PV2Way Developer's Guide
OpenCORE 2.0, rev. 2
Feb 6, 2010

Table of Contents

List of Figures

1. Introduction

This document is a guide for developers writing clients to the PV2Way SDK. A client of PV2Way can be an application or an adapter layer to adapt PV2Way interface to a higher-level interface used by the application. This document describes how to use PV2Way interface and its extensions to create, configure and control a video telephony session.

PV2Way is a platform-agnostic SDK that provides video telephony capabilities for its clients. The SDK is capable of executing 3G-324M standard video telephony calls. The input media data are typically provided by live source(s) such as camera and microphone. Output media data is forwarded to presentation sinks such as display and speaker. The sources and sinks are provided by the client application.

2. Architecture

2.1. Static Design

The following diagram illustrates the major participants in a video telephony session using the PV2Way SDK. A client obtains a reference to the 2way engine (CPV2WayInterface) using either of the factory classes CPV2WayEngineFactory or CPV2WayProxyFactory. The client must also implement PV2Way's observer interfaces in order to receive command completion, status information and error information. PV2Way is dependent on its client to provide media data sources and sinks for a video telephony session. These media sources and sinks should either be a Media IO Component (MIO) or implement the PVMFNodeInterface (Node) to allow PV2Way Engine to control them in a generic way. A reference to the communications end point is needed by PV2Way to communicate with a peer terminal. This interface acts as the data link to the peer terminal for the video telephony session. Just like media sources and sinks, this can be implemented either as an MIO or a Node. The Client control of the session is performed through CPV2WayInterface. The figure below shows the relationship between PV2Way, the client, and other objects owned by the client. MIOs are described in detail later in the document.

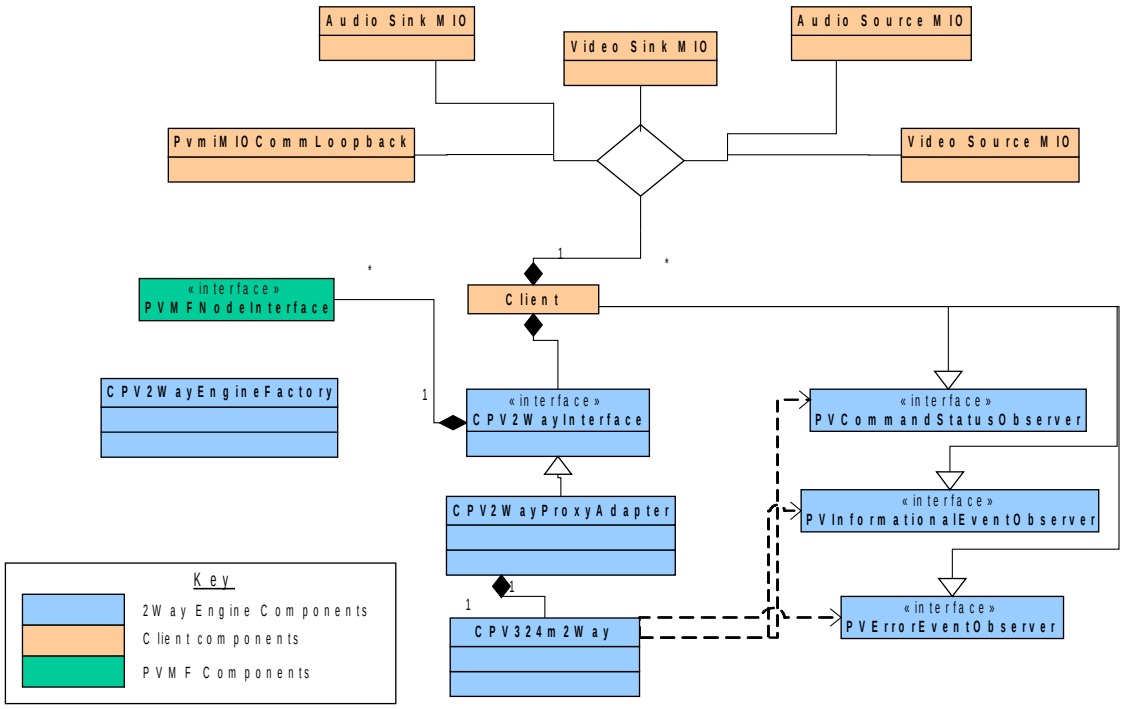

Figure 1: High Level Class Diagram

2.2. SDK State Machine Design

PV2Way SDK has 7 states: Idle, Initializing, Setup, Connecting, Connected, Disconnecting, and Resetting.

- **Idle**: The state immediately after the PV2Way instance has been successfully created or instantiated. Some optional PV2Way components must be selected in this state.
- **Initializing**: The PV2Way is in this state when it is initializing from the Idle to the Setup state. The terminal queries the available device capabilities (encode, decode, mux), acquires resources to make a two-way call (codecs, formats, memory etc) and transitions to the Setup state when it will be ready to accept setup parameters and Connect. If initializing fails, the PV2Way relinquishes the resources and reverts to the Idle state.
- **Setup**: The state where the PV2Way instance is in the process of receiving setup parameters from the application, for encoding, multiplexing, capturing and rendering. Each time a new set of parameters is passed in, validation will take place and a status will be returned accordingly.

- **Connecting**: The state where the PV2Way instance has received a call to start connecting. The communications end point is also to be provided as part of this process. The PV2Way engine starts communication with the remote terminal to exchange media capabilities and channel configuration in preparation for the establishment of media channels.
- **Connected**: The state after all mandatory control signaling is completed. PV2Way will establish media tracks based on local and remote capabilities and indicate outgoing track establishment to the client, upon which the client is required to provide media sources for these tracks. Similarly, incoming channel notifications will be passed to the client after which the client can add media sinks to associate with particular incoming channels.
- **Disconnecting**: The state where the terminal is shutting down all channels and the multiplex.
- **Resetting**: The state where the terminal is releasing all resources and transitioning to the Idle state.

To transition from one state to another, the user will need to call the session control APIs of PV2WayInterface. The figure below illustrates state transition of PV2Way SDK.

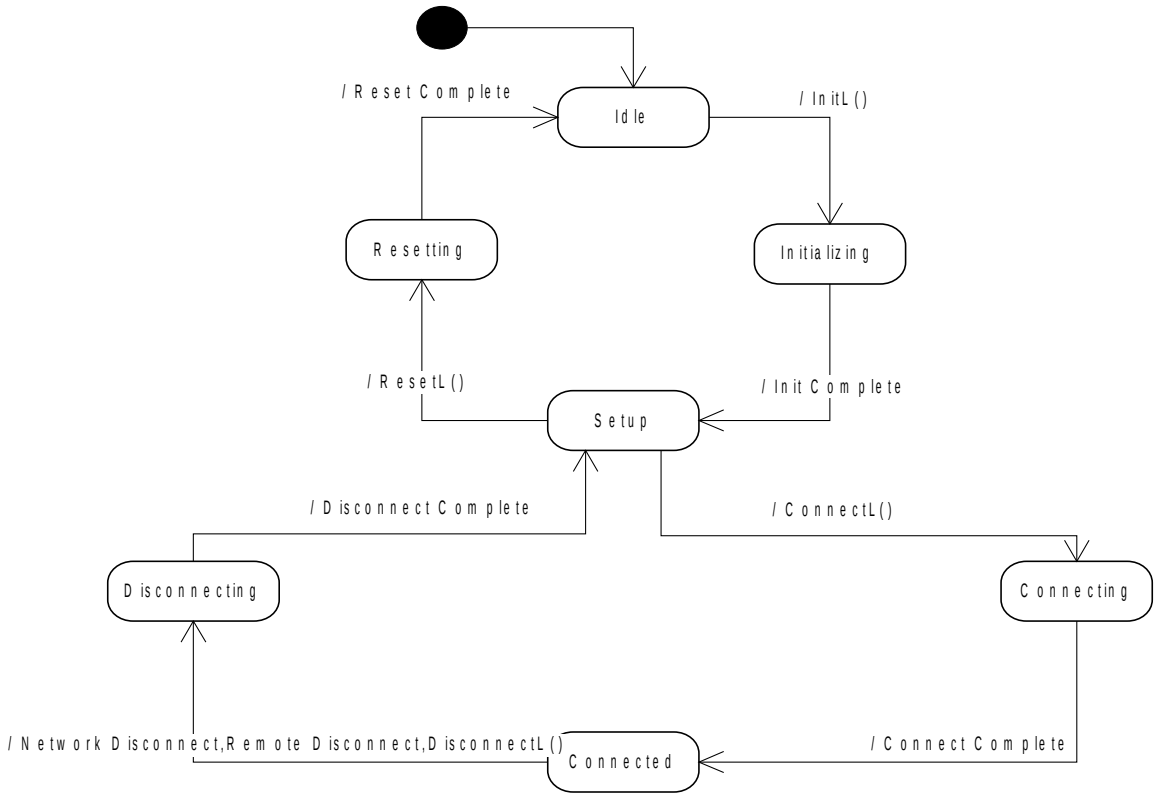

Figure 2: State Transition Diagram

2.3. Command Processing

The PV2Way SDK exposes functionalities via commands implemented either by the CPV2WayInterface class or by extension interfaces. The following are some common features of these APIs:

1. Most APIs to the PV2Way SDK are asynchronous. The user is expected to implement an observer to monitor the status of issued commands.
2. The asynchronous APIs return a unique command id if the request is accepted.
3. The API will leave if there is an error queuing the request.
4. Optional opaque data can be passed in with every asynchronous call to allow the user to associate arbitrary data/state information with the call.
5. Reference to data structures can be passed in some APIs to allow the user to specify where the data should be written. The PV2Way SDK will have complete ownership of the reference until the call is completed
6. Once the command is processed, the user is notified via the command observer of the command id, completion status, the opaque context data and additional opaque data which is to be interpreted based on the command type and the completion status.
7. All pending commands can be cancelled by using the CancelAllCommands API. The CancelAllCommands command itself cannot be cancelled. Cancelled commands will still return a command completion notification with an indication that it was cancelled.

2.4. Sources and Sinks

PV2Way can handle several types of media sources and sinks and can adapt its internal configuration based on the types of the sources and sinks, and the time when they are added to the PV2Way.

2.4.1. Defining Media Capability

The user of the SDK should define the scope of the call during initialization. The PV2WayInitInfo class allows the user to advertise the formats and capabilities of available sources and sinks for each incoming and outgoing channel of media that is to be established. The order of formats specifies the preference in decreasing order. For instance, user may define the following:

Available outgoing audio formats: {AMR}

Available video formats: {Mpeg4, YUV422}

Available incoming audio formats: {AMR}

Available incoming video formats: {Mpeg4, YUV420}

These capabilities along with the capabilities of the PV2Way engine (internal codecs, stack) will be used to negotiate the session with the peer. However, it is not guaranteed that the most preferred formats will be negotiated or, that any format will be negotiated at all.

2.4.2. Media IO Components

In PV2Way SDK and PV multimedia framework architecture, the Media IO (MIO) component is a communication layer for the 2way engine to interact with the device's capturing, rendering and network interfaces. As access to media capturing/rendering & network functionalities are different for every platform that PV software works on, the MIO component serves as a glue layer for different PV modules. In order to act as an adapter between PV modules and underlying hardware interfaces, a MIO component has several responsibilities.

- Control the underlying hardware based on commands from other PV modules.
- Exchange capabilities information of the underlying hardware with other PV modules.
- Send/Receive media data to/from the different underlying hardware interfaces.

Several PV2Way engines APIs (CPV2WayInterface) accept a reference to PVMFNodeInterface for sources and sinks (including the COMM interface). Factories that provide a node wrapper for a given MIO are provided in the PV2Way SDK so that the client does not have to adapt to the PVMFNodeInterface, but can simply provide all its adaptations using MIOs. The PVCommsIONodeFactory class creates a node wrapper for the COMM MIO. The PvmfMediaInputNodeFactory class creates a node wrapper for media input MIOs. The PvmfMediaOutputNodeFactory class creates a node wrapper for the media output MIOs.

In the rest of this document, a node is used interchangeably with an MIO in the context of sources, sinks and the comms interface.

Detailed information on Media I/O components can be found in the "Media I/O Developers Guide."

2.4.3. Track Indications

Media sources and sinks should be added after an indication from PV2Way is received notifying establishment of the track (PVT_INDICATION_OUTGOING_TRACK/ PVT_INDICATION_INCOMING_TRACK). This can happen in the EConnecting and EConnected states. The indications convey the following:

- The port tag associated with the track. The Client is expected to pass in this tag when it calls AddDataSource/AddDataSink.
- The media type associated with the track
- A PV2WayTrackInfoInterface extension interface, which conveys the format type as a MIME string, Format Specific Information if any, etc.

The following are the types of sinks/sources supported:

- Raw data sources: These sources output uncompressed data – YUV, PCM etc.
- Compressed data sources: These sources output compressed data – MPEG-4, H.263, AMR.
- Raw data sinks: These sinks receive uncompressed data – YUV, PCM etc. The PV2Way Engine will establish a codec in the datapath to convert the incoming compressed bitstream to a raw type supported by the sink.
- Compressed data sinks: These sinks receive compressed data – MPEG-4, H.263, AMR.

2.4.4. Removing sources and sinks

Sources and sinks added during a call should be removed before disconnecting the call.

2.4.5. Channel re-negotiation

If the peer does not support one of our outgoing track formats or if a conflict arises, one or more of the tracks may need to be closed and replaced with tracks agreeable to both sides. As soon as the PV2Way SDK detects this condition, it passes a PVT_INDICATION_CLOSING_TRACK indication to the Client specifying the unique tag for the channel. Subsequently, it closes the track and passes a PVT_INDICATION_CLOSE_TRACK indication. This is illustrated in Section 3.8.

For outgoing channels the Client needs to wait for the PVT_INDICATION_OUTGOING_TRACK indication and then add a compatible source. For incoming channels, the user needs to wait for the PVT_INDICATION_INCOMING_TRACK indication before adding a compatible sink.

2.5. Comm Interface

The Comm Interface represents the data link to the remote video telephony terminal. It typically implements a 64kbps circuit-switched connection to a serial-port or the baseband of a phone. The PV2Way Engine interfaces to it using an MIO or PVMFNodeInterface. Once the Comm Interface indicates a successful start the PV2Way Engine begins sending and receiving a 3G-324M compliant bitstream with the peer terminal via the Comm Interface.

More details on this can be found in Reference 4.

2.6. Lip Synchronization

The strategy for lip synchronization of incoming media is pretty similar to how its handled in the PVPlayer, so only the differences are addressed here.
The pv2way engine will provide timestamped audio and video samples to the Media I/O Components along with a clock just like the PVPlayer case. The active audio rendering MIO would adjust the clock based on the number of samples rendered and the video rendering MIO would pace itself based on the shared clock.
–The pv2way engine would attempt to smoothen the incoming timestamps especially for audio to reduce jitter. The jitter comes from the fact that the protocol does not support bitstream timestamps. Without bitstream timestamps, the timestamps are based on arrival time and this makes them prone to a lot of jitter.
–The pv2way engine disables buffering/delaying/dropping of audio and video within 2way datapaths for lip synchronization. Buffering/delaying/dropping decisions are left to the MIOs.
–Incoming skew indication will be forwarded to the application via the H324MConfigInterface. This provides any inherent skew between the incoming audio and video streams that needs to be compensated for at the time of rendering. This is not very common, but applications need to be able to handle it if the skew is significant.
For outgoing media, the pv2way engine monitors the outgoing audio and video streams for significant skew and if detected, it issues a Skew Indication message to the peer.

2.7. Extension Interfaces

Extended configuration of the engine for protocol-specific features and for value adds may be acheived using extension interfaces (if they are supported.) This is done using the QueryInterface mechanism.
H.324-specific configuration may be acheived using the H324MConfigInterface.

More details may be found in the pv2way engine API document.

3. Usage Scenarios

The following sections illustrate the interactions that happen between the client and the PV2Way components to handle some of the typical commands.

3.1. Initializing

The Init command initializes an instance of PV2Way and acquires all the resources for a video telephony call. The Command Status Observer is notified of the outcomes.

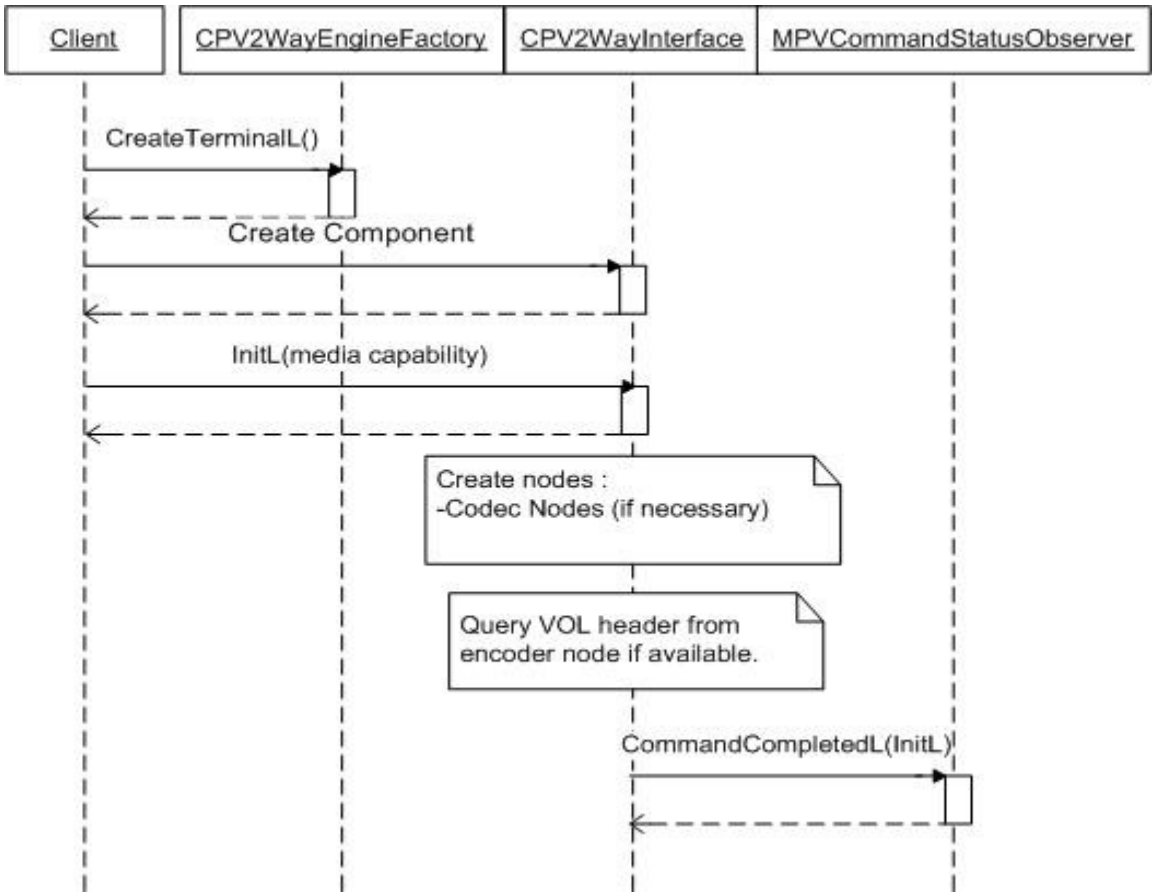

Figure 3: Sequence Diagram - Initializing

3.2. Connecting

Commanding PV2Way to Connect begins the 3G-324M call setup procedure. The CommNode is used as the network connection to the peer 3G-324M terminal and must therefore be connected to the shared data link once the Start command is returned with a successful result.

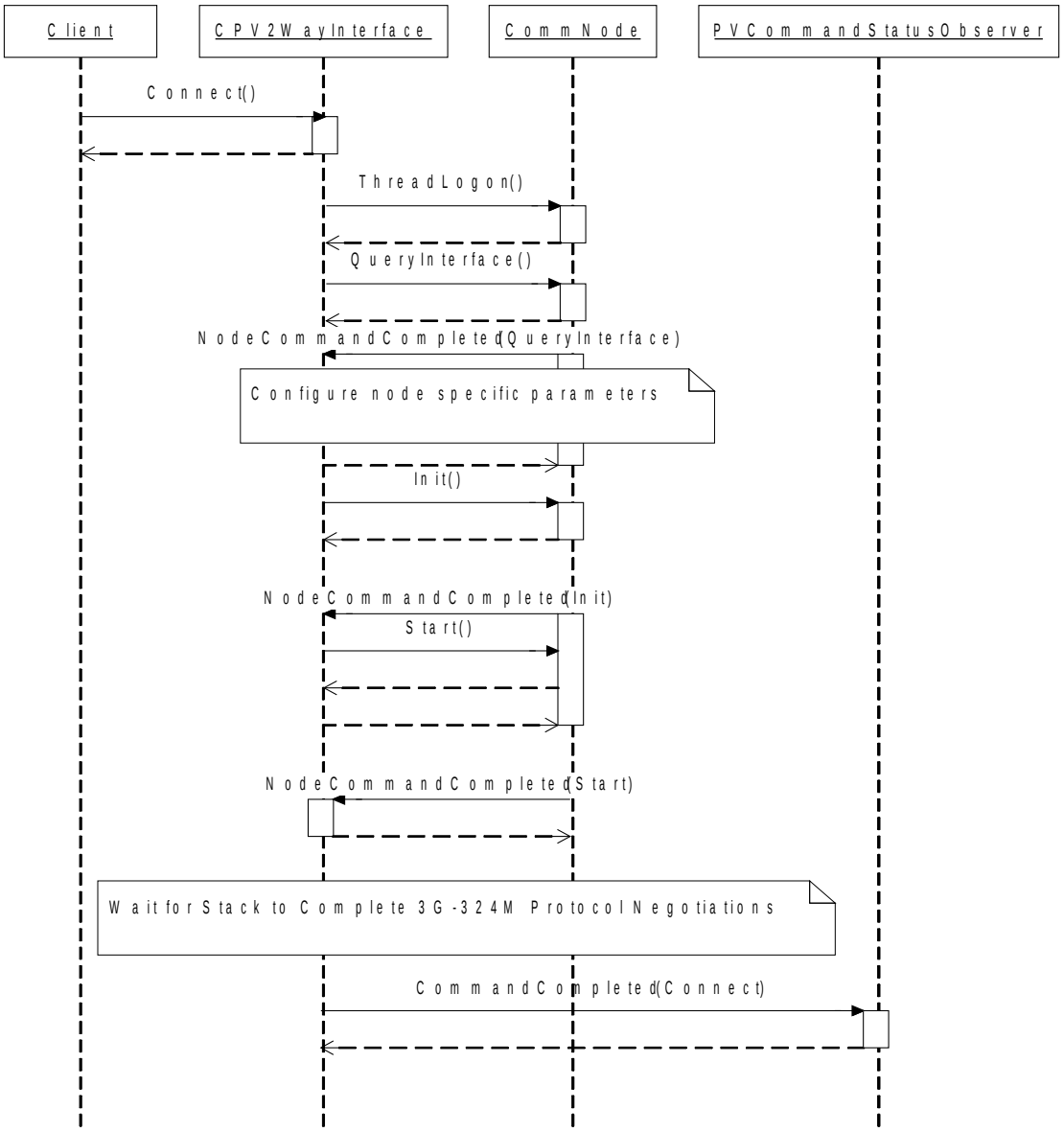

Figure 4: Sequence Diagram - Connecting

3.3. Adding Data Sources

The PV2Way Client needs to create media i/o components for source data (audio, video) and provide them to PV2Way via the AddDataSource() method after receiving the PVT_INDICATION_OUTGOING_TRACK for the channel. Data from these sources would be (optionally) encoded, multiplexed and transmitted to the peer. The PVMFMediaInputNode is used to encapsulate the media i/o components before passing them via the AddDataSource call.

The media i/o component is configured using the capability exchange interfaces.

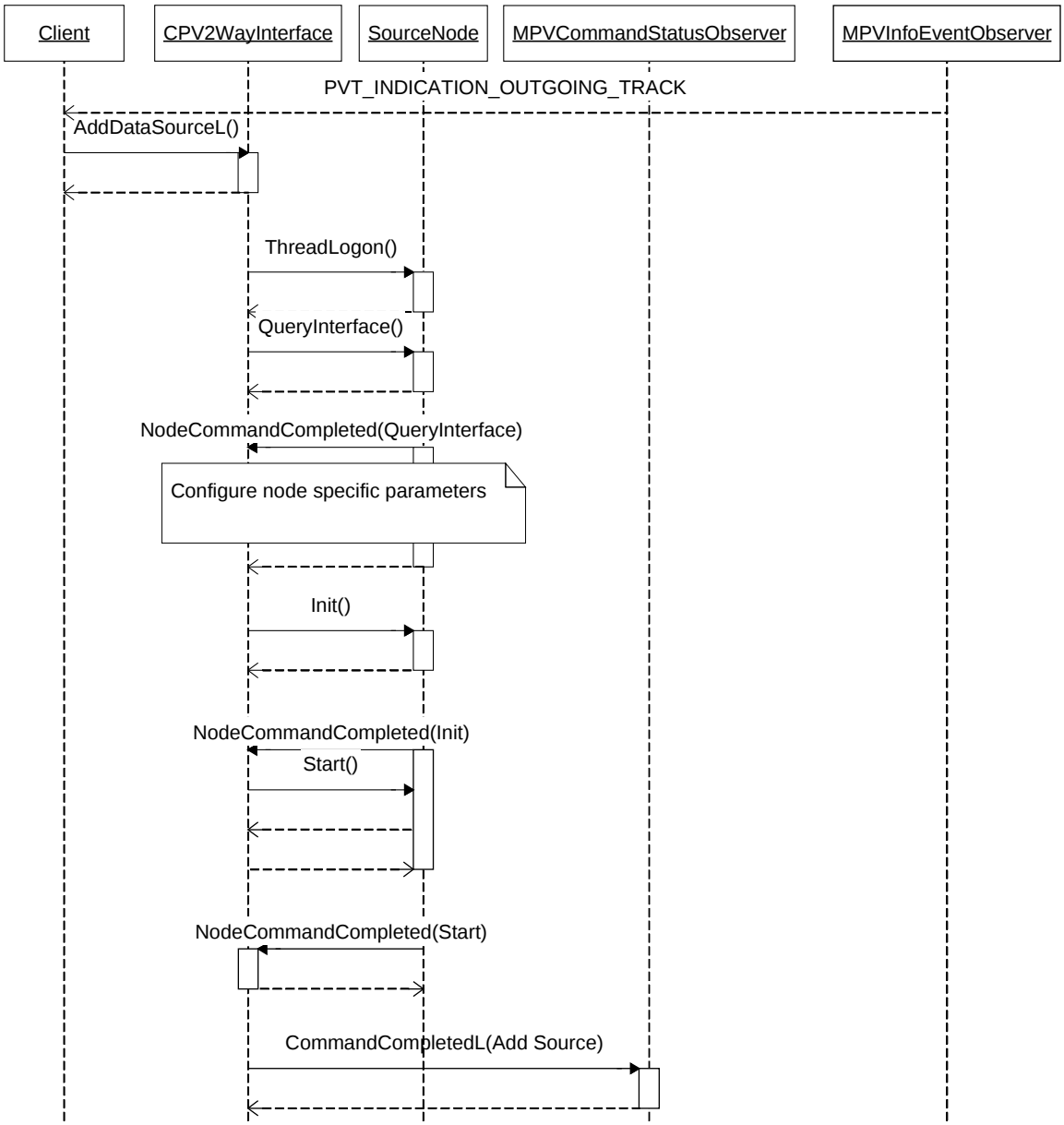

Figure 5: Sequence Diagram - Adding a Data Source

3.4. Removing Data Sources

Data Sources should be removed before disconnecting the call.

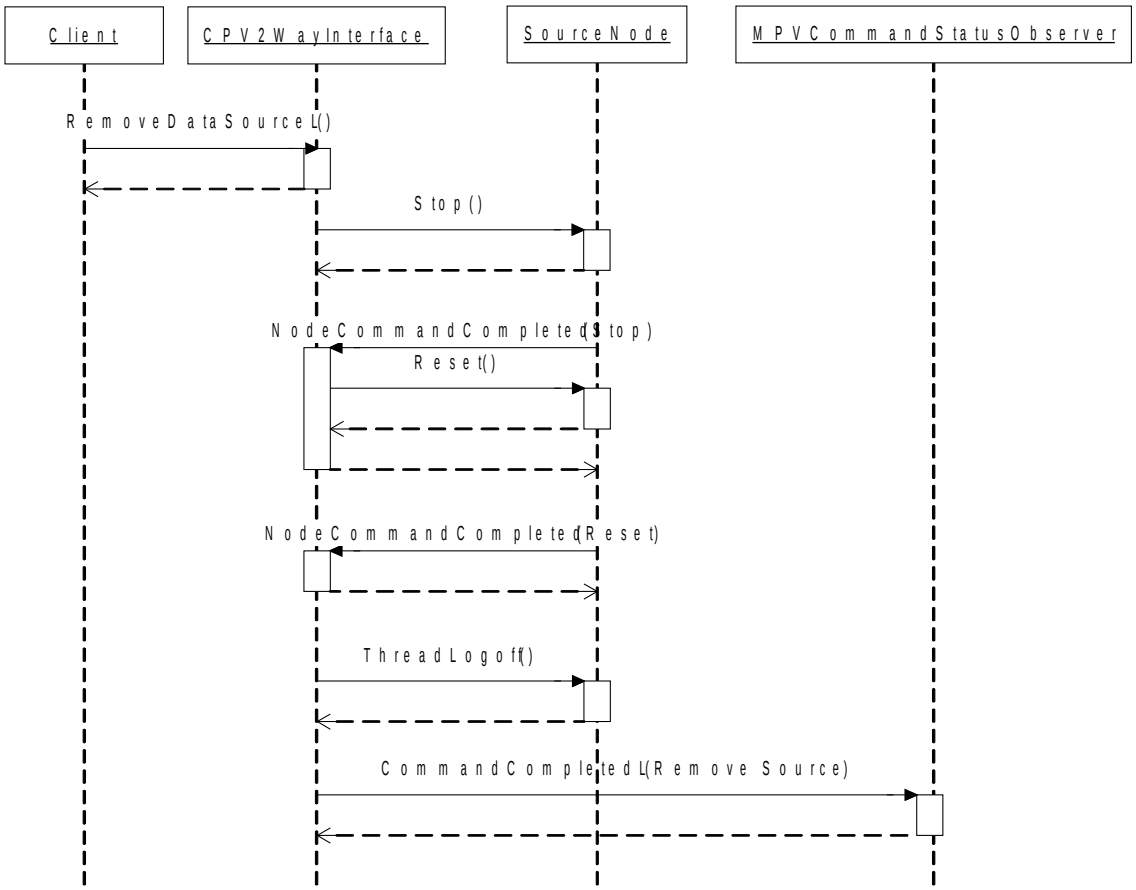

Figure 6: Sequence Diagram – Removing a Data Source

3.5. Adding Data Sinks

The PV2Way Client needs to create media i/o components for sink data (audio, video) and provide them to PV2Way via the AddDataSink() method after receiving the PVT_INDICATION_INDICATION_TRACK for the channel. Incoming media data from the peer would be de-multiplexed, (optionally) decoded and provided to these sinks. The PVMFMediaOutputNode is used to encapsulate the media i/o components before passing them via the AddDataSink call. The media i/o component is configured using the capability exchange interfaces.

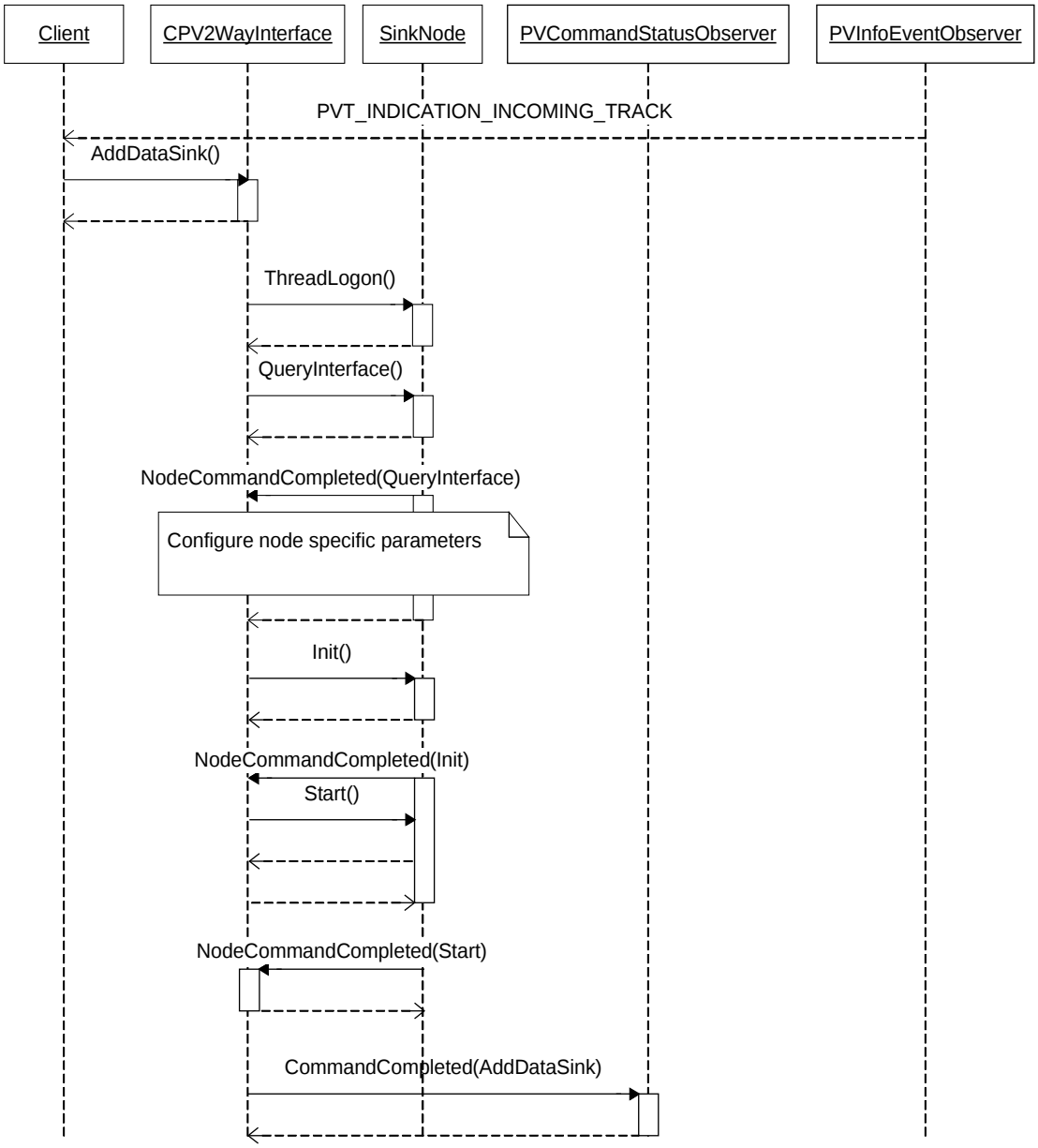

Figure 7: Sequence Diagram - Adding a Data Sink

3.6. Removing Data Sinks

Data Sinks should be removed before disconnecting the call..

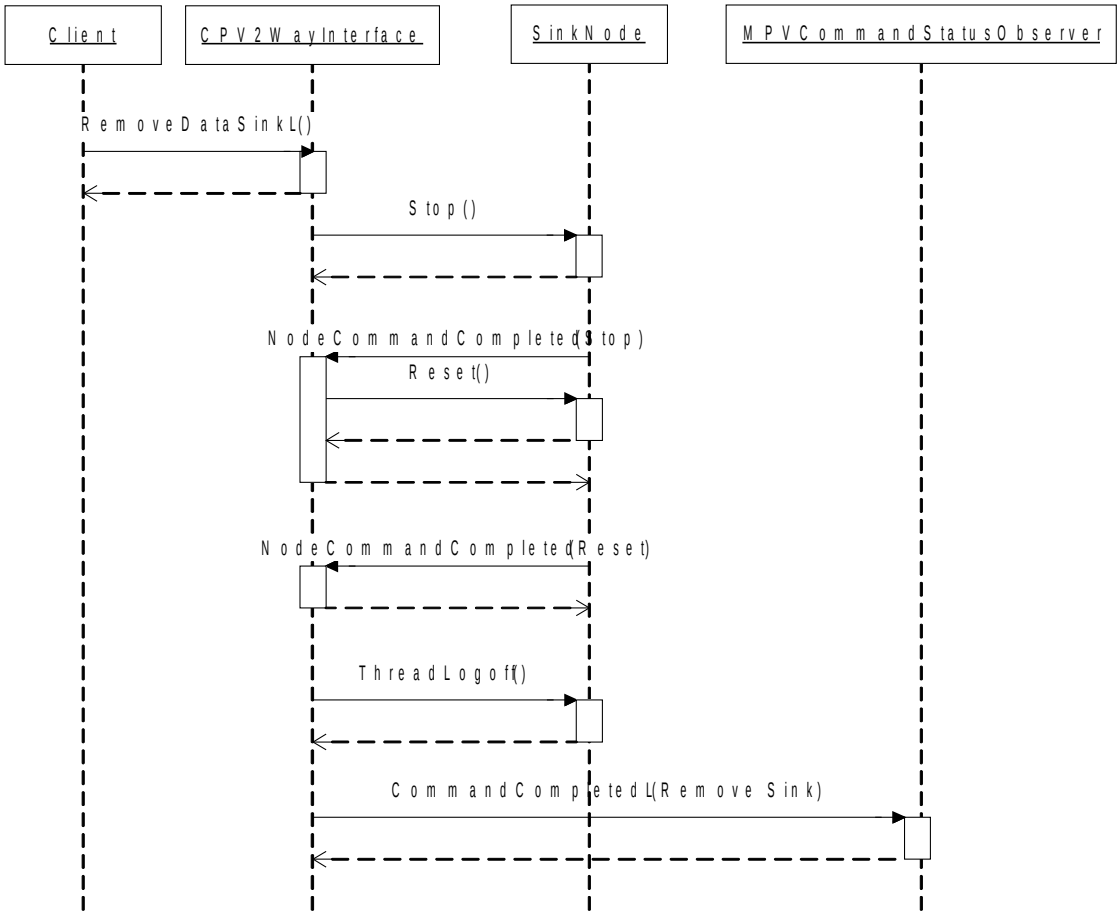

Figure 8: Sequence Diagram - Removing a Data Sink

3.7. Data Sink Removed

The following events happen when a track is closed by the peer during a video conferencing session.

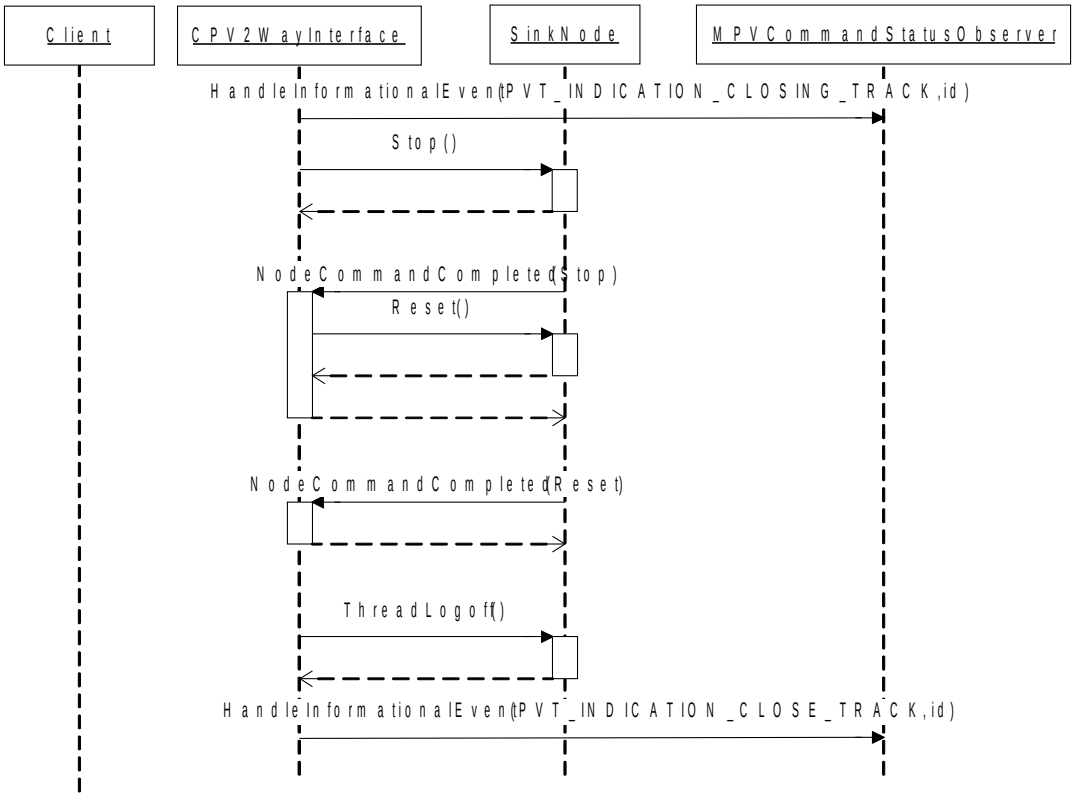

Figure 9: Sequence Diagram - Data Sink Removed

3.8. Disconnecting

To terminate a video telephony call, a client needs to call Disconnect after all the Sources and Sinks are removed. This will cause protocol negotiations (EndSessionCommand) with the peer after which the multiplex and the Comms MIO will be stopped.

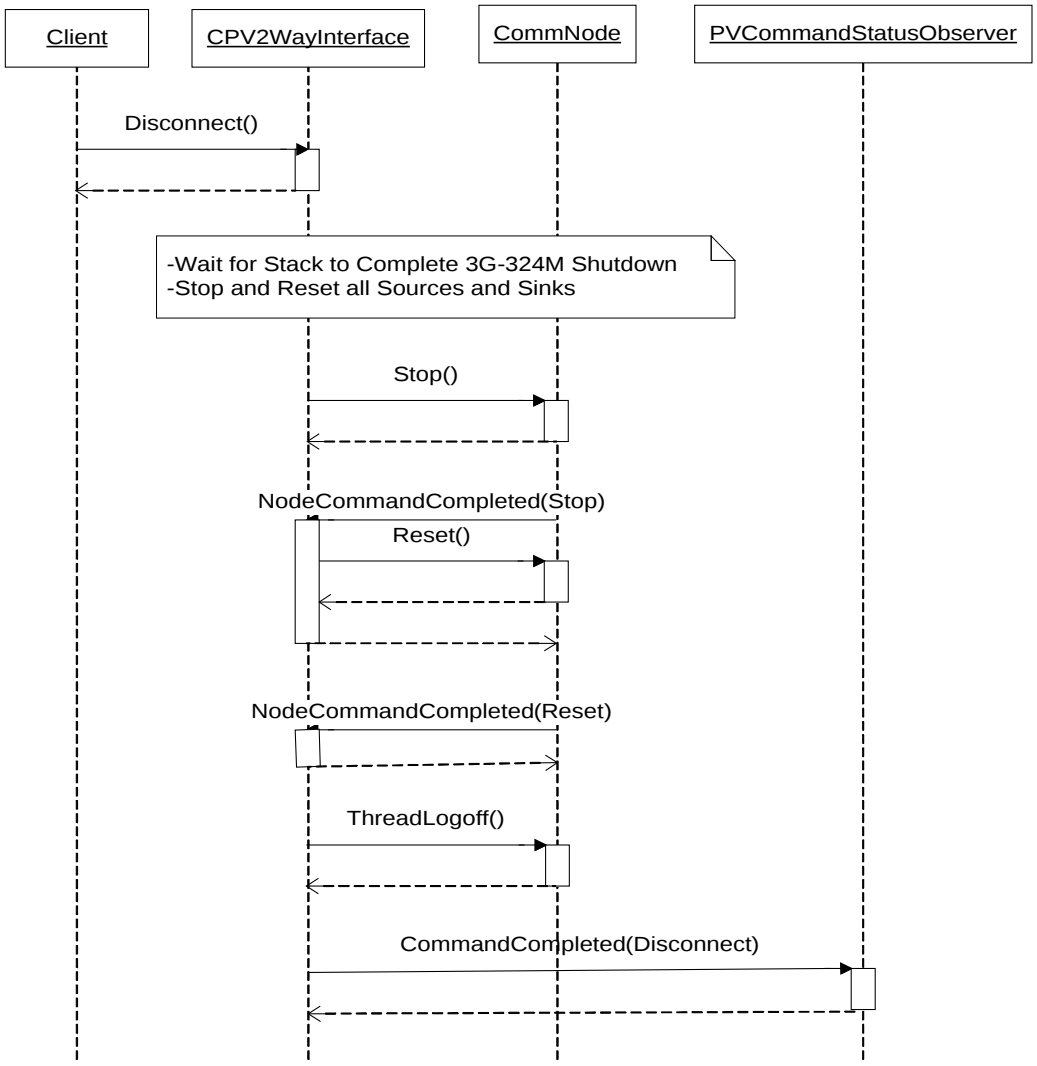

Figure 10: Sequence Diagram - Disconnecting

3.9. Resetting

Resetting the PV2Way Engine releases all resources acquired by the SDK. This command must be issued after Disconnect completes and prior to deleting the PV2Way instance.

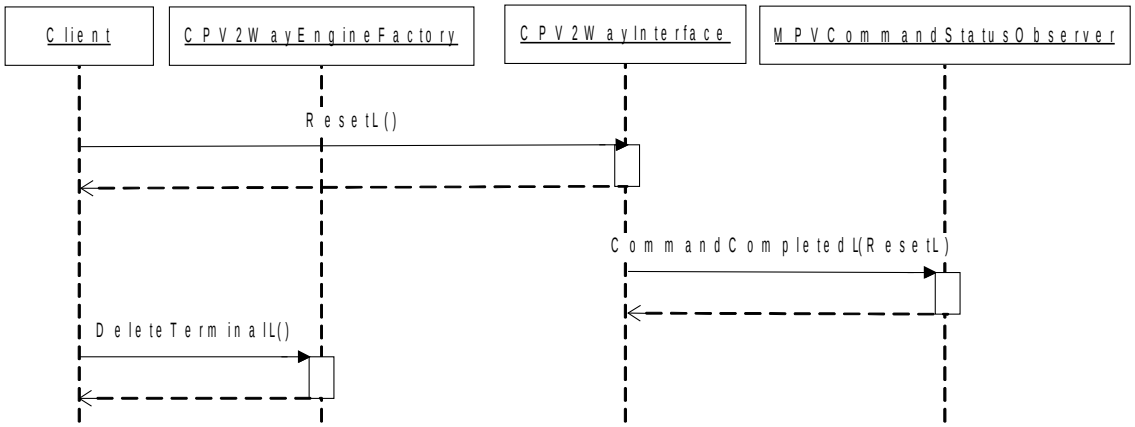

Figure 11: Sequence Diagram - Resetting

3.10. Complete Call Sequence

The following is the complete sequence of interactions that happen during a typical video telephony call.

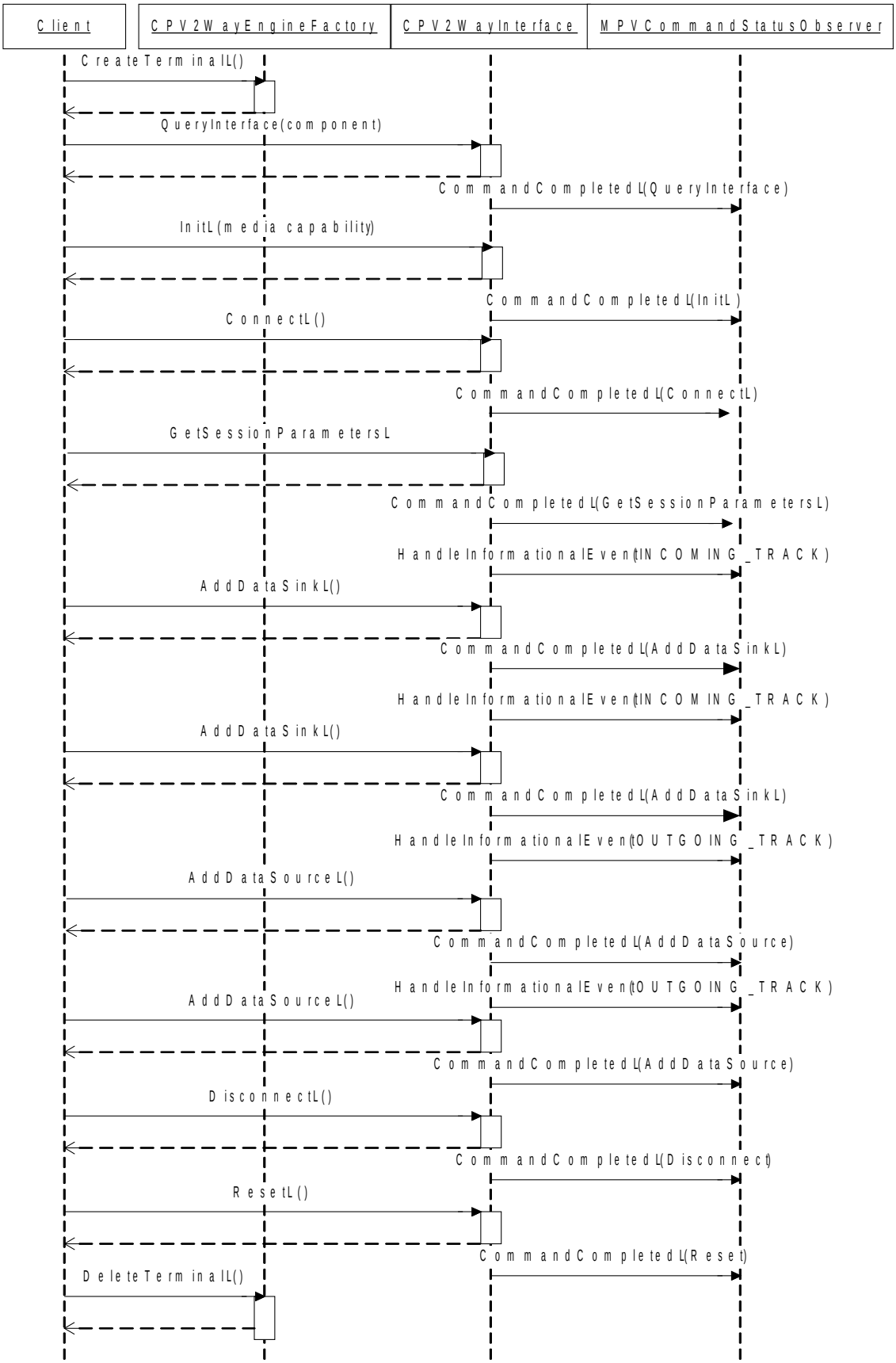

Figure 12: Sequence Diagram - Complete Call Sequence

4. Proxy Adapter

When building multimedia applications with their requisite complex timing requirements, it is often useful to allocate processing among multiple threads to get some degree of timing independence. Multithreading is a useful tool that simplifies the task of sharing CPU resources among multiple independent components. It is also possible to share processing among components within a single thread using concepts like active objects, which implement a cooperative multitasking model. Cooperative multitasking definitely has its place as another tool that should be utilized to allow complex systems to be built up from simpler components. Either tool on its own will not lead to a good solution for any reasonably complex system, so it's important to make proper use of each.

PV2Way has an optional Proxy Adapter that may be used to separate the client thread from the PV2Way thread. This separation can be used to ensure that the UI thread can remain responsive to user events and also for protection of time-critical processing within PV2Way. All callbacks (command completion, unsolicited events) will be made in the context of the thread that PV2Way was created. Sources and sinks passed in by the application should have the mechanism and capability to operate (data passing, state change) in a different thread from which it was created. The PV2Way Engine utilizes the Source/Sink ThreadLogon routines to notify external data sources and sinks of the thread contexts for data passing and observer callbacks.

The interface to the client does not change when the Proxy Adapter is in use. This is illustrated in Figure 1.

5. Logging

The 2Way Logging APIs provide methods for the application to control the logging level at any point in the logger tree using hierarchical tags to specify the control point. This will give the application complete control over the amount of log messages being produced by PV2Way, and by using the hierarchical tags, it provides very fine-grained control to turn up logging just where it is needed.
The application will be able to control the logging destinations by passing in Logger Appender instances that are created externally. Logger Appender classes are provided for most common simple cases such as logging to a file.

5.1. Logging incoming/outgoing audio/video data

Logging data coming in and going out of the PV2Way SDK proves to be useful at times especially in cases where there is large packet loss, data needs to be verified for timing issues, or audio & video data needs to be ascertained for their arrival times for lipsync purposes. An easy way of collecting logs for incoming and outgoing audio & video data is by the use of hierarchical logger. These tags provide the flexibility of restricting the number of log messages by categorizing them for incoming or outgoing audio, video or both kinds of data.

Some examples below typify their usage.

Logger tag: **datapath.outgoing**

This tag allows the user to log messages depicting all outgoing multiplexed data, be it audio or video at the stack level.

Following are the sample log messages printed/outputted:

PVLOG:TID(0x73c):Time=5656: Outgoing audio frames received. Stats: Entry time=2266, lcn=65543, size=254

PVLOG:TID(0x73c):Time=5656: Stats of the outgoing audio SDU are: timestamp=2266, size=31

PVLOG:TID(0x73c):Time=5657: Outgoing video frames received. Stats: Entry time=2286, lcn=65542, size=31

PVLOG:TID(0x73c):Time=5656: Stats of the outgoing video SDU are: timestamp=2286, size=254

Logger tag: **datapath.outgoing.video.h223.lcn**

This tag allows the user to only explicitly log outgoing multiplexed video data at the stack level. Following are the sample log messages printed/outputted:

PVLOG:TID(0x73c):Time=5657: Outgoing video frames received. Stats: Entry time=2286, lcn=65542, size=254

PVLOG:TID(0x73c):Time=5656: Stats of the outgoing video SDU are: timestamp=2286, size=254

Logger tag: **datapath.outgoing.audio.h223.lcn**

This tag allows the user to only explicitly log outgoing multiplexed audio data at the stack level.

Following are the sample log messages printed/outputted:

PVLOG:TID(0x73c): Time=5657: Outgoing audio frames received. Stats: Entry time=2266, lcn=65542, size=31

PVLOG:TID(0x73c):Time=5656: Stats of the outgoing audio SDU are: timestamp=2266, size=31

Logger tag: **datapath.incoming**

This tag allows the user to log all incoming de-multiplexed data be it audio or video at the stack level.

Following are the sample log messages printed/outputted:

PVLOG:TID(0xb98):Time=5016:Incoming audio SDU received. Stats: Entry time=2016, lcn=65542, size=31,FmtType=X-AMR-IF2

PVLOG:TID(0x73c):Time=6141:Incoming video SDU received. Stats: Entry time=2735, lcn=65543, size=254,FmtType=video/H263-2000

Logger tag: **datapath.incoming.video.h223.lcn**

This tag allows the user to only log incoming de-multiplexed video data at the stack level. Following are the sample log messages printed/outputted:

PVLOG:TID(0x73c):Time=6062:Incoming video SDU received. Stats: Entry time=2657, lcn=65543, size=254,FmtType=video/H263-2000

PVLOG:TID(0x73c):Time=6141:Incoming video SDU received. Stats: Entry time=2735, lcn=65543, size=254,FmtType=video/H263-2000

PVLOG:TID(0x73c):Time=6219:Incoming video SDU received. Stats: Entry time=2782, lcn=65543, size=254,FmtType=video/H263-2000

Logger tag: **datapath.incoming.audio.h223.lcn**

This tag allows the user to only log incoming de-multiplexed audio data at the stack level.

Following are the sample log messages printed/outputted:

PVLOG:TID(0xb98):Time=4969:Incoming audio SDU received. Stats: Entry time=1969, lcn=65542, size=31,FmtType=X-AMR-IF2
PVLOG:TID(0xb98):Time=5016:Incoming audio SDU received. Stats: Entry time=2016, lcn=65542, size=31,FmtType=X-AMR-IF2
PVLOG:TID(0xb98):Time=5063:Incoming audio SDU received. Stats: Entry time=2063, lcn=65542, size=31,FmtType=X-AMR-IF2
PVLOG:TID(0xb98):Time=5094:Incoming audio SDU received. Stats: Entry time=2094, lcn=65542, size=31,FmtType=X-AMR-IF2

6. Feature List

The specifications which define the 3G-324M standard present various mandatory and optional elements which may or may not be supported in a particular terminal implementation. The purpose of this section is to provide a feature-level summary of the elements supported by PV2Way 3G-324M solution. Features are classified by feature groups which relate to specific areas of the 3G-324M standard.

In general, features which are listed as supported may be used and/or configured through PV2Way API's, i.e. no special build time provisions should be necessary to activate such features. For other features (either absent from the list, or listed as "Not Supported"), some customization would be needed in order to add or utilize the feature.

6.1. Specification Versions

This section lists the 3G-324M system specifications and indicates the versions employed by PV2Way.

6.1.1. Normative

- 3GPP TS 26.110 v6.0.0: "Codec for Circuit Switched Multimedia Telephony Service; General description".
- 3GPP TS 26.111 v6.1.0: "Codec for Circuit Switched Multimedia Telephony Service; Modifications to H.324"
- ITU-T Recommendation H.324: "Terminal for low bitrate multimedia communication", February 1998.
- ITU-T Recommendation H.223: "Multiplexing protocol for low bit rate multimedia communication", March 1996.

- ITU-T Recommendation H.223 - Annex A: "Multiplexing protocol for low bit rate multimedia mobile communication over low error-prone channels", February 1998.

- ITU-T Recommendation H.223 - Annex B: "Multiplexing protocol for low bit rate multimedia mobile communication over moderate error-prone channels", February 1998.

- ITU-T Recommendation H.245, Version 6: "Control protocol for multimedia communication", February 2000.

6.1.2. Informative

- 3GPP TR 26.911 v6.0.0: "Codec for circuit switched multimedia telephony service; Terminal Implementor's Guide".

7. Supported Features

The features supported by PacketVideo's 3G-324M solution are broken down into logical feature groups and presented in the following tables.

7.1. Feature Group 1: H.223

Group	#	Feature	Support?	Direction	Comment
H.223	1	Level 0	Yes	In/Out	
H.223	2	Level 1 (H.223 Annex A)	Yes	In/Out	Support for Annex A in single or double flag modes.
H.223	3	Level 2 (H.223 Annex B)	Yes	In/Out	Support for Annex B with or without optional header.
H.223	4	Level 3a (H.223 Annex C)	No	--	
H.223	5	Level 3b (H.223 Annex D)	No	--	
H.223	6	Control over AL1	Yes	In/Out	Framed transfer mode, per H.324 recommendation
H.223	7	Audio over AL2	Yes	In/Out	CRC and Optional SN are both supported
H.223	8	Video over AL2	Yes	In/Out	CRC and Optional SN are both supported
H.223	9	Video over AL3	Yes	In/Out	Support for 0, 1 or 2 control field octets. Retransmission not supported (this is in line with the recommendation in 3GPP TS 26.911)

7.2. Feature Group 2: H.245

Group	#	Feature	Supported?	Direction	Comment
H.245	1	Terminal Capability Set	Yes	In/Out	
H.245	2	Master Slave Determination	Yes	In/Out	
H.245	3	Multiplex Entry Send	Yes	In/Out	
H.245	4	Request Multiplex Entry	Yes	In/Out	
H.245	5	Open Logical Channel	Yes	In/Out	For video case, both unidirectional and bidirectional are supported
H.245	6	Request Channel Close	Yes	In/Out	
H.245	7	End Session	Yes	In/Out	
H.245	8	Request Mode	No	--	Currently no outgoing Mode Requests are sent. Incoming Mode Requests are rejected, consistent for case with no Transmit Capabilities in outgoing TCS.
H.245	9	User Input Indication	Yes	In/Out	
H.245	10	Round Trip Delay	Yes	In/Out	
H.245	11	Flow Control	Yes	In/Out	H.245 signaling support only
H.245	12	Video Spatial/Temporal Tradeoff (Command)	Yes	In/Out	
H.245	13	Video Fast Update	Yes	In/Out	Picture update only
H.245	14	Multiplex Reconfiguration (level change)	No	--	
H.245	15	H.223 Skew Indication	Yes	In/Out	
H.245	16	Vendor Identification Indication	Yes	In/Out	
H.245	17	Mux PDU size restriction signaling	Yes	In/Out	As recommended in Section 5 of 3GPP TS 26.911.
H.245	18	Timer/Counter configuration	Yes	--	Various H.245 timer and counter values may be configured via API
H.245	19	Fast call setup	Yes	--	Optimal standards-compliant bundling of H.245 messages

					to reduce setup time.

7.3. Feature Group 3: H.324/3G-324M

Group	#	Feature	Supported?	Direction	Comment
H.324	1	H.324 Annex C	Yes	--	Mobile annex, as required in 3G-324M specs
H.324	2	SRP	Yes	In/Out	
H.324	3	NSRP	Yes	In/Out	
H.324	4	WNSRP	Yes	In/Out	
H.324	5	CCSRL	Yes	In/Out	
H.324	6	Audio Channels	Yes	In/Out	
H.324	7	Video Channels	Yes	In/Out	
H.324	8	Data Channels	No	--	
H.324	9	Loopback Mode	Yes	--	External loopback support provided for development and testing. Multiplex bitstream looped within terminal prior to transmission. H.245 Maintenance loop not currently supported.
H.324	10	Encryption	No	--	
H.324	11	Pause/Resume	Yes	In/Out	Option to stop sending A/V data on outgoing side, or to pause rendering of data on incoming side.
H.324	12	MONA-MPC	Optional	In/Out	Standards Based Enhancement for faster Call Setup.

7.4. Feature Group 4: Audio

Group	#	Feature	Supported?	Direction	Comment
Audio	1	AMR-NB	Yes	In/Out	All modes including SID
Audio	2	G.723.1	No	--	Not required for 3G-324M

7.5. Feature Group 5: Video

Group	#	Feature	Supported?	Direction	Comment
Video	1	H.263 baseline	Yes	In/Out	

		(Profile 0, Level 10)			
Video	2	MPEG-4 Simple Profile	Yes	In/Out	Level 0 is baseline support; Could support higher levels as well.
Video	3	H.261	No	--	Not required for 3G-324M
Video	4	H.263 additional annexes	No	--	Not required for 3G-324M
Video	5	MPEG-4 Resync Markers	Yes	In/Out	
Video	6	MPEG-4 HEC	Yes	In/Out	
Video	7	MPEG-4 Data Partitioning	Yes	In/Out	
Video	8	MPEG-4 RVLC's	Yes	In/Out	
Video	9	Error concealment	Yes	In	As recommended in 3GPP TS 26.911. Advanced error concealment provided by PacketVideo software decoders; this may or may not be available if a non-PV video decoder is used.

8. References

*1.*PacketVideo Corp. *PV2Way API Document*

2.ITU-T *H.324 Terminal For Low Bitrate Multimedia Communication*

3.PacketVideo Corp. *Media I/O Developers Guide*

4.PacketVideo Corp. *Guidelines for Developing Baseband Communications IO Components"*.