[MC-DPLVP]:
DirectPlay Voice Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.3	Minor	Updated the technical content.
11/30/2007	1.0	Major	Updated and revised the technical content.
01/25/2008	2.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

This document specifies the DirectPlay Voice Protocol Specification to the **DirectPlay Protocol**. This protocol is used to provide voice communications for applications that use the DirectPlay Protocol to communicate.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Globally Unique Identifier (GUID)
HRESULT

The following terms are defined in [MC-DPL4CS]:

Game Session
Host
Player
Player ID

The following terms are defined in [MC-DPL8CS]:

Client/Server Mode
Peer-to-Peer Mode

The following terms are specific to this document:

Codec: An algorithm for converting audio encoded in one binary form to another.

Compression ID: A **GUID** that indicates which **codec** is to be used for encoding and decoding speech data.

DirectPlay 4: The protocol specified by [MC-DPL4CS].

DirectPlay 8: The protocol specified by [MC-DPL8CS].

DirectPlay Client: When the underlying transport is **DirectPlay 4**, this is equivalent to a Player as specified in [MC-DPL4CS]. When the underlying transport is **DirectPlay 8**, this is equivalent to a peer when a peer-to-peer session or a Client, when a Client/Server session is running as specified in [MC-DPL8CS].

DirectPlay Client/Server Session: When the underlying transport is **DirectPlay 4**, this is a Game Session with the DPSESSION_CLIENTSERVER flag set in the Flags field of the DPSESSIONDESC2 field as specified in [MC-DPL4CS]. When the underlying transport is **DirectPlay 8**, this is equivalent to a Client/Server **DirectPlay Session** as specified in [MC-DPL8CS].

DirectPlay Host: When the underlying transport is **DirectPlay 4**, this is equivalent to the Host as specified in [MC-DPL4CS]. When the underlying transport is **DirectPlay 8**, this is equivalent to a Host when a peer-to peer session is running or a server when a Client/Server session is running as specified in [MC-DPL8CS].

DirectPlay Peer-to-Peer Session: When the underlying transport is **DirectPlay 4**, this is a Game Session without the DPSESSION_CLIENTSERVER flag set in the Flags field of the DPSESSIONDESC2 field as specified in [MC-DPL4CS]. When the underlying transport is **DirectPlay 8** this is equivalent to a peer-to-peer **DirectPlay Session** as specified in [MC-DPL8CS].

DirectPlay Protocol: Either the **DirectPlay 4** or **DirectPlay 8** protocol.

DirectPlay Protocol Voice Message Type: In **DirectPlay 4**, this is a DPSP_MSG_VOICE message as specified in [MC-DPL4CS]. In **DirectPlay 8**, this is a packet whose DFRAME has the PACKET_COMMAND_USER_2 flag set in the **bCommand** field as specified in [MC-DPL8R].

DirectPlay Session: When the underlying transport is **DirectPlay 4**, this is equivalent to a Game Session as specified in [MC-DPL4CS]. When the underlying transport is **DirectPlay 8**, this is equivalent to a **DirectPlay Session** as specified in [MC-DPL8CS].

DVID: A unique 32-bit identifier for an individual client instance or a group of client instances in the protocol. The unique identifier is equivalent to the Player ID of the client or group if the underlying transport is **DirectPlay 4**. If the underlying transport is **DirectPlay 8**, the identifier is equivalent to the DirectPlay Identifier of the client or group. The value of 0x00000000 is reserved to indicate all **voice clients** in the session.

Encoded Voice Stream: A stream of binary data representing speech encoded by a **codec**.

Host Migration: The algorithm and protocol as specified by the **Host Migration** Extension 1.3.3.1 used to elect a new **voice server** when the existing **voice server** exits the session.

Host Order ID: A monotonically increasing 32-bit identifier representing the priority of a **voice client** in a **host migration** election. The **voice server** assigns these to **voice clients** as they connect.

Jitter Buffer: A buffer that is used to re-order **speech message** voice data to reconstruct an **encoded voice stream** in time for playback. All **speech messages** are sent unordered and non-guaranteed in order to provide the lowest possible latency. As a result, **speech messages** are not guaranteed to arrive, may be duplicated, and are not guaranteed to arrive in order. When two packets are received out of order and there are three packets of **jitter buffer**, the packets can easily be correctly reordered before they are required for playback. If the packets are not correctly reordered in time, the audio is skipped.

Message Number: An 8-bit identifier that identifies which **voice burst** a **speech frame** belongs to. This number starts at zero and can wrap.

Sequence Number: An 8-bit identifier that specifies the location order of a **speech frame** within a **voice burst**, and which is used to reorder **speech frames** upon their receipt. The value of the number starts at zero, increases by one for each **speech frame** within the **voice burst**, and may wrap through reuse of older "low" values that are no longer in the system.

Note When wrapping occurs, the only values in the system will be near the high-end of the value space. For example, consider values between 0x00000000 and 0xFFFFFFFF. If the **sequence number** wraps, the number changes from 0xFFFFFFFF to 0x00000000; there are no values close to 0x00000000 and this value becomes reusable. When wrapping occurs, the client must understand that after the **sequence number** changes from 0x00000000 to 0xFFFFFFFF, the next value will be 0x00000000 again. In this manner, the receiver can perform guaranteed reordering.

Server Controlled Targeting: When a **voice server** controls the **voice target list** for the **voice clients**. This is enabled when the SessionFlags field of the Connect Accept Message (section 2.2.3.2) contains the DVSESSION_SERVERCONTROLTARGET (0x00000002) value.

Speech Frame: **Encoded voice streams** are broken into pieces each of which is called a **Speech Frame**. For this protocol, the size of each **speech frame** depends on the **codec** selected. This list of **codecs** and their frame sizes is specified in section 1.3.7.

Speech Message: A protocol message that contains at least a single **speech frame**, a **message number**, and a **sequence number**.

Voice Burst: Individual **speech frames** are grouped together into **voice bursts**. A **voice burst** contains a set of **speech frames** during which the timing is preserved by receiving clients. Gaps in the **voice burst** will be filled with silence to preserve the timing of the received packets. Timing is not guaranteed to be preserved between **voice bursts**. For more information about **voice bursts**, see section 1.3.1.

Voice Client: Any **DirectPlay Client** or a **DirectPlay Host** that understands the DirectPlay Voice Protocol and acts in a **Voice Client** role as specified in section 3.1.

Voice Server: The **DirectPlay client** that is responsible for coordinating the **voice session**. This **DirectPlay client** understands the DirectPlay Voice Protocol and acts in a **Voice Server** role as specified in section 3.2.

Voice Session: The collection of **DirectPlay clients** and the **DirectPlay Host** that are running either a **voice client** and/or a **voice server**.

Voice Session Subprotocol: The type of subprotocol used by the **voice server** once the connection subprotocol has completed. See section 1.3 for a complete list of **voice session subprotocols**.

Voice Session Type: The **DirectPlay Protocol**: DirectPlay Voice Extension supports four different **voice session types**. Each **voice session type** corresponds to a **voice session** protocol as specified in section 1.3.

Voice Target List: An array of **DVID** values each representing a user or group of users that the **speech messages** from a **voice client** are intended for.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MC-DPL4CS] Microsoft Corporation, "DirectPlay 4 Protocol: Core and Service Providers Specification" September 2007.

[MC-DPL8CS] Microsoft Corporation, "DirectPlay 8 Protocol: Core and Service Providers Specification" September 2007.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[WindowsAudioCodecs] Microsoft Corporation, "Compressing sound files", http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/compressing_sound_files.mspx?mfr=true

1.3 Protocol Overview (Synopsis)

The DirectPlay Voice Protocol enables **DirectPlay clients** within a **DirectPlay session** to communicate voice. The exchange is coordinated by a **voice server** and works independently of the version of DirectPlay in use. The protocol depends on the underlying DirectPlay session to handle connectivity and transport between the **voice clients** and the voice server.

The DirectPlay Voice Protocol provides a virtual session that exists within the DirectPlay session. It requires an existing DirectPlay session to operate but maintains its own session coordinated by the voice server. This means that not every DirectPlay client is required to participate. The voice server maintains the list of voice clients in the session and coordinates the distribution of this information.

The voice server and voice clients use the Connection Subprotocol (section 1.3.2) to establish initial connectivity, and then the voice server tells the voice clients which **voice session subprotocol** to use based on the **voice session type** determined at initialization of the voice server. The following table shows the list of voice session types and the corresponding **voice session** protocol that is used.

Voice session type	Voice session subprotocol
Peer session	Peer Voice Session Subprotocol (section 1.3.3)
Mixing session	Mixing Voice Session Subprotocol (section 1.3.4)
Forwarding session	Forwarding Voice Session Subprotocol (section 1.3.5)
Echo session	Echo Voice Session Subprotocol (section 1.3.6)

The DirectPlay Voice Protocol also provides an extension called the Host Migration Extension (section 1.3.3.1). This extension allows sessions running under the Peer Voice Session Subprotocol to elect a new voice server if the existing voice server becomes unavailable.

The upper layer records audio from the user and uses a **codec** to create an **encoded voice stream**. The encoded voice stream is broken into **voice bursts** and then broken further into **speech messages**. The exact methodology of distributing speech messages varies depending on the type of voice session subprotocol. For more information about voice bursts, see section 1.3.1.

Only a single codec is used for any given session. When a voice server is started, a codec is chosen to be used by that voice server. Any voice clients that wish to connect to the voice server MUST support the codec chosen by the voice server.

A list of codecs required to support the DirectPlay Voice Protocol is provided in section 1.3.7. For information on what voice clients should do when they do not support the codec chosen by the voice server, see section 3.1.5.1.3.

1.3.1 How DirectPlay Handles Voice Bursts

A voice burst occurs when DirectPlay detects voice. Essentially, a single voice burst translates to a "segment of voice recorded sequentially". Multiple voice bursts simply mean that after a burst there is silence. When voice data is transmitted again, that transmittal is the next voice burst.

For example, if you have the following:

1. Begin Recording
2. Two seconds of audio data
3. End Recording
4. One-second pause
5. Begin Recording
6. Three second of audio data
7. End Recording

The above scenario generates two voice bursts. The first voice burst is 2 seconds of audio time and the second is 3 seconds of audio. The timing within an audio segment is preserved. The 1 second between bursts (per the example above) is not preserved. As a result, the time between voice bursts may end up being 0.8 seconds, or 1.5 seconds, or some other length, depending on network conditions.

All audio data within a voice burst is continuous. Each voice burst is equivalent to a single, continuous buffer of audio. When no audible audio (silence) is recorded, no transmission is necessary. Silence represents the time "between" voice bursts. DirectPlay continues to record/send audio 400 ms after a user stops speaking. This allows for up to 400 ms of "silence" audio to be sent between words/sentences that the user speaks. This is not required in any way, but buffering in this manner allows for a more continuous transmission of voice data. The algorithm used to begin/end the recording of audio data can be determined by any developer writing to the DirectPlay Voice Protocol.

The DirectPlay Voice Protocol analyzes audio data to determine if it needs to be sent. Essentially, this is accomplished by checking to see if there is any audio data over a certain threshold. Determining when a voice burst needs to start is left to the implementation. A common way this is accomplished is to have the user press a button on the computer, such as "Press to Talk". The exact manner in which the start of the voice burst is determined is of no consequence to the DirectPlay Voice Protocol or the implementation. As long as an audio burst is a continuous stream of audio data, the data is considered to be in a voice burst. If two audio samples are appended when they are not sequential, this is not a true voice burst since the audio is not continuous.

A voice burst can be arbitrarily long, even indefinite, as determined by the implementation. However, the longer the voice burst, the more likely that audio data will be lost resulting in "pops" or "skips" in the playback.

1.3.2 Connection Subprotocol

The voice client and voice server use the connection subprotocol to establish connectivity and to communicate which voice session subprotocol and codec to use for further communications.

The following illustration shows the connection subprotocol message sequence for a successful connect attempt:

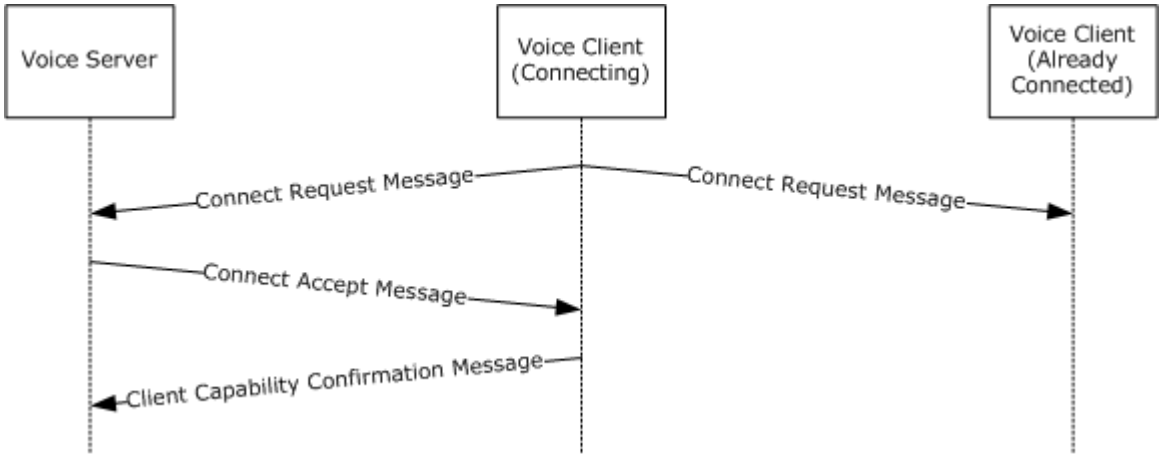


Figure 1: Connection protocol successful connect sequence

1.3.3 Peer Voice Session Subprotocol

The peer voice session subprotocol specifies the communication between a voice server and voice clients and between individual voice clients when a peer voice session type is used. The peer voice session subprotocol requires that the underlying DirectPlay Session is running in peer-to-peer mode. The peer voice session subprotocol is indicated when the **SessionType** field of the Connect Accept Message (section 2.2.3.2) is set to DVSESSIONTYPE_PEER (0x00000001).

The roles in this subprotocol are as follows:

- The voice server is responsible for maintaining and distributing the list of voice clients in the voice session.
- The voice clients are responsible for:
  - Sending speech messages directly to any voice clients that they want to send them to.
  - Receiving and processing speech messages from other voice clients and converting them back into an encoded voice stream. The voice client maintains a **jitter buffer** for each voice client in the voice client list.
  - Electing and creating a new voice server for voice sessions that use the **host migration** extension.

1.3.3.1 Host Migration Extension

The Host Migration Extension enables a set of voice clients to elect and create a new voice server to replace a voice server that has become unavailable.

A voice server can become unavailable either because connectivity is lost or the voice server has chosen to be shutdown. If the voice server chooses to shutdown, it informs all clients that a host migration needs to take place. The DirectPlay layer is responsible for determining when connectivity has been lost. In either case, the voice clients run a host migration election to determine who will create the new voice server.

Each voice client is assigned a **Host Order ID**, which indicates its priority in determining who should create the new voice server. The voice client that has the lowest Host Order ID is elected to create the new voice server. The lowest Host Order ID typically belongs to the oldest voice client in

the session, and is therefore, most likely to have the most up-to-date list of voice clients. This algorithm enables each voice client to run the algorithm separately. However, all come up with the same answer.

The elected voice client immediately creates the new voice server and initializes it by using the voice client list from the voice client. This new voice server informs everyone that it is ready to accept voice clients. Through the use of the Host Migration Extension, each voice client updates their Current Voice Server DVID (section 3.1.1) value to equal the **DVID** of the new voice server.

All the voice clients in the session send a message to the new voice server as soon as they complete the host migration election and again when they receive notification from the new voice server that it is ready to accept voice clients. This ensures that the list of voice clients on the new voice server is up to date.

The following illustration shows the typical message flow of a successful host migration when the voice server has chosen to shutdown:

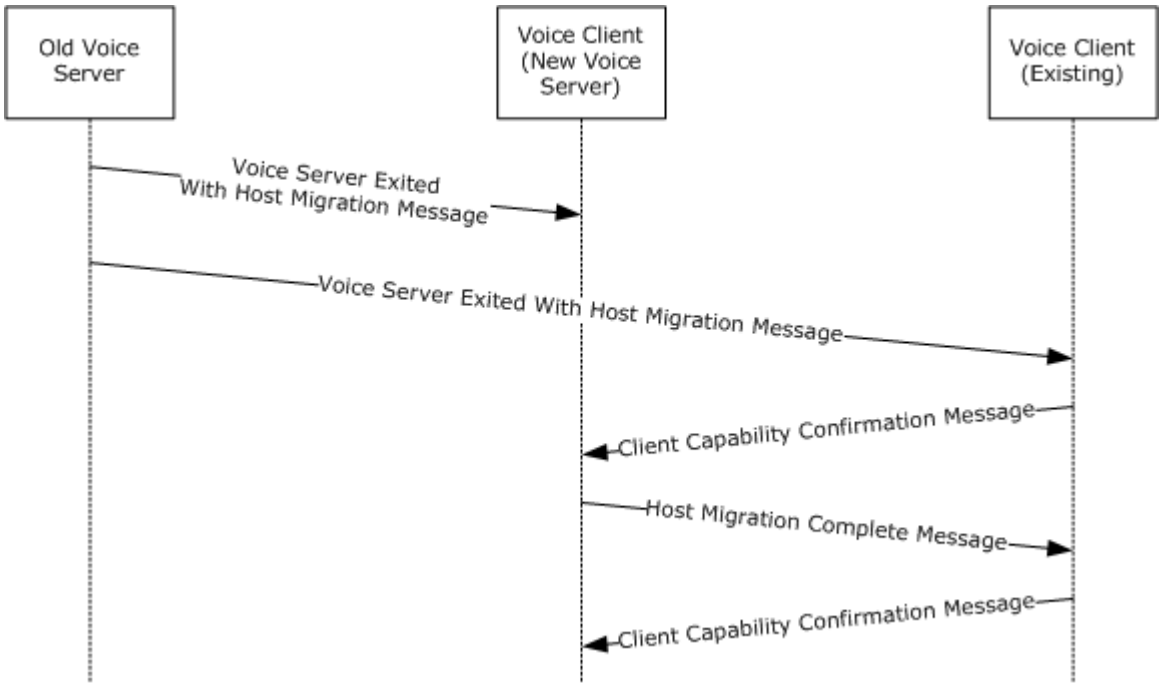


Figure 2: Successful host migration sequence

The extension needs to be enabled when the protocol is initialized. Voice clients determine that the host migration extension is in use if the Peer Voice Session Subprotocol is being used and the **SessionFlags** field in the Connect Accept Message (section 2.2.3.2) does not contain the DVSESSION_NOHOSTMIGRATION (0x00000001) flag.

1.3.4 Mixing Voice Session Subprotocol

The mixing voice session subprotocol specifies the communication between a voice server and voice clients when a mixing session type is used. The mixing voice session subprotocol requires a DirectPlay session that provides connectivity between the voice server and the voice clients. Connectivity between individual voice clients is not required. The mixing voice session subprotocol is

indicated when the **SessionType** field of the Connect Accept Message (Section 2.2.3.2) is set to DVSESSIONTYPE_MIXING (0x00000002).

The roles in this subprotocol are as follows:

- The voice server is responsible for the following:
  - Maintaining the list of voice clients in the voice session.
  - Receiving speech messages from each of the voice clients and distributing them to each of the voice clients for which they are intended. The voice server distributes the speech messages in a single encoded voice stream to each voice client by mixing together all speech messages intended for each individual voice client. The voice server maintains a jitter buffer for each voice client in the voice client list.
- The voice clients are responsible for the following:
  - Sending voice data and the targets of the data to the voice server.
  - Receiving and processing speech messages from the voice server and converting them back into an encoded voice stream. The voice client maintains a single jitter buffer for incoming speech messages from the voice server.

  Note The voice clients do not maintain a list of voice clients.

1.3.5 Forwarding Voice Session Subprotocol

The forwarding voice session subprotocol specifies the communication between a voice server and voice clients when a forwarding session type is used. The forwarding voice session subprotocol requires at a minimum a **DirectPlay client-server session**, but it will also work with a **DirectPlay peer-to-peer session**. Connectivity between individual voice clients is not required. The forwarding voice session subprotocol is indicated when the **SessionType** field of the Connect Accept Message (Section 2.2.3.2) is set to DVSESSIONTYPE_FORWARDING (0x00000003).

The roles in this subprotocol are as follows:

- The voice server is responsible for:
  - Maintaining the list of voice clients in the voice session.
  - Receiving speech messages from each of the voice clients and forwarding them to each of the voice clients for which they are intended. Speech messages intended for each individual voice client.
- The voice clients are responsible for:
  - Sending voice data and the targets of the data to the voice server.
  - Receiving and processing speech messages from the voice server and converting them back into an encoded voice stream. The voice client maintains a jitter buffer for each voice client in the voice client list.

  Note The voice clients do not maintain a list of voice clients.

1.3.6 Echo Voice Session Subprotocol

The echo voice session subprotocol specifies the communication between a voice server and voice clients when an echo session type is used. The echo voice session subprotocol requires a DirectPlay client-server session but it will also operate with a DirectPlay peer-to-peer session. The forwarding voice session subprotocol is indicated when the **SessionType** field of the Connect Accept Message (section 2.2.3.2) is set to DVSESSIONTYPE_ECHO (0x00000004).

The roles in this subprotocol are as follows:

- The voice server is responsible for maintaining a list of voice clients and forwarding any incoming speech messages back to the sender.
- The voice clients are responsible for:
  - Sending voice data to the server.
  - Receiving and processing speech messages from the voice server and converting them back into an encoded voice stream. The voice client maintains a single jitter buffer for incoming speech messages from the voice server.

Note The voice clients do not maintain a list of voice clients.

1.3.7 Required Codecs

The DirectPlay Voice Protocol requires communication using one of the codecs listed in the following table. The codecs are opaque to the DirectPlay Voice Protocol with the exception that in a voice session, the voice client and voice server need to use the same codec. The following table lists the codecs, the frame size used with the protocol, and the approximate amount of time (in milliseconds) that each frame represents.

Required Codec	Frame size (bytes)	Frame size (millisecond of data)	Approximate bitrate (bits/s)
Voxware VR12 (*)	Variable (Max 21)	90	Variable (Max 1822)
Voxware SC03 (*)	40	100	3200
Voxware SC06 (*)	80	100	6400
Truespeech (*)	96	90	8536
Global System for Mobile Communications (GSM) (*)	130	80	13000
Microsoft Adaptive Delta Pulse Code Modulation (MS ADPCM) (**)	256	63	32768
Pulse Code Modulation (PCM) - 8 Khz, 8-bit, and Mono (*)	394	50	64000

(*) = This codec is licensed through a third-party.

(**) = Microsoft Adaptive Delta Pulse Code Modulation (MS ADPCM) is included with the Microsoft Windows Development Kit (WDK). For more information about the MS ADPCM codec, see [WindowsAudioCodecs].

1.4 Relationship to Other Protocols

The DirectPlay Protocol: DirectPlay Voice Extension is embedded in either the **DirectPlay 4** or **DirectPlay 8** protocols. The **DirectPlay Protocol Voice Message Type** is used for all DirectPlay Protocol: DirectPlay Voice Extensions.

1.5 Prerequisites/Preconditions

The DirectPlay Voice Protocol operates only after a DirectPlay session is established. If the DirectPlay session is terminated then the DirectPlay Voice Protocol Specification is also terminated.

There are further restrictions on the Voice Session type based on the DirectPlay session type. The following table illustrates the restrictions.

Session type	DirectPlay peer-to-peer session	DirectPlay client/server session
Peer Voice Session	Supported	Not Supported
Mixing Voice Session	Supported	Supported
Forwarding Voice Session	Supported	Supported
Echo Voice Session	Supported	Supported

1.6 Applicability Statement

The DirectPlay Voice Protocol is designed to provide voice communications between voice clients within a DirectPlay session.

The following table describes the characteristics of each session type based on latency of voice traffic, CPU usage of the voice clients and the voice server, and the bandwidth usage of the server and clients.

Voice Session Type	Latency (Voice Traffic)	CPU Usage (Voice Clients)	CPU Usage (Voice Server)	Bandwidth (Voice Clients)	Bandwidth (Voice Server)
Peer Voice Session	Lowest	Scales with number of people talking to a voice client	Fixed	Scales with number of people talking to voice client. Also scales by number of people the voice client is talking to.	Fixed
Mixing Voice Session	Highest	Fixed	Scales with number of unique mixes and number of people talking.	Fixed	Scales with number of people receiving voice.
Forwarding Voice Session	Medium	Fixed	Fixed	Scales with number of people talking to the voice client.	Scales with number of people talking.
Echo Voice Session	Medium	Fixed	Fixed	Fixed	Scales with number of people talking.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports**: This protocol can be implemented on top of DirectPlay 4 and DirectPlay 8 protocols.
- **Supported Codecs**: This protocol supports multiple codecs for encoding voice data into encoded voice streams. The connection subprotocol is used to inform clients which codec they must use through the Connect Accept Message (section 2.2.3.2).
- **Capability Negotiation**: This voice server decides which voice session subprotocol, which codec, and what session characteristics will be used for the communications. This information is communicated to clients through the connection subprotocol in the Connect Accept Message (section 2.2.3.2).

The DirectPlay Protocol: DirectPlay Voice Extension provides version fields in the connection subprotocol but they are not used.

1.8 Vendor-Extensible Fields

This protocol uses **HRESULT** values as specified in [MS-ERREF] section 2.1. Vendors can define their own HRESULT values, provided they set the C bit (0x20000000) for each vendor-defined value, indicating the value is a customer code.

1.9 Standards Assignments

This protocol contains no standards assignments.

2 Messages

The following sections specify how DirectPlay Voice Protocol messages are transported and DirectPlay Voice Protocol message syntax.

2.1 Transport

This protocol is designed to operate over the DirectPlay 4 or DirectPlay 8 protocols using the voice message type.

2.2 Message Syntax

The following sections contain DirectPlay Voice Protocol message syntax. All numeric values are transported in little-endian format.

The following table indicates which messages are used for each session type subprotocol.

Message type	Connection subprotocol	Peer voice session subprotocol	Mixing voice session subprotocol	Forwarding voice session subprotocol	Echo voice session subprotocol
Add Voice Client Message (section 2.2.2.4)	No	Yes	Yes	No	No
Remove Voice Client Message (section 2.2.4.3)	No	Yes	No	No	No
Session Lost Message (section 2.2.2.1)	No	Yes	Yes	Yes	Yes
Host Migration Complete Message (section 2.2.4.5.2)	No	Yes (*)	No	No	No
Set Client Voice Target Message (section 2.2.2.5)	No	Yes (**)	Yes (**)	Yes (**)	Yes (**)
Connect Request Message (section 2.2.3.1)	Yes	No	No	No	No
Connect Refuse Message (section 2.2.3.3)	Yes	No	No	No	No

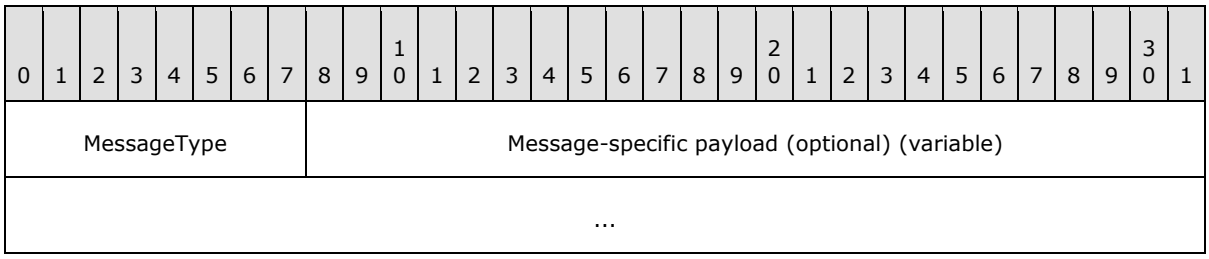
Message type	Connection subprotocol	Peer voice session subprotocol	Mixing voice session subprotocol	Forwarding voice session subprotocol	Echo voice session subprotocol
Client Disconnect Request Message (section 2.2.2.2)	No	Yes	Yes	Yes	Yes
Speech Message (section 2.2.4.4)	No	Yes	No	No	No
Connect Accept Message (section 2.2.3.2)	Yes	No	No	No	No
Client Capability Confirmation Message (section 2.2.2.8)	Yes	Yes (*)	No	No	No
Client Disconnect Confirmation Message (section 2.2.2.3)	No	Yes	Yes	Yes	Yes
Speech With Bounce Message (section 2.2.2.7)	No	No	Yes	No	Yes
Voice Client List Message (section 2.2.4.2)	No	Yes	No	No	No
Voice Server Exited With Host Migration Message (section 2.2.4.5.1)	No	Yes (*)	No	No	No
Speech With Target Message (section 2.2.2.6)	No	No	Yes	Yes	No
Speech With From Message (section 2.2.5.1)	No	No	No	Yes	No

(*) = Only when the Host Migration extension is enabled. For additional information, see Host Migration Extension (section 1.3.3.1) .

(**) = Only when **server controlled targeting** is enabled. This is indicated by the value DVSESSION_SERVERCONTROLTARGET (0x00000002) being present in the **SessionFlags** field in the Connect Accept Message (section 2.2.3.2).

2.2.1 The Common Message Header

All messages in the DirectPlay Voice Protocol share a common header, which is followed by a message-specific payload, as specified in the sections below. Some message types do not have any message-specific payload.



MessageType (1 byte): An 8-bit unsigned integer representing a unique packet type that identifies the message. **MessageType** MUST be one of the following values.

Value	Meaning
DVMSGID_CREATEVOICEPLAYER 0x01	Add Voice Client Message (section 2.2.2.4)
DVMSGID_DELETEVOICEPLAYER 0x02	Remove Voice Client Message (section 2.2.4.3)
DVMSGID_SESSIONLOST 0x03	Session Lost Message (section 2.2.2.1)
DVMSGID_HOSTMIGRATED 0x0C	Host Migration Complete Message (section 2.2.4.5.2)
DVMSGID_SETTARGETS 0x0D	Set Client Voice Target Message (section 2.2.2.5)
DVMSGID_CONNECTREQUEST 0x51	Connect Request Message (section 2.2.3.1)
DVMSGID_CONNECTREFUSE 0x53	Connect Refuse Message (section 2.2.3.3)
DVMSGID_DISCONNECT 0x54	Client Disconnect Request Message (section 2.2.2.2)
DVMSGID_SPEECH 0x55	Speech Message (section 2.2.4.4)
DVMSGID_CONNECTACCEPT 0x56	Connect Accept Message (section 2.2.3.2)

Value	Meaning
DVMSGID_SETTINGSCONFIRM 0x58	Client Capability Confirmation Message (section 2.2.2.8)
DVMSGID_DISCONNECTCONFIRM 0x5A	Client Disconnect Confirmation Message (section 2.2.2.3)
DVMSGID_SPEECHBOUNCE 0x60	Speech With Bounce Message (section 2.2.2.7)
DVMSGID_PLAYERLIST 0x61	Voice Client List Message (section 2.2.4.2)
DVMSGID_HOSTMIGRATELEAVE 0x62	Voice Server Exited With Host Migration Message (section 2.2.4.5.1)
DVMSGID_SPEECHWITHTARGET 0x63	Speech With Target Message (section 2.2.2.6)
DVMSGID_SPEECHWITHFROM 0x64	Speech With From Message (section 2.2.5.1)

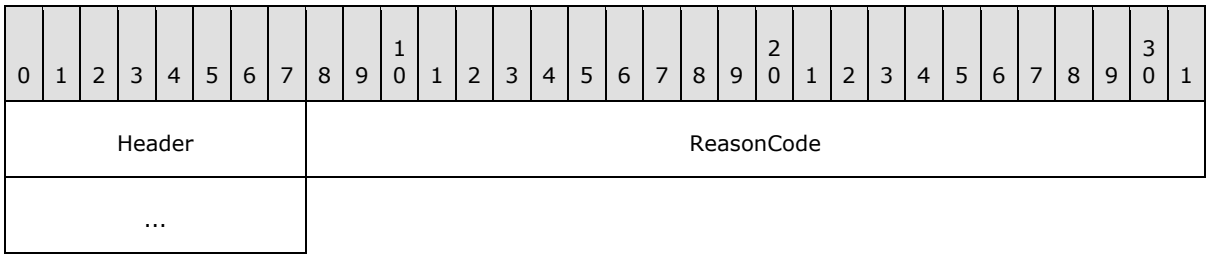
Message-specific payload (optional) (variable): A variable length field the size of which depends on the type of packet designated in the **MessageType** field.

2.2.2 Common Messages

The following are messages that are applicable to more than one session type subprotocol or the connection subprotocol.

2.2.2.1 Session Lost Message

The voice server sends this message to the voice client in all voice session subprotocols to indicate that the session has ended. This message can also be sent when the new voice server fails to initialize during host migration. These messages are transmitted as ordered and guaranteed with the DirectPlay protocol voice message type.

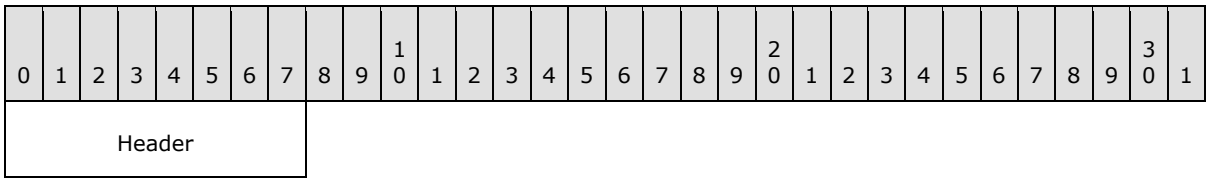


Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SESSIONLOST (0x03).

ReasonCode (4 bytes): This MUST be DVERR_SESSIONLOST (0x8015012C).

2.2.2.2 Client Disconnect Request Message

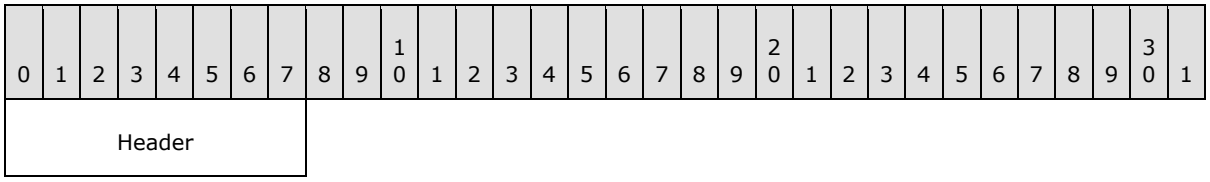
This message is sent by the voice client in all voice session subprotocols when they are exiting the voice session gracefully. These messages are transmitted as ordered and guaranteed with the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_DISCONNECT (0x54).

2.2.2.3 Client Disconnect Confirmation Message

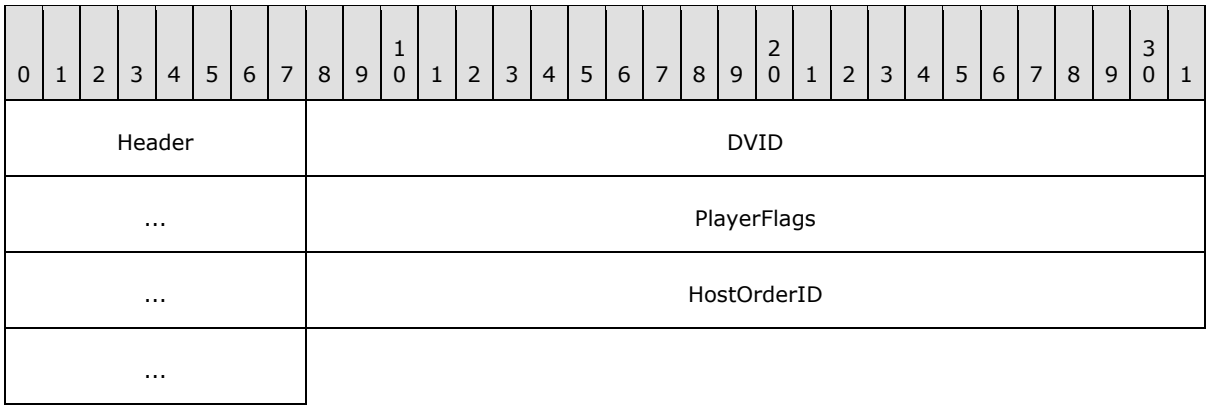
This message is sent from the voice server to the voice client in all voice session subprotocols when the voice server has successfully removed a voice client from the voice session. These messages are transmitted as ordered and guaranteed with the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_DISCONNECTCONFIRM (0x5A).

2.2.2.4 Add Voice Client Message

In the peer voice session subprotocol (section 1.3.3) this message is sent from the voice host to all voice clients in the session to instruct them to add a voice client to their list of voice clients. In the mixing voice session subprotocol (section 1.3.4) this message is sent from the voice host to the voice client to indicate that the voice client has successfully joined the voice session. These messages are transmitted as ordered, guaranteed messages using the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_CREATEVOICEPLAYER (x01).

DVID (4 bytes): A 32-bit unsigned integer containing the **DVID** of the voice client that is to be added.

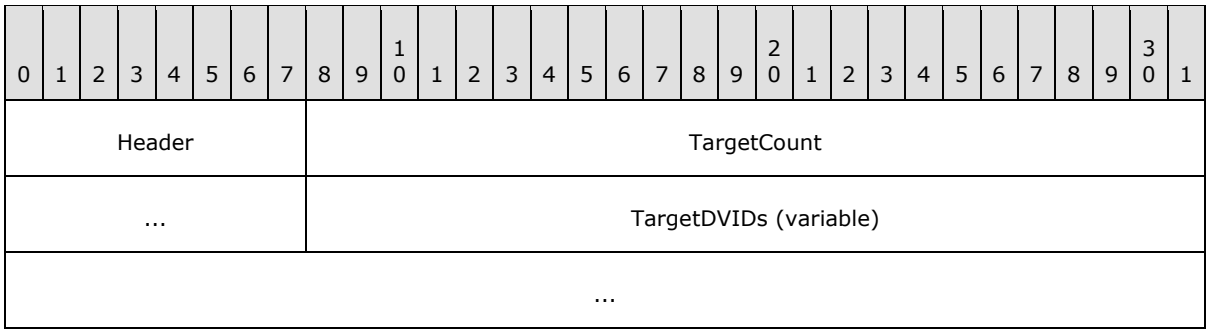
PlayerFlags (4 bytes): A 32-bit unsigned integer. This represents a set of bit flags representing voice client audio capabilities. This field MUST be composed of the bitwise OR of zero or more of the following values:

Value	Meaning
DVPLAYERCAPS_HALFDUPLEX 0x00000001	The voice client cannot record audio and will therefore not be transmitting any speech messages.

HostOrderID (4 bytes): A 32-bit unsigned integer representing the host order ID of the voice client. When the Host Migration Extension (section 1.3.3.1) is enabled, this field MUST be set to the host order ID of the voice client to be added. Otherwise, **HostOrderID** SHOULD be set to 0xFFFFFFFF.

2.2.2.5 Set Client Voice Target Message

This message is sent from the voice server to a voice client in all voice session subprotocols to update the voice client's**voice target list**, and is only used when server-controlled targeting is enabled. This message is sent guaranteed and ordered by the DirectPlay protocol voice message type.



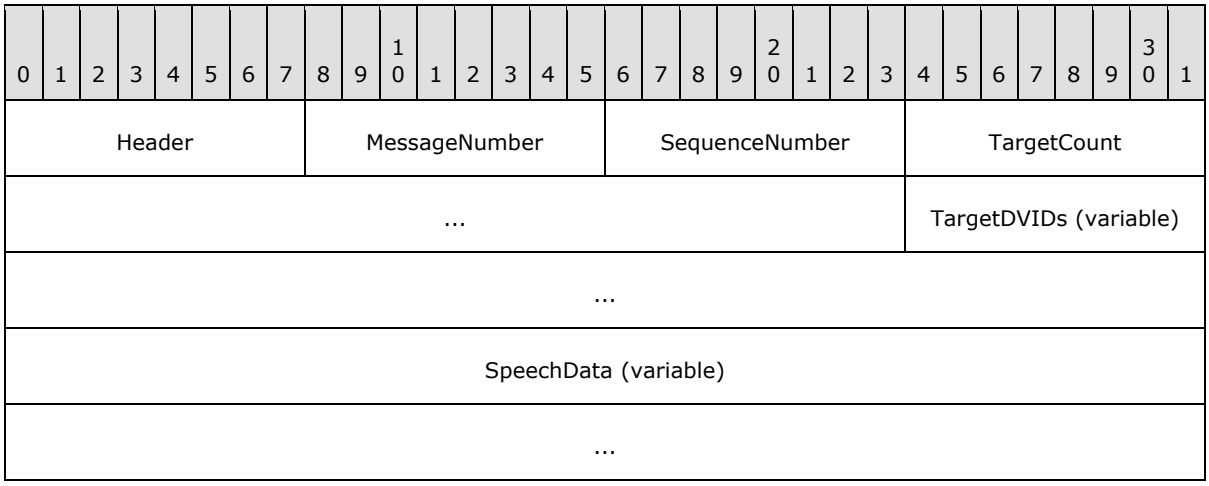
Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SETTARGETS (0x0D).

TargetCount (4 bytes): A 32-bit unsigned integer. This value MUST contain the length of the **TargetDVIDs** field in DVID elements, which MUST be greater than or equal to zero. The field MUST be less than or equal to 64.

TargetDVIDs (variable): An optional array of DVID values each representing a user or group of users that the speech data is intended for. The number of DVIDs in this array MUST be equal to the value of the **TargetCount** field. This field will not contain any data if **TargetCount** is zero. This field cannot contain duplicate DVID values.

2.2.2.6 Speech with Target Message

When using the Mixing voice session subprotocol (section 1.3.4) or the Forwarding voice session subprotocol (section 1.3.5) this speech message is sent from the voice client to the voice server. The message contains the required fields of a speech message, in addition to the list of DVIDs it is intended for. This message is sent unordered and non-guaranteed using the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SPEECHWITHTARGET (0x63).

MessageNumber (1 byte): An 8-bit unsigned integer representing this speech message's **Message Number**.

SequenceNumber (1 byte): An 8-bit unsigned integer representing this speech message's **Sequence Number**.

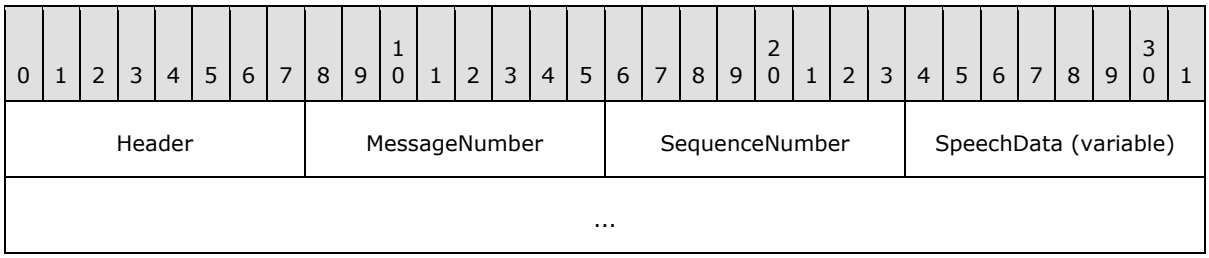
TargetCount (4 bytes): A 32-bit unsigned integer. This value MUST contain the length of the **TargetDVIDs** array in DVID elements. This field MUST be greater than zero and less than or equal to 64.

TargetDVIDs (variable): An array of DVID values each representing a user or group of users that the speech data is intended for. The number of DVIDs in this array MUST be equal to the value of the **TargetCount** field and cannot contain duplicates.

SpeechData (variable): An array of bytes containing a **speech frame** encoded in the currently selected codec.

2.2.2.7 Speech with Bounce Message

This message is used in both the Echo voice session subprotocol (section 1.3.6) and the Mixing voice session subprotocol (section 1.3.4). When used in the echo voice session subprotocol it is used to send a single frame of audio received from the voice client back to the same voice client. It contains the minimum fields necessary for a speech message. When used in the mixing voice session subprotocol it is used to transmit a single packet of mixed audio to a voice client. This message is sent unordered and non-guaranteed by using the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SPEECHBOUNCE (0x60).

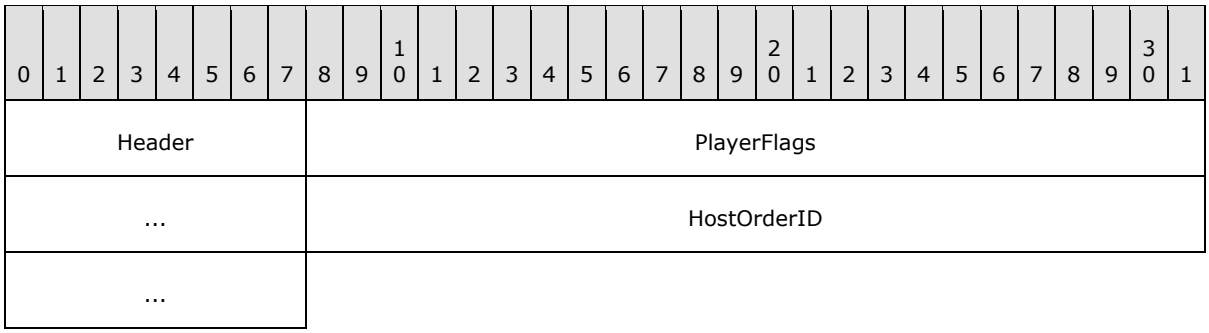
MessageNumber (1 byte): An 8-bit unsigned integer representing this speech message's Message Number.

SequenceNumber (1 byte): An 8-bit unsigned integer representing this speech message's Sequence Number.

SpeechData (variable): An array of bytes containing a speech frame encoded in the currently selected codec.

2.2.2.8 Client Capability Confirmation Message

This message is used by the connection subprotocol (section 1.3.2), and the host migration extension to the peer voice session subprotocol (section 1.3.3). For the connection subprotocol, it is sent by the voice client to complete capability negotiation. For the host migration extension, it is used by voice clients to communicate capabilities to the new host when a host migration is complete. This message is sent as an ordered, guaranteed message using the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SETTINGSCONFIRM (0x58).

PlayerFlags (4 bytes): A 32-bit unsigned integer. This represents a set of bit flags representing client audio capabilities. This field MUST be composed of the bitwise OR of zero or more of the following values:

Value	Meaning
DVPLAYERCAPS_HALFDUPLEX 0x00000001	The voice client cannot record audio and will therefore not be transmitting any encoded audio data.

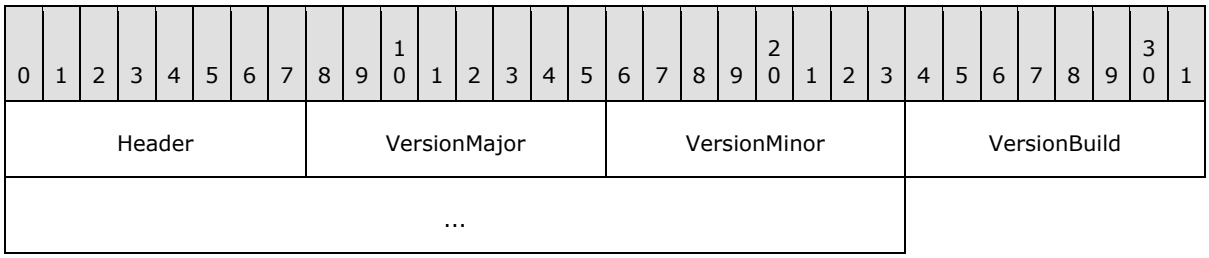
HostOrderID (4 bytes): A 32-bit unsigned integer representing the host order ID of the voice client. When the Host Migration Extension (section 1.3.3.1) is enabled, and the Client Capability Confirmation Message is used during host migration, the **HostOrderID** field MUST be set to the **Current Host Order ID** of the voice client. Otherwise, **HostOrderID** MUST be set to 0xFFFFFFFF. For more information about the **Current Host Order ID**, see sections 3.1.1 and 3.1.3.

2.2.3 Connection Subprotocol Messages

The following messages are used only in the connection subprotocol.

2.2.3.1 Connect Request Message

When the underlying DirectPlay session is peer-to-peer, this message is sent to all DirectPlay clients. When the underlying DirectPlay session is client-server, this message is sent to the **DirectPlay Host**. This message is sent as an ordered, guaranteed message by using the DirectPlay protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_CONNECTREQUEST (0x51).

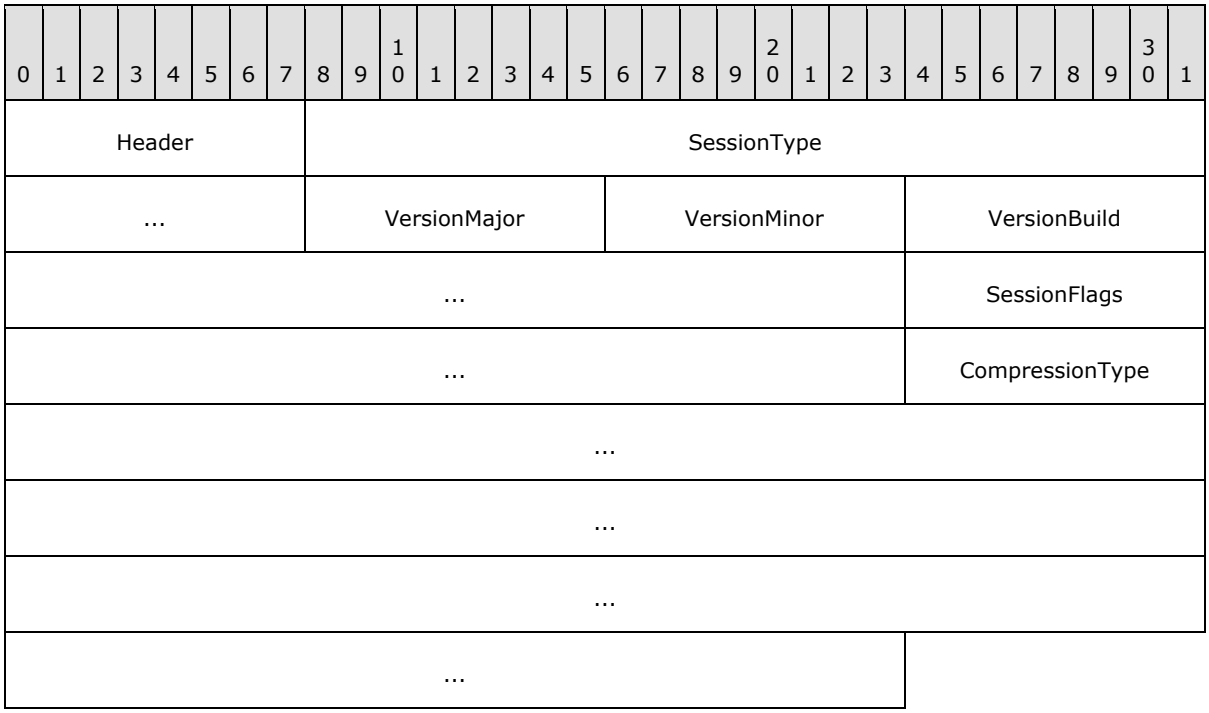
VersionMajor (1 byte): An 8-bit unsigned integer. This value MUST be 0x01.

VersionMinor (1 byte): An 8-bit unsigned integer. This value MUST be 0x00.

VersionBuild (4 bytes): A 32-bit unsigned integer. This value MUST be 0x00000003.

2.2.3.2 Connect Accept Message

This message is sent by the voice server to inform the voice client of configuration of the session. This message is sent as an ordered and guaranteed message by using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_CONNECTACCEPT (0x56).

SessionType (4 bytes): A 32-bit unsigned integer. This value MUST be one of the following. It indicates which type of voice session is in use and, therefore, which voice session subprotocol to use when communicating for this voice session.

Value	Meaning
DVSESSIONTYPE_PEER 0x00000001	The peer-to-peer voice session type. Use the Peer Voice Session Subprotocol (section 1.3.3).
DVSESSIONTYPE_MIXING 0x00000002	The mixing voice session type. Use the Mixing Voice Session Subprotocol (section 1.3.4).
DVSESSIONTYPE_FORWARDING 0x00000003	The forwarding voice session type. Use the Forwarding Voice Session Subprotocol (section 1.3.5).
DVSESSIONTYPE_ECHO 0x00000004	The echo voice session type. Use the Echo Voice Session Subprotocol (section 1.3.6).

VersionMajor (1 byte): An 8-bit, unsigned integer. This value MUST be 0x01.

VersionMinor (1 byte): An 8-bit, unsigned integer. This value MUST be 0x00.

VersionBuild (4 bytes): A 32-bit, unsigned integer. This value MUST be 0x00000003.

SessionFlags (4 bytes): A 32-bit, unsigned integer representing a set of bit flags that indicate the session flags for the voice session. This field SHOULD be composed of the bitwise OR of zero or more of the following:

Value	Meaning
DVSESSION_NOHOSTMIGRATION 0x00000001	The Host Migration Extension (section 1.3.3.1) is not available in the current voice session. If the voice server leaves the session, the session will end.
DVSESSION_SERVERCONTROLTARGET 0x00000002	The targets of a voice client's voice data is controlled by the voice server instead of the voice client itself.

CompressionType (16 bytes): A 16-byte **globally unique identifier (GUID)** value indicating the **compression ID** for the codec that MUST be used for encoding voice data in this voice session. For a full description of the codecs and required parameters, see Required Codecs (section 1.3.7).

Note When the client receives the Connect Accept Message, it should check its local capabilities to ensure that it can support the specified codec. If it cannot support the specified codec, then the client will not send any more messages and MUST not reply with a Connect Capability Confirmation Message.

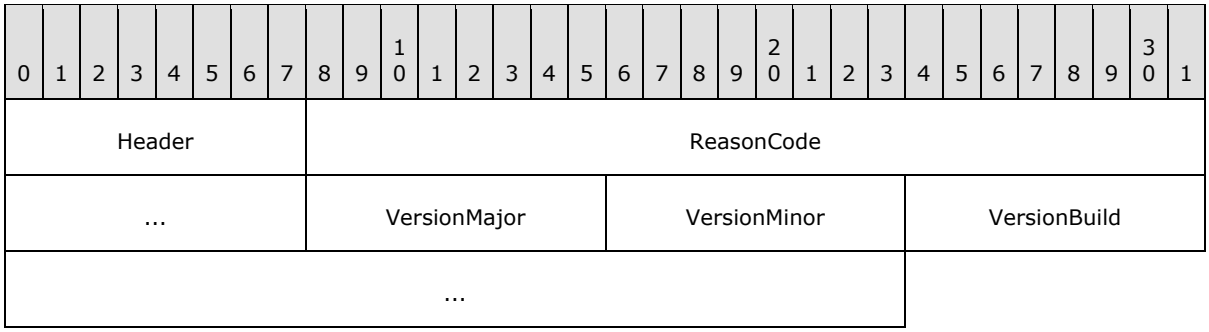
This MUST be one of the following values:

Value	Meaning
DPVCTGUID_ADPCM {699B52C1-A885-46a8-A308-97172419ADC7}	Microsoft Adaptive Delta Pulse Code Modulation (MS-ADPCM)
DPVCTGUID_GSM {24768C60-5A0D-11d3-9BE4-525400D985E7}	Global System for Mobile Communications (GSM)
DPVCTGUID_NONE {8DE12FD4-7CB3-48ce-A7E8-9C47A22E8AC5}	Pulse Code Modulation (PCM)
DPVCTGUID_SC03 {7D82A29B-2242-4f82-8F39-5D1153DF3E41}	Voxware SC03
DPVCTGUID_SC06 {53DEF900-7168-4633-B47F-D143916A13C7}	Voxware SC06
DPVCTGUID_TRUESPEECH {D7954361-5A0B-11d3-9BE4-525400D985E7}	Truespeech
DPVCTGUID_VR12	Voxware VR12

Value	Meaning
{FE44A9FE-8ED4-48bf-9D66-1B1ADFF9FF6D}	

2.2.3.3 Connect Refuse Message

This message is sent by the voice server if a voice client requests to connect and the voice server is either shutting down or not yet initialized. This message is sent as an ordered and guaranteed message by using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_CONNECTREFUSE (0x53).

ReasonCode (4 bytes): A HRESULT value indicating the reason the connection request was refused. This field MUST be set to DVERR_NOTHOSTING (0x8015017B).

VersionMajor (1 byte): An 8-bit unsigned integer. This value MUST be 0x01.

VersionMinor (1 byte): An 8-bit unsigned integer. This value MUST be 0x00.

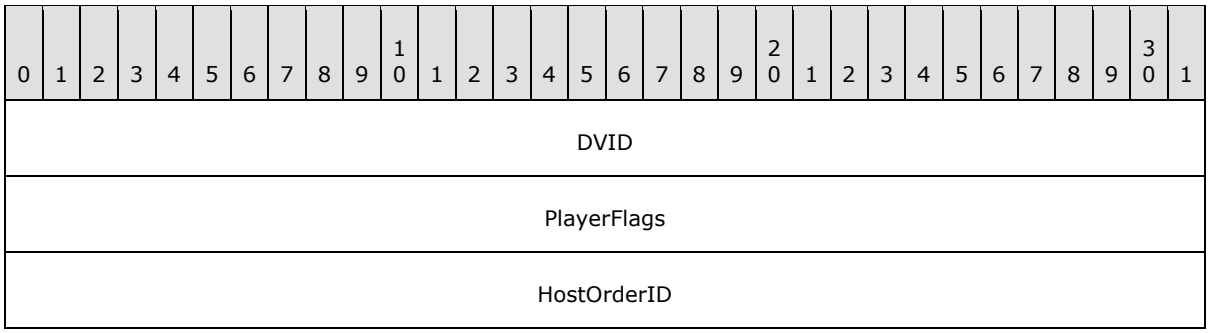
VersionBuild (4 bytes): A 32-bit unsigned integer. This value MUST be 0x00000003.

2.2.4 Peer Voice Session Subprotocol Messages

The following are messages that are specific to the Peer Voice Session Subprotocol (section 1.3.3).

2.2.4.1 Voice Client List Entry Structure

This structure is used to describe a single voice client.



DVID (4 bytes): A 32-bit unsigned integer representing the DVID of the voice client.

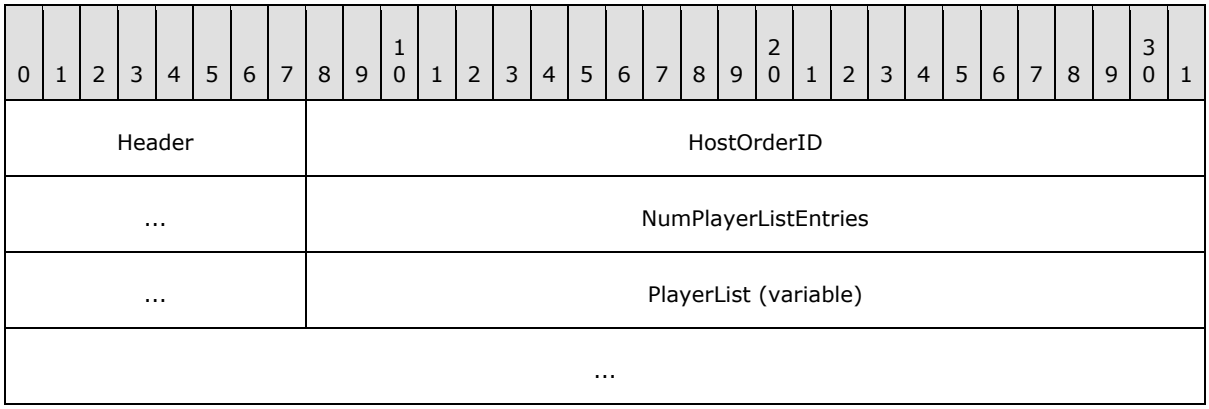
PlayerFlags (4 bytes): A 32-bit unsigned integer. This represents a set of bit flags representing client audio capabilities. This field MUST be composed of the bitwise OR of zero or more of the following values:

Value	Meaning
DVPLAYERCAPS_HALFDUPLEX 0x00000001	The voice client cannot record audio and will therefore not be transmitting any encoded audio data.

HostOrderID (4 bytes): A 32-bit unsigned integer representing the host order ID of the voice client. When the Host Migration Extension (section 1.3.3.1) is enabled, this field MUST be set to the host order ID of the voice client that is being described. Otherwise, **HostOrderID** SHOULD be set to 0xFFFFFFFF.

2.2.4.2 Voice Client List Message

This message is sent from the voice server to a voice client to provide that client with a list of all voice clients that are currently part of the voice session. If the session is large, then the server may send more than one of these messages. The message is sent as an ordered, guaranteed message using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_PLAYERLIST (0x61).

HostOrderID (4 bytes): A 32-bit unsigned integer representing the host order ID of the voice client. When the Host Migration Extension (section 1.3.3.1) is enabled, this field MUST be set

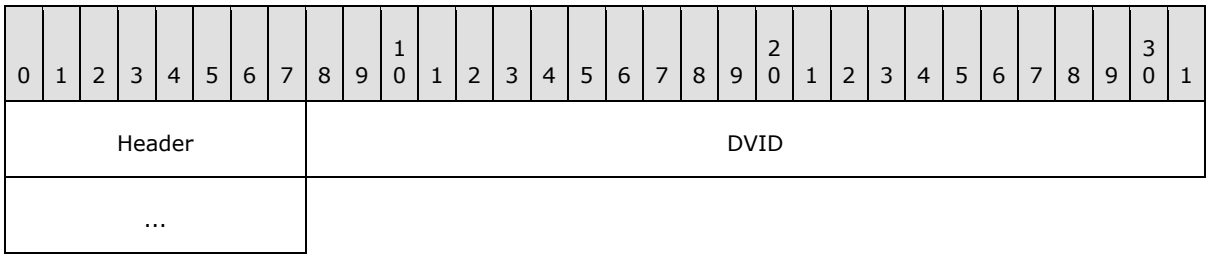
to the host order ID of the voice client to which the Voice Client List Message is being sent. Otherwise, **HostOrderID** SHOULD be set to 0xFFFFFFFF.

NumPlayerListEntries (4 bytes): A 32-bit unsigned integer. This value MUST indicate the length of the **PlayerList** field in Voice Client List Entry structure. This field will never be a value greater than 0x00000052. It is possible for this value to be zero.

PlayerList (variable): An optional array of Voice Client List Entry (section 2.2.4.1) structures. The number of structures present MUST equal the value of **NumPlayerListEntries**. If **NumPlayerListEntries** is zero, then there will be no data in this portion of the message.

2.2.4.3 Remove Voice Client Message

This message is sent from the voice server to a voice client to instruct them to remove a specific voice client from their list of voice clients. This message is sent ordered and guaranteed by using the DirectPlay Protocol voice message type.

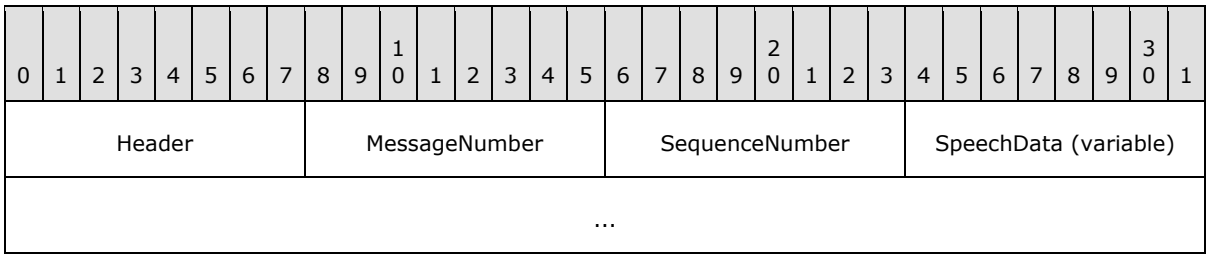


Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_DELETEVOICEPLAYER (0x02).

DVID (4 bytes): A 32-bit unsigned integer representing the DVID of the voice client to remove.

2.2.4.4 Speech Message

This message is sent from one voice client to another to send a single speech frame. It contains the minimum fields needed for a speech message. This message is sent unordered and non-guaranteed by using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SPEECH (0x55).

MessageNumber (1 byte): An 8-bit unsigned integer representing this speech message's message number.

SequenceNumber (1 byte): An 8-bit unsigned integer representing this speech message's sequence number.

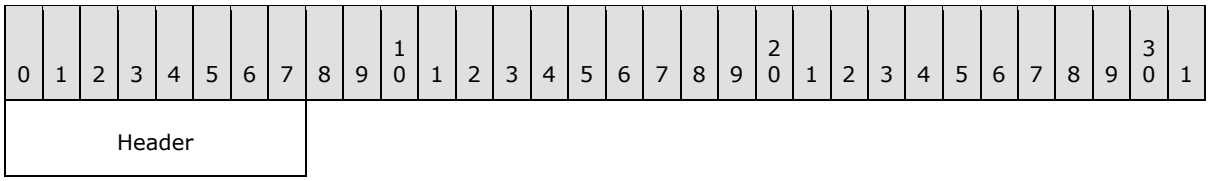
SpeechData (variable): An array of bytes containing a speech frame encoded in the currently selected codec.

2.2.4.5 Host Migration Extension Messages

The following are messages which are specific to the Host Migration Extension (section 1.3.3.1) of the Peer Voice Session Subprotocol (section 1.3.3).

2.2.4.5.1 Voice Server Exited With Host Migration Message

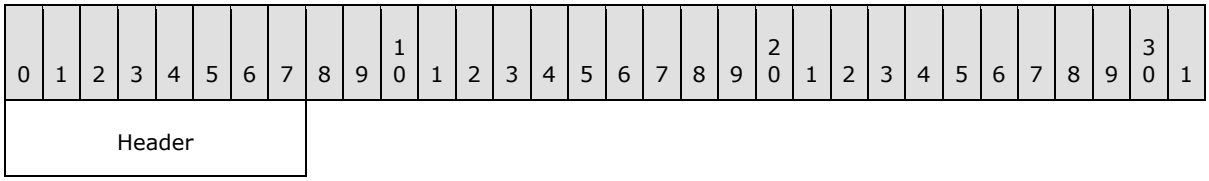
This message is sent by the voice server when it is exiting the voice session and host migration is available. The message is sent ordered and guaranteed by using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_HOSTMIGRATELEAVE (0x62).

2.2.4.5.2 Host Migration Complete Message

When a host migration occurs, this message is sent when a new voice server has been successfully created and initialized. This message is sent by the new voice server to all of the voice clients in the voice session. This message is sent ordered and guaranteed by using the DirectPlay Protocol voice message type.



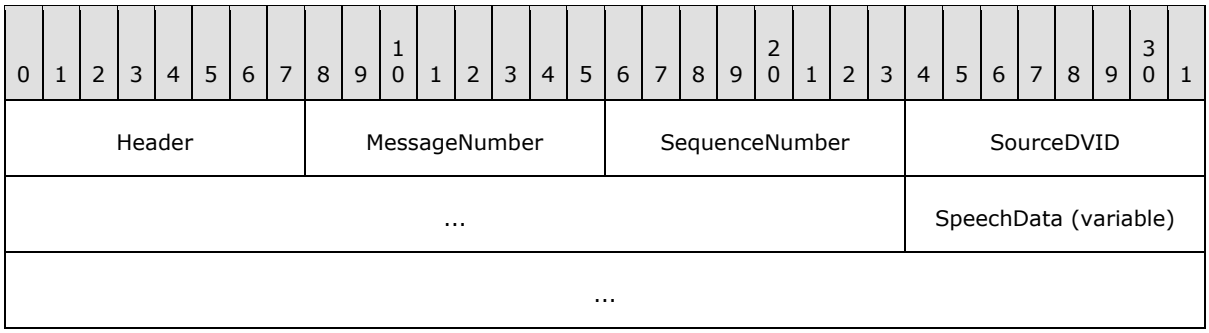
Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_HOSTMIGRATED (0x0C).

2.2.5 Forwarding Voice Session Subprotocol Messages

The following are messages that are specific to the Forwarding Voice Session Subprotocol (section 1.3.5).

2.2.5.1 Speech With From Message

This message is sent from the voice server to a voice client to relay a speech message on behalf of the voice client. It contains the minimum fields needed for a speech message as well as a DVID identifying the original voice client who sent the message. This message is sent unordered and non-guaranteed by using the DirectPlay Protocol voice message type.



Header (1 byte): The common message header (as specified in section 2.2.1). The **MessageType** field MUST be set to DVMSGID_SPEECHWITHFROM (0x64).

MessageNumber (1 byte): An 8-bit unsigned integer representing this speech message's message number.

SequenceNumber (1 byte): An 8-bit unsigned integer representing this speech message's sequence number.

SourceDVID (4 bytes): A 32-bit unsigned integer representing the DVID of the original voice client who sent the message.

SpeechData (variable): An array of bytes containing a speech frame encoded in the currently selected codec.

3 Protocol Details

The following sections specify details of the DirectPlay Voice Protocol, including abstract data models, interface method syntax, and message processing rules.

3.1 Voice Client Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The following are abstract data items maintained by voice clients:

Voice Client List: When the voice client is operating with the Peer Voice Session Subprotocol (section 1.3.3) it maintains a list of other voice clients in the voice session. The voice client list contains the DVID and host order ID of each voice client in the session. Voice Clients are added and removed from the list in response to messages from the Voice Server during connection and throughout the lifetime of the session.

Current Host Order ID: When the Host Migration Extension (section 1.3.3.1) is active the voice client stores what its host order ID is.

Current Voice Server DVID: The DVID of the current voice server. This may change if a host migration occurs.

Current Voice Target List: The voice client maintains a voice target list. This list is used to target speech messages sent from the voice client. If the voice target list is empty then speech messages are discarded instead of sent.

Jitter Buffers: When the voice client is using the Peer Voice Session Subprotocol (section 1.3.3) or the Forwarding Voice Session Subprotocol (section 1.3.5) it maintains a jitter buffer for each voice client in the voice client list. When the voice client is using any of the other session subprotocols, the voice client maintains a single jitter buffer for incoming speech messages from the voice server.

Current Message Number: The message number of the current voice burst. When a new voice burst begins this should be incremented.

Current Sequence Number: The sequence number of the next speech message being sent within the current voice burst. This value should be reset to zero when a new voice burst starts.

3.1.2 Timers

The **Connection Request Timer** is created when the voice client begins its connect sequence. It fires every 1250ms and expires after 30000ms or when the voice client completes a successful connection with the Connection Subprotocol (section 1.3.2).

The **Speech Data Transmission Timer** fires on the period indicated by the currently selected codec as described by the Frame Size (ms of data) column in the list of codecs specified in section 1.3.7.

3.1.3 Initialization

The underlying DirectPlay session must be established before the voice client can begin operation. In addition, the following MUST occur:

- The **Voice Client List** is set to empty.
- **Current Host Order ID** is set to 0xFFFFFFFF.
- **Current Voice Server DVID** is set to zero if the DirectPlay Session is peer-to-peer or the DVID of the voice server if the DirectPlay Session is client-server.
- **Current Voice Target List** is set to empty.
- **Jitter Buffers** are initialized.
- **Current Message Number** is set to zero.
- **Current Sequence Number** is set to zero.

3.1.4 Higher-Layer Triggered Events

The following are events which can be triggered by a higher level:

Disconnect Request: The higher level can initiate a disconnection from the voice session. This will cause a Client Disconnect Confirm Message (section 2.2.2.3) to be sent to the voice server. This will also halt the transmission of speech messages.

Start Voice Burst: The higher level can start a voice burst. When a voice burst is started, speech messages are transmitted until the voice burst is stopped. When a voice burst is started, the message number is incremented by one and the sequence number is reset to zero. If the **Current Voice Target List** is empty, starting a voice burst has no effect.

Stop Voice Burst: The higher level can stop a voice burst. When a voice burst is stopped, speech messages stop transmitting and the sequence number is no longer incremented.

Change Voice Client List: The higher level can modify the **Current Voice Target List**. If the voice target list is emptied by the higher level and a voice burst is being transmitted, the voice burst will stop transmitting.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Connection Subprotocol

3.1.5.1.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.1.5.1.2 Sending Connect Request Message

The structure and fields of the Connect Request Message are specified in section 2.2.3.1. This message is handled differently depending on the underlying DirectPlay Session Type:

Peer-to-Peer: The message is sent to all DirectPlay clients.

Client/Server: This message is sent to the DirectPlay session host.

The client MUST send this message before any other messages.

3.1.5.1.3 Receiving Connect Accept Message

The structure and fields of the Connect Accept Message are specified in section 2.2.3.2.

This MUST be the first message a voice client receives from a voice server. The client MUST only receive this message after it has successfully sent a Connect Request Message. The client will only process the first Connect Accept Message it receives, it will ignore any others it receives.

When the client receives this message it should check its local capabilities to ensure that it can support the specified codec. If it cannot support the specified codec, then the client will not send any more messages. If the client can support the specified codec, it should reply with a Connect Capability Confirmation Message.

The voice client SHOULD update the value of its **Current Voice Server DVID** to match the DVID of the sender of the Connect Accept Message. The DVID value is determined differently depending on the DirectPlay Protocol version.<1>

3.1.5.1.4 Receiving Connect Refuse Message

The structure and fields of the Connect Refuse Message are specified in section 2.2.3.3.

This message MUST be the first message a voice client receives from a voice server. The voice client MUST only receive this message after it has successfully sent a Connect Request Message. The client will only process the first Connect Refuse Message it receives and will ignore any others it receives.

When the client receives this message it should stop communicating. The **ReasonCode** field from the Connect Refuse Message can optionally be passed up to higher layers to indicate the cause of the failure.

3.1.5.1.5 Sending a Client Capability Confirmation Message

The structure and fields of the Client Capability Confirmation Message are specified in section 2.2.2.8.

This message MUST be sent only after a Connect Accept Message has been received. After sending this message, the voice client will start using the voice session subprotocol specified in the **SessionType** field of the Connect Accept Message. In the Client Capability Confirmation Message, the **HostOrderID** field MUST be set to 0xFFFFFFFF.

3.1.5.2 Common Messages for All Voice Session Subprotocols

3.1.5.2.1 Receiving a Session Lost Message

The structure and fields of the Session Lost Message are specified in section 2.2.2.1. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

This message indicates that the voice session has ended. The voice client SHOULD ignore all messages after this message is received and stop sending messages. If the voice client is currently awaiting a Client Disconnect Confirmation Message, this message will stop waiting.

3.1.5.2.2 Receiving a Set Client Voice Target Message

The structure and fields of the Set Client Voice Target Message are specified in section 2.2.2.5. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

This message is sent by the voice server to the voice client to update the **Current Voice Target List**. The message should only be processed if the session is running Server Controlled Targeting.

When the message is processed the voice client should replace their **Current Voice Target List** with the list of voice targets in this message. All speech messages transmitted after this point should use the new set of targets.

3.1.5.2.3 Sending a Client Disconnect Request Message

The structure and fields of the Client Disconnect Request Message are specified in section 2.2.2.2.

This message is sent by the voice client to the voice server when the higher level indicates that the voice client should disconnect. When this message is sent, the voice client stops any currently transmitting voice bursts. The voice client then waits for a Client Disconnect Confirmation Message. If a Session Lost Message or a Client Disconnect Confirmation Message is received while waiting, the voice client stops waiting. If a Host Migration Complete Message is received when the Host Migration Extension (as specified in section 1.3.3.1) is active while waiting, then the voice client will retransmit the Client Disconnect Request Message.

3.1.5.2.4 Receiving a Client Disconnect Confirmation Message

The structure and fields of the Client Disconnect Confirmation Message are specified in section 2.2.2.3. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

This message indicates that the current voice server has received the client's request to disconnect. When this message is received, the voice client should cease all communications.

3.1.5.2.4.1 Receiving an Add Voice Client Message

The structure and fields of the Add Voice Client Message are specified in section 2.2.2.4. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

The client should add the voice client described by the Add Voice Client Message into its **Voice Client List**. Duplicate entries in the **Voice Client List** should be ignored.

If the voice session is running the Mixing Voice Session Subprotocol (section 1.3.4), this message is also used to signal the voice client that they can start transmitting speech messages if a voice burst is active and there is a non-empty **Current Voice Target List**.

3.1.5.3 Peer Voice Session Subprotocol

The Peer Voice Session Subprotocol (section 1.3.3) is used by the voice client only after it has received a Connect Accept Message and responded with a Client Capability Confirmation Message. The Peer Voice Session Subprotocol is used when the **SessionType** field of the Connect Accept Message is set to DVSESSIONTYPE_PEER (0x00000001).

3.1.5.3.1 Handling Unrecognized Message

If an unrecognized or malformed message is received it is ignored.

3.1.5.3.2 Receiving a Voice Client List Message

The structure and fields of the Voice Client List Message are specified in section 2.2.4.2. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

The client should iterate through the list of voice clients in the Voice Client List Message and add each of them to the **Voice Client List**. Duplicate entries in the **Voice Client List** should be ignored.

3.1.5.3.3 Receiving a Remove Voice Client Message

The structure and fields of the Remove Voice Client Message are specified in section 2.2.4.3. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

The voice client should remove the voice client identified by the Remove Voice Client Message from its **Voice Client List**. If the specified voice client is already removed this message will be safely ignored.

3.1.5.3.4 Sending a Speech Message

The structure and fields of the Speech Message are specified in section 2.2.4.4.

When the **Speech Data Transmission Timer** fires, if a voice burst is active, the next speech message is sent. The Speech Message is sent directly by the voice client to the voice clients in the **Current Voice Target List**. Speech Message should not be sent before the Voice Client List Message is received. Speech messages should not be sent after the Client Disconnect Request Message is sent to the voice server.

3.1.5.3.5 Receiving a Speech Message

The structure and fields of the Speech Message are specified in section 2.2.4.4.

A Speech Message is directly received from other voice clients. The speech message SHOULD be placed in the Jitter Buffer of the voice client it came from. The voice client SHOULD ignore Speech messages from Voice Clients which are not yet present in the Voice Client List.

3.1.5.3.6 Host Migration Extension

3.1.5.3.6.1 Receiving a Voice Server Exited With Host Migration Message

The structure and fields of the **Voice Server Exited with Host Migration Message** are specified in section 2.2.4.5.1.

This message indicates that the Voice Server is exiting the session and host migration is available in the voice session. See Host Migration Extension for information on what the voice client should do in response to the message.

3.1.5.3.6.2 Receiving a Host Migration Complete Message

The structure and fields of the **Host Migration Complete Message** are specified in section 2.2.4.5.2. This message MUST come from the current voice server that is identified by the **Current Voice Server DVID.**

This message indicates that the voice server has exited the session, a new voice server has been elected, and it has successfully initialized the new voice server. The voice client MUST respond to this message with a Client Capability Confirmation Message sent to the new voice server. If this message is received from a client who is not the newly elected host it is ignored.

3.1.5.3.6.3 Sending a Client Capability Confirmation Message

The structure and fields of the Client Capability Confirmation Message are specified in section 2.2.2.8. This message MUST be sent to the current voice server that is identified by the **Current Voice Server DVID**.

The voice client will send a Client Capability Confirmation Message to the new voice server when either a Host Migration Complete Message (section 2.2.4.5.2) or a Voice Server Exited with Host Migration Message (section 2.2.4.5.1) is received. The Client Capability Confirmation Message MUST have the **HostOrderID** field set to the **Current Host Order ID** of the voice client.

3.1.5.4 Mixing Voice Session Subprotocol

3.1.5.4.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.1.5.4.2 Receiving a Speech with Bounce Message

The structure and fields of the Speech with Bounce Message are specified in section 2.2.2.7. This message MUST come from the current voice server, which is identified by the Current Voice Server DVID. The voice client MUST pass the message to the jitter buffer.

3.1.5.4.3 Sending a Speech with Target Message

The structure and fields of the Speech with Target Message are specified in section 2.2.2.6. This message MUST be sent to the current voice server that is identified by the Current Voice Server DVID.

When the **Speech Data Transmission Timer** fires if a voice burst is active the next Speech with Target Message is sent. The Speech with Target Message is sent directly by the voice client to the voice server and the clients in the **Current Voice Target List**. Speech Messages should not be sent before the Client Capability Confirmation Message is sent. Speech with Target Message should not be sent after the Client Disconnect Request Message is sent to the voice server.

3.1.5.5 Forwarding Voice Session Subprotocol

3.1.5.5.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.1.5.5.2 Receiving a Speech With From Message

The structure and fields of the Speech with From Message are specified in section 2.2.5.1. This message MUST come from the current voice server that is identified by the Current Voice Server DVID.

The voice client should take the Speech with From Message and put it into the Jitter Buffer for the voice client identified in the **SourceDVID** field.

3.1.5.5.3 Sending a Speech With Target Message

The structure and fields of the Speech with Target Message are specified in section 2.2.2.6. This message MUST be sent to the current voice server that is identified by the Current Voice Server DVID.

When the **Speech Data Transmission Timer** fires, if a voice burst is active, the next Speech with Target Message is sent. The Speech with Target Message is sent directly by the voice client to the voice server and the clients in the **Current Voice Target List**. Speech Messages should not be sent before the Client Capability Confirmation Message is sent. Speech with Target Message should not be sent after the Client Disconnect Request Message is sent to the voice server.

3.1.5.6 Echo Voice Session Subprotocol

3.1.5.6.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.1.5.6.2 Sending a Speech Message

The structure and fields of the Speech Message are specified in section 2.2.4.4.

When the **Speech Data Transmission Timer** fires, if a voice burst is active, the next speech message is sent. The speech message is sent directly to the voice server. Speech messages should not be sent before the Voice Client List Message is received. Speech messages should not be sent after the Client Disconnect Request Message is sent to the voice server.

3.1.5.6.3 Receiving a Speech with Bounce Message

The structure and fields of the Speech with Bounce Message are specified in section 2.2.2.7. This message MUST come from the current voice server, which is identified by the Current Voice Server DVID. The voice client MUST pass the message to the jitter buffer.

3.1.6 Timer Events

When the **Connection Request Timer** fires, the voice client will re-send the Connection Request Message (section 2.2.3.1). If the timer expires, the voice client will fail its connection attempt.

When the **Speech Data Transmission Timer** fires, the voice client checks to see if a voice burst is active and if it is, it sends the next speech message. The exact type of message sent to transmit the speech message depends on the voice session subprotocol.

3.1.7 Other Local Events

When the DirectPlay Protocol indicates that a DirectPlay client has left the voice session for any reason, the voice clients check to see if the DirectPlay client that left was the voice session host

(voice server). If the DirectPlay client that left was the voice server and the host migration extension is active, host migration is performed as specified in section 1.3.3.1. If the DirectPlay client that left was the voice server and the host migration extension is not active, the DirectPlay Voice Protocol will terminate all sending and receiving.

When the DirectPlay protocol indicates that connectivity with the DirectPlay voice session has been lost, the DirectPlay Voice Protocol will terminate all sending and receiving.

3.2 Voice Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Note In a **peer-to-peer** voice session, the voice client maintains the abstract data items identified in section 3.1.1. In a **client/server** voice session, the voice server maintains both the abstract data items identified in section 3.1.1, as well as those listed in this topic. Although the abstract data items identified in section 3.1.1 use the same names as those listed in this topic, there is no internal conflict. In a client/server configuration, the voice server maintains the abstract data items identified in this topic from the perspective of a voice server, while the items listed in section 3.1.1 are maintained from the perspective of the voice server as a **player** in the voice session, and therefore, as a voice client. Differentiation between the similarly named abstract data items is maintained internally by DirectPlay.

The following are abstract data items maintained by the voice server:

Voice Client List: A list of voice clients in the voice session. The voice client list contains the DVID and host order ID of each voice client in the session.

Next Host Order ID: When the Host Migration Extension (section 1.3.3.1) is active the voice server stores what the value of the next host order ID.

Client Voice Target List: The voice server maintains a voice target list for each voice client in the voice session when server controlled targeting is enabled. This list is used to target speech messages sent from the voice client. If the voice target list is empty then speech messages are discarded instead of sent.

Jitter Buffers: When the voice server is using the Mixing Voice Session Subprotocol (section 1.3.4) it maintains a jitter buffer for each voice client in the voice session.

Current Message Number: When the voice server is using the Mixing Voice Session Subprotocol (section 1.3.4) this contains the message number of the current voice burst. This value is stored per voice client. When a new voice burst begins to a voice client this value should be incremented for the specific voice client.

Current Sequence Number: When the voice server is using the Mixing Voice Session Subprotocol (section 1.3.4) this contains the sequence number of the next speech message being sent within the current voice burst. This value is stored per voice client. This value should be reset to zero when a new voice burst begins.

3.2.2 Timers

When the voice server is using the Mixing Voice Session Subprotocol (section 1.3.4), a **Speech Data Transmission Timer** is created. It fires on the period indicated by the currently selected codec as specified by the Frame Size (ms of data) column in the list of codecs in section 1.3.7.

3.2.3 Initialization

The underlying DirectPlay Session must be established before the voice client can begin operation. In addition the following MUST occur:

- **Voice Client List** is set to empty.
- **Next Host Order ID** is set to zero if the Host Migration Extension (section 1.3.3.1) is enabled.
- **Current Voice Target List** is set to empty for each voice client if server controlled targeting is enabled.
- **Jitter Buffers** are initialized if the voice session is using the Mixing Voice Session Subprotocol (section 1.3.4).
- **Current Message Number** is set to zero for all voice clients if the voice session is using the Mixing Voice Session Subprotocol (section 1.3.4).
- **Current Sequence Number** is set to zero for all voice clients if the voice session is using the Mixing Voice Session Subprotocol (section 1.3.4).

3.2.3.1 Initialization with Host Migration Extension

When the Host Migration Extension (section 1.3.3.1) is active, a voice server will be created by a voice client when host migration occurs. The voice server should start with the **Voice Client List** with the list of clients the voice client currently has. The voice server should also initialize the **Next Host Order ID** to the highest host order ID in the **Voice Client List** plus 255.

3.2.4 Higher-Layer Triggered Events

The following are events which can be triggered by a higher level:

Shutdown Request: The higher level can initiate a shutdown for the voice session. This will cause a Session Lost Message (section 2.2.2.1) to be sent to all voice clients if the Host Migration Extension (section 1.3.3.1) is disabled. If the Host Migration extension is enabled then this will cause a Voice Server Exited with Host Migration (section 2.2.4.5.1). It is also possible to disable the Host Migration extension before shutdown to cause a session to stop without migrating the host.

Change Voice Client List: The higher level can modify the **Current Voice Target List** for a specific voice client if Server Controlled Targeting is enabled. This will trigger a Set Client Voice Target Message (section 2.2.2.5) to be sent to the voice client.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Connection Subprotocol

3.2.5.1.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.2.5.1.2 Receiving Connect Request Message

The structure and fields of the Connect Request Message are specified in section 2.2.3.1.

Voice clients send this message to request that the connection process with the voice server start. This MUST be the first message sent to a voice server from a new voice client. Voice clients may send more than one of these messages during the connection process.

The voice server MUST respond to this message with a Connect Accept Message (section 2.2.3.2) if the voice server is ready to communicate with voice clients. If the server responds with a Connect Accept Message, it does not need to respond to any further Connect Request Messages received from a voice client. If the voice server is not ready to communicate with voice clients, it should respond with a Connect Refuse Message (section 2.2.3.3).

3.2.5.1.3 Sending Connect Accept Message

The structure and fields of the Connect Accept Message are specified in section 2.2.3.2.

The voice server responds directly to a voice client with this message after receiving a Connect Request Message (section 2.2.3.1) if the voice server is ready to accept connections.

3.2.5.1.4 Sending Connect Refuse Message

The structure and fields of the Connect Refuse Message are specified in section 2.2.3.3.

The voice server responds directly to a voice client with this message after receiving a Connect Request Message (section 2.2.3.1) if the voice server is not ready to accept connections.

3.2.5.1.5 Receiving a Client Capability Confirmation Message

The structure and fields of the Client Capability Confirmation Message are specified in section 2.2.2.8.

The voice client will send the voice server a Client Capability Confirmation Message when it has confirmed that it can support the type of voice session subprotocol and codec that the voice server is using. Once this message is received from a voice client, the voice server can start talking the specific voice session subprotocol with the voice client. The voice server should also take the following steps:

- Add the specified voice client to the **Voice Client List**.
- If the Host Migration Extension (section 1.3.3.1) is enabled:
  - Assign the value of the **Next Host Order ID** to the new voice client.
  - Increment the **Next Host Order ID**.
  - If the voice client presents a host order ID that is not equal to 0xFFFFFFFF and exceeds the current value of **Next Host Order ID**, the **Next Host Order ID** is incremented by 255.
- If the server is running in Mixing Voice Session Subprotocol (section 1.3.4) it should initialize the jitter buffer for the individual voice client.

3.2.5.2 Common Message for All Voice Session Subprotocols

3.2.5.2.1 Sending a Session Lost Message

The structure and fields of the Session Lost Message are specified in section 2.2.2.1.

The voice server sends this message to all voice clients in the session when the voice server is shutting down and the Host Migration Extension (section 1.3.3.1) is not enabled. Once this message is sent, the voice server should no longer respond to messages from voice clients.

3.2.5.2.2 Sending a Set Client Voice Target Message

The structure and fields of the Set Client Voice Target Message are specified in section 2.2.2.5.

The voice server sends a Set Client Voice Target Message when the higher level triggers a **Change Voice Client List** event.

3.2.5.2.3 Receiving a Client Disconnect Request Message

The structure and fields of the Client Disconnect Request Message are specified in section 2.2.2.2.

The voice client sends this message to the voice server when it wants to disconnect gracefully from the session. The voice server should remove the voice client from the **Voice Client List.** If the voice server is running the Peer Voice Session Subprotocol (section 1.3.3), the voice server will also send a Remove Voice Client Message (section 2.2.4.3) to all the voice clients. The voice server will then respond to the sender of this message with a Client Disconnect Confirm (section 2.2.2.3) message.

3.2.5.2.4 Sending a Client Disconnect Confirm Message

The structure and fields of the Client Disconnect Confirmation Message are specified in section 2.2.2.3.

The voice server sends this message to the voice client in response to a Client Disconnect Request Message (section 2.2.2.2).

3.2.5.3 Peer Voice Session Subprotocol

3.2.5.3.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.2.5.3.2 Sending a Voice Client List Message

The structure and fields of the Voice Client List Message are specified in section 2.2.4.2.

The voice server sends the voice client a list of the active voice clients in the session including the voice client this message is sent to. The voice server sends the list using one or move Voice Client List Messages. Each individual message can hold the information for up to 0x52 voice clients. If there are more than 0x52 voice clients then one Voice Client List Message is also sent for each 0x52 clients and one for any remaining voice clients. After the voice server has sent the last Voice Client List Message list to the voice client needed to provide the full voice client list an Add Voice Client Message (section 2.2.2.4) is sent to all voice clients in the session.

3.2.5.3.3 Sending an Add Voice Client Message

The structure and fields of the Add Voice Client Message are specified in section 2.2.2.4.

The voice server sends the Add Voice Client Message to all voice clients in the session to indicate that a new voice client has joined the voice session. This message is sent after the Voice Client List Messages are sent to the individual voice client.

3.2.5.3.4 Sending a Remove Voice Client Message

The structure and fields of the Remove Voice Client Message are specified in section 2.2.4.3.

The voice server sends the Remove Voice Client Message to all voice clients in the session when a voice client has left the session. If the voice client leaves the session gracefully, by sending a Client Disconnect Request Message (section 2.2.2.2) then this message is sent after the Client Disconnect Confirm Message (section 2.2.2.3) is sent to the voice client. If the voice client is removed from the session through a notification from the DirectPlay layer then this message is sent after the notification is received.

3.2.5.3.5 Host Migration Extension

3.2.5.3.5.1 Sending a Voice Server Exited With Host Migration Message

The structure and fields of the Voice Server Exited with Host Migration Message are specified in section 2.2.4.5.1.

The voice server sends this message to all voice clients when the voice server shuts down. The voice server can ignore all messages after this message is sent.

3.2.5.3.5.2 Sending a Host Migration Complete Message

The structure and fields of the Host Migration Complete Message are specified in section 2.2.4.5.2.

When a new voice server is created by a voice client and is ready to receive traffic it sends this message to all voice clients in the session.

3.2.5.3.5.3 Receiving a Client Capability Confirmation Message

The structure and fields of the Client Capability Confirmation Message are specified in section 2.2.2.8.

When voice clients determine a host migration is occurring they send a Client Capability Confirmation Message to the new voice server. If the new voice server does not know about the voice client the voice server treats the message as if it was receiving a Client Capability Confirmation Message from the Connection Subprotocol (section 1.3.2).

3.2.5.4 Mixing Voice Session Subprotocol

3.2.5.4.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.2.5.4.2 Sending an Add Voice Client Message

The structure and fields of the Add Voice Client Message are specified in section 2.2.2.4.

The voice server sends an Add Voice Client Message after they have received the Client Capability Confirmation Message (section 2.2.2.8) during the Connection Subprotocol (section 1.3.2).

3.2.5.4.3 Sending a Speech with Bounce Message

The structure and fields of the Speech with Bounce Message are specified in section 2.2.2.7.

When the **Speech Data Transmission Timer** fires, the voice server does the following for each voice client:

- Checks the jitter buffer for each of the other voice clients to see if any of them have a speech message to send to the voice client. If no other voice clients have a speech message for the voice client, the voice server moves onto the next voice client.
- Decompresses each of the voice messages that are intended for the voice client using the codec in use by this session.
- Mixes the audio to create a single voice message that combines all the voice messages.
- Compresses the combined voice message by using the codec in use by this session.
- Transmits the combined voice message by using a Speech with Bounce Message to the voice client.

3.2.5.4.4 Receiving a Speech with Target Message

The structure and fields of the Speech with Target Message are specified in section 2.2.2.6.

The voice server receives a Speech with Target Message from a voice client it places the enclosed speech message and the list of targets into the jitter buffer for the originating voice client.

3.2.5.5 Forwarding Voice Session Subprotocol

3.2.5.5.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.2.5.5.2 Receiving a Speech with Target Message

The structure and fields of the Speech with Target Message are specified in section 2.2.2.6.

The voice server receives a Speech with Target Message from a voice client it creates a new Speech with From Message (section 2.2.5.1) and copies the **SpeechData**, **MessageNumber** and **SequenceNumber** fields from the Speech with Target Message. The voice server then copies the DVID of the voice client that sent the message into the **SourceDVID** field of the new Speech with Target Message. The Speech with Target Message is then sent to the list of DVIDs from the **TargetDVIDs** field of the Speech with Target Message.

3.2.5.5.3 Sending a Speech With From Message

The structure and fields of the Speech with From Message are specified in section 2.2.5.1. See Receiving a Speech with Target Message (section 3.2.5.5.2) for information on this message.

3.2.5.6 Echo Voice Session Subprotocol

3.2.5.6.1 Handling Unrecognized Messages

If an unrecognized or malformed message is received it is ignored.

3.2.5.6.2 Receiving a Speech Message

The structure and fields of the Speech Message are specified in section 2.2.4.4.

When the voice server receives a **Speech Message** it immediately sends a Speech with Bounce Message (Section 2.2.2.7) back to the originating voice client. The **MessageNumber**, **SequenceNumber** and **SpeechData** fields are copied from the contents of the Speech Message.

3.2.5.6.3 Sending a Speech Bounce Message

The structure and fields of the Speech with Bounce Message are specified in section 2.2.2.7. See Receiving a Speech Message (section 3.2.5.6.2) for information on this message.

3.2.6 Timer Events

When the voice server is using the Mixing Voice Session Subprotocol (section 1.3.4) a **Speech Data Transmission Timer** is created. It fires on the period indicated by the currently selected codec as specified by the Frame Size (ms of data) column in the list of codecs in section 1.3.7.

When the **Speech Data Transmission Timer** fires, the next Speech with Bounce Message (section 2.2.2.7) is sent to each voice client in the session (if there is one). For additional information, see Sending a Speech with Bounce Message (section 3.2.5.4.3).

3.2.7 Other Local Events

When the DirectPlay protocol indicates that a DirectPlay client has left the session for any reason the voice server will check to see if the DirectPlay client was one of the voice clients. If the DirectPlay client was one of the voice clients then the voice server removes it from the voice client list. If the Peer Voice Session Subprotocol (section 1.3.3) is being used then a Remove Voice Client Message is sent to all voice clients.

When the DirectPlay protocol indicates that connectivity with the DirectPlay session has been lost the DirectPlay Voice Protocol will terminate all sending and receiving.

4 Protocol Examples

The following section describes a common scenario in which the DirectPlay Voice Protocol Specification is used

4.1 Successful Connect Sequence

The following examples demonstrate successful connect sequences.

Connect Request Message (section 2.2.3.1):

DirectPlay Protocol, Client to Server, Guaranteed and Ordered

```
00000000 51 01 00 03 00 00 00
51 -> MessageType = DVMSGID_CONNECTREQUEST = 0x51
01 -> VersionMajor = 0x01
00 -> VersionMinor = 0x00
03 00 00 00 -> VersionBuild = 0x00000003
```

Connect Accept Message (section 2.2.3.2):

DirectPlay Protocol, Server to Client, Guaranteed and Ordered

```
00000000 56 01 00 00 00 01 00 03
00000008 00 00 00 00 00 00 00 9B
00000010 A2 82 7D 42 22 82 4F 8F
00000018 39 5D 11 53 DF 3E 41
56 -> MessageType = DVMSGID_CONNECTACCEPT = 0x56
01 00 00 00 -> SessionType = DVSESSIONTYPE_PEER = 0x00000001
01 -> VersionMajor = 0x01
00 -> VersionMinor = 0x00
03 00 00 00 -> VersionBuild = 0x00000003
00 00 00 00 -> Session Flags = 0x00000000
9B A2 82 7D 42 22 82 4F 8F 39 5D 11 53 DF 3E 41 -> CompressionType =
DPVCTGUID_SC03 {7D82A29B-2242-4F82-8F39-5D1153DF3E41}
```

Client Capability Confirmation Message (section 2.2.2.8):

DirectPlay Protocol, Client to Server, Guaranteed and Ordered

```
00000000 58 00 00 00 00 FF FF FF FF
58 -> MessageType = DVMSGID_SETTINGSCONFIRM = 0x58
00 00 00 00 -> PlayerFlags = 0x00000000
FF FF FF FF -> HostOrderID = 0xFFFFFFFF
```

At this point the voice server starts communicating with the Peer Voice Session Subprotocol (section 1.3.3).

Voice Client List Message (section 2.2.4.2):

DirectPlay Protocol, Server to Client, Guaranteed and Ordered

```
00000000 61 01 00 00 00 02 00 00
00000008 00 AE F4 42 59 00 00 00
00000010 00 01 00 00 00 AE F4 52
00000018 59 00 00 00 00 00 00 00
00000020 00
61 -> MessageType = DVMSGID_PLAYERLIST = 0x61
01 00 00 00 -> HostOrderID = 0x00000001
02 00 00 00 -> NumPlayerListEntries = 0x00000002
AE F4 42 59 -> Voice Client[0] DVID = 0x5942F4AE
00 00 00 00 -> Voice Client[0] PlayerFlags = 0x00000000
01 00 00 00 -> Voice Client[0] Host Order ID = 0x00000001
AE F4 52 59 -> Voice Client[1] DVID = 0x5952F4AE
00 00 00 00 -> Voice Client[1] PlayerFlags = 0x00000000
00 00 00 00 -> Voice Client[1] Host Order ID = 0x00000000
```

Add Voice Client Message (section 2.2.2.4):

DirectPlay Protocol, Server to Server, Guaranteed and Ordered

```
00000000 01 AE F4 42 59 00 00 00
00000008 00 01 00 00 00
01 -> MessageType = DVMSGID_CREATEVOICEPLAYER = 0x01
AE F4 42 59 -> DVID = 0x5942F4AE
00 00 00 00 -> PlayerFlags = 0x00000000
01 00 00 00 -> HostOrderID = 0x00000001
```

5 Security

The following sections specify security considerations for implementers of the DirectPlay Voice Protocol.

5.1 Security Considerations for Implementers

The DirectPlay Voice Protocol Specification does not provide any specific security features. The following are some considerations implementers should be aware of:

- Check all packets to ensure they are of the proper length and contain valid values.
- Do not allocate any resources for a voice client until a Client Capability Confirmation Message (section 2.2.2.8) has been received.

5.2 Index of Security Parameters

None.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows XP
- Windows Server 2003

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Windows does not follow the prescription.

<1> Section 3.1.5.1.3: In DirectPlay 4, the Connect Accept Message arrives as type **DPSP_MSG_VOICE** ([MC-DPL4CS] section 2.2.54). This message type includes the **dwIDFrom** field which identifies the **Player ID** of the source for the voice data. The value of the **dwIDFrom** field is passed up to the voice layer of the DirectPlay Protocol and becomes the DVID of the sender of the Connect Accept Message.

In DirectPlay 8, whenever any voice message is passed up to the voice layer of the DirectPlay Protocol, the **DPNID** of the sender of the message is always included. The value of the **DPNID** becomes the DVID of the sender of the Connect Accept Message.

7 Index

T

V

W