more details.

5.1.1 Summary

TelcoBridges offering includes a complete H.223 multiplexing/demultiplexing engine capable of transmitting mixed audio, video and data over error-prone channels. An API is provided for the management of multiplexing resources, including management of MUX table entries and AL (Adaptation Layer) channels.

5.1.2 Features

The H.223 Manager supports the following features:

- Up to 127 full-duplex connections per adapter (each including audio, video and data channels).
- Initial Setup procedure at level 0, 1 (Annex A), 2 (Annex B) or Annex B with header.
- AL1, AL2 and AL3 formats.
- Up to 8 jitter buffers of 1460 bytes per channel for transmission.
- Packet confirmation events.
- High/Low watermark events when streaming.

5.2 API Messages

The documentation is accessible via HTML/CHM format. Refer to the TB640 documentation directory in the software package.

5.3 Call setup

This section explains the steps that must be executed by customer application when establishing a call. The main steps are:

- To allocate an H.223 MUX resource and to connect it to a trunk.
- To exchange capabilities with H.245
- To allocate AL channels and MUX table entries and connect AL channels to stream resources.
- To start sending and receiving video stream, audio stream and data on the AL channels.

These steps are explained in more details in the following sections.

5.3.1 Allocating a MUX resource

The allocation of a MUX resource is the first step in any call establishment. It is done through the TB640_MSG_ID_H223_OP_MUX_ALLOC API message. See H.223 Message API documentation for parameters details.

The MUX resource then needs to be connected to an allocated trunk resource. A trunk resource can be allocated using the TB640_MSG_ID_TRUNK_RES_ALLOC message while the connection is accomplished through the TB640_MSG_ ID_CONN_OP_CREATE message.

Once the connection is made, the H.223 MUX will try to synchronize with a peer H.223 MUX and agree on the format to use, either level 0, 1 (Annex A), 2 (Annex B) or Annex B with header. Once the synchronization is achieved, the TB640 blade will automatically allocate AL resources on channel 0 and default MUX table entries with index 0 for both directions (to TDM and from TDM). It will send an event of type TB640_MSG_ID_H223_NOTIF_MUX_SYNC to the host, indicating the level at which it connected and the two AL resources handles.

AL resources on logical channel 0 are used to send/receive H.245 packets. AL parameters for logical channel 0 are:

Table 1 – AL channel 0 parameters

Parameters	Value	TB640 value
Channel number	0	0
Channel type	Data	TB640_H223_MUX_CHANNEL_TYPE_DATA
AL option	Segmentable	TB640_H223_MUX_AL_OPTION_SEGMENTABLE
AL type	AL1	TB640_H223_MUX_AL_TYPE_AL1
Packet type	H223	TBX_STREAM_PACKET_TYPE_H223
AL1 option	Framed	TB640_H223_MUX_AL1_OPTION_FRAMED
Max RTP size	Not used	0
Max SDU size	Unlimited	0
Watermarks	Disabled	0

MUX table entry 0 is used to send/receive data on LCN0 (logical channel number 0). MUX table parameters are:

Table 2 – MUX table entry 0 parameters

Parameters	Value	TB640 value
Table entry index	0	0
Entry description	{LCN0,RC UCF}	{LCN0,RC UCF}

5.3.2 Exchanging capabilities

Using the newly allocated LCN0 handle, the application must synchronize the H.245 stack with the peer by exchanging H.223 capabilities and negotiating the master/slave configuration. With TB640_MSG_ID_H223_OP_GET_CAPABILITY message, the application can retrieve TB640 blade H.223 capabilities, **format a H.245 message** and send it using TB640_MSG_ID_H223_OP_MUX_AL_SEND message. H.245 peer messages will be received with TB640_MSG_ID_H223_NOTIF_MUX_AL_RECEIVE event.

5.3.3 Allocating MUX table entries and AL channels

H.245 protocol is used to open and close logical channel and associated MUX table entries. Generally, a H.245 command is sent (request multiplexEntrySend, request openLogicalChannel, etc) to initiate the communication. When the corresponding acknowledge is received, the TB640_MSG_ID_H223_OP_MUX_TABLE_ENTRY_ALLOC and TB640_MSG_ID_H223_OP_MUX_AL_ALLOC messages are used to open the required MUX table entry and LCN (To TDM direction) on the TB640 blade.

✍ The H.245 stack should answer the requests from peer and all required MUX table entries (TB640_MSG_ID_H223_OP_MUX_TABLE_ENTRY_ALLOC) and LCN in reception direction (From TDM) need to be opened with TB640_MSG_ID_H223_OP_MUX_AL_ALLOC message.

5.3.4 Connecting Channel Resources

Streaming data to/from an audio or video AL resource requires connecting this resource to a RTP stream resource on the TB640 blade. The stream resource is allocated using the TB640_MSG_ ID_ STREAM_RES_ALLOC message and the connection is accomplished through the TB640_MSG_ ID_CONN_OP_CREATE message.

✍ Figure 2 shows all connections for a playing application, all arrows are going toward TB640 blade. TB640 H223 solution also supports record or mixed application.

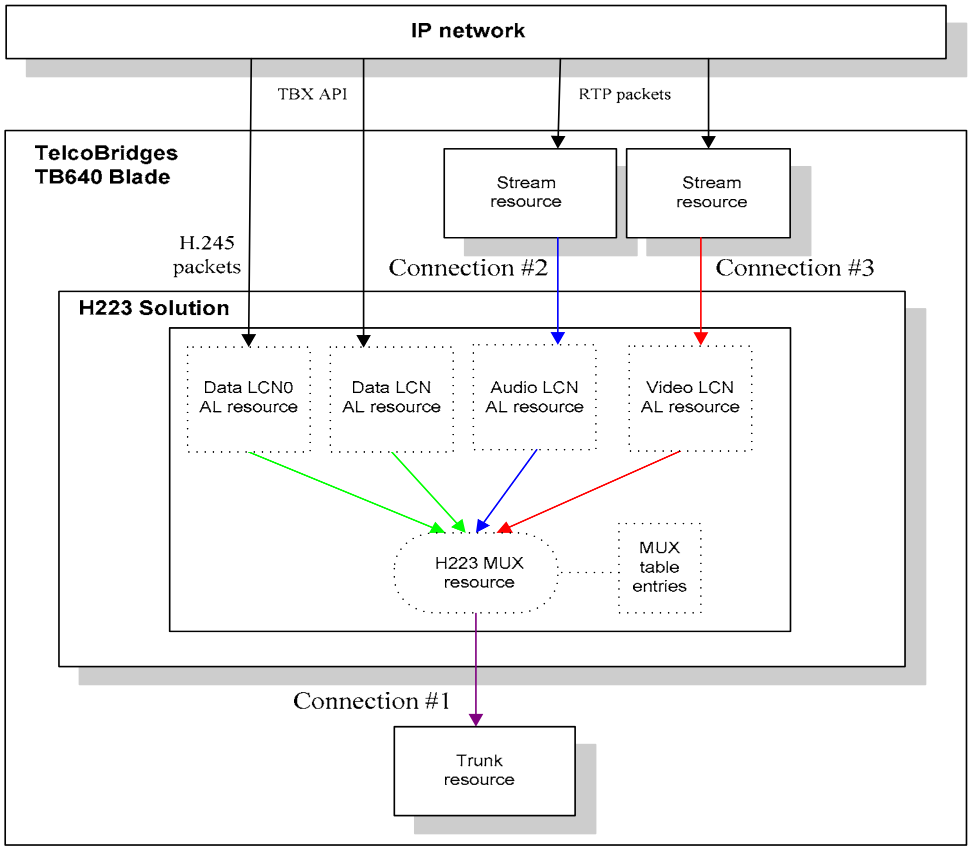

Figure 2 – How to connect H.223 resources

5.3.5 Sending and receiving data on AL resource

The recommended way to send data such as H.245 message is by API messages. For outgoing packets, TB640_MSG_ID_H223_OP_MUX_AL_SEND message is used. When the packet is completely transmitted, a TB640_MSG_ID_H223_NOTIF_MUX_AL_SEND_END event is generated from TB640 blade. There can be a maximum of two packets for each AL data channel in the waiting queue on the TB640 blade before sending a third one. This means the application needs to wait for TB640_MSG_ID_H223_NOTIF_MUX_AL_SEND_END event before sending the third one, this event acts as a confirmation. If the application try to send this third packet before waiting for the acknowledge, the TB640 blade will refuse to send it and return an error code.

For incoming packets, TB640_MSG_ID_H223_NOTIF_MUX_AL_RECEIVE event is used to indicate a new data packet. This event returns a timestamp in millisecond and a flag indicating if the data contains CRC errors.

5.3.6 Sending and receiving stream on AL resource

Streaming is achieved with RTP packets over the local IP network. The media server transcodes audio and video files (or other media sources) in a format known by the peer phone. They are segmented in several AL-SDU frames and finally played at the required speed via RTP packets by the media server. Huge AL-SDU frame (over 1460 bytes) requires to be split in several RTP packets. The marker flag in RTP header is used to indicate the end of an AL-SDU frame.

AL resource stream rate is the responsibility of the media server application and the required total bandwidth for all AL resources on the same MUX resource should not exceed 64 kbps (1 TDM timeslot).

Table 3 – AL Stream rate example

LCN	Type	Rate
0	Control data	4 kbps
10	Audio	12 kbps
12	Video	40 kbps
Low level	MUX Headers and flags, AL Headers	3 kbps (may vary)
Total Bandwidth		59 kbps

TB640 stores up to eight incoming RTP buffers, it can also generate event when it reaches a high watermark threshold or comes back to the lower threshold configured by customer application.

TelcoBridges provides a stream library to accelerate the development of customer application. This library generates RTP packets and offers a callback function to receive RTP packets. This stream library also provides network redundancy with the TB640 blade. See *TBX Streamlib API Reference Guide* document for more information.

✍ The segmentation feature provided in stream library should not be used by H.324M system because video stream, for example, requires a precise segmentation. Packets cannot be segmented everywhere in the audio or video stream.

5.3.7 Channel resource management

When the call is completed, every resource needs to be freed. Each allocation message as a free message associated that must be called in reverse order. TB640_MSG_ID_H223_OP_MUX_FREE message must be called last. Other useful API messages are used to manage resources.

- Get/Set parameters without freeing the resource
- Get list of one kind of resource
- Get statistics
- Get states

5.4 MUX table creation

Multiplex entry descriptors are various and unlimited. The basic implementation is to configure one MUX table entry for each AL channel allocated. This works in most of the cases but this may cause channel starvation. This is why it is recommended to create one or several multiplex table entries with mixed AL channels. The best practice is to have a multiplex table entry starting with a fixed number of audio channel bytes following by video channel bytes until the end of the frame. The fixed number of audio bytes should correspond to the size of an audio sample plus the AL overhead of the audio channel.
Table 4 illustrates an example of MUX table entries that should be used in a scenario when the audio channel transport AMR sample, most of them being 12.2kbps samples (31 bytes), silence samples (6 bytes) or empty samples (1 byte). We'll consider that AL2 without sequence number is used for that channel, so we must add 1 byte for the CRC.

LCN0 stands for control channel, LCN1 for audio channel and LCN2 for video channel.
Entries 3, 4 and 5 will protect starvation of AL audio channel (LCN1) and enhance synchronization between the audio and video streams.

Table 4 – MUX table entry examples

MUX table entry	Description	Result
0	{LCN0,RC UCF}	All H.245 data, configured by default
1	{LCN1,RC UCF}	All audio
2	{LCN2,RC UCF}	All video
3	{{LCN1,RC2},{LCN2,RC UCF}}	2 audio bytes, then all video
4	{{LCN1,RC7},{LCN3,RC UCF}}	7 audio bytes, then all video
5	{{LCN1,RC32},{LCN3,RC UCF}}	32 audio bytes, then all video

End of the document