**[MS-RDPEAI]:**

**Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension**

Intellectual Property Rights Notice for Open Specifications Documentation

* **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
* **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights**. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 12/5/2008 | 0.1 | Major | Initial Availability |
| 1/16/2009 | 0.1.1 | Editorial | Changed language and formatting in the technical content. |
| 2/27/2009 | 0.1.2 | Editorial | Changed language and formatting in the technical content. |
| 4/10/2009 | 0.1.3 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 0.2 | Minor | Clarified the meaning of the technical content. |
| 7/2/2009 | 0.2.1 | Editorial | Changed language and formatting in the technical content. |
| 8/14/2009 | 0.2.2 | Editorial | Changed language and formatting in the technical content. |
| 9/25/2009 | 1.0 | Major | Updated and revised the technical content. |
| 11/6/2009 | 2.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 1/29/2010 | 3.0 | Major | Updated and revised the technical content. |
| 3/12/2010 | 4.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 4.0.1 | Editorial | Changed language and formatting in the technical content. |
| 6/4/2010 | 4.1 | Minor | Clarified the meaning of the technical content. |
| 7/16/2010 | 4.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/27/2010 | 4.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 4.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010 | 4.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011 | 4.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/11/2011 | 5.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 5.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/6/2011 | 5.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 5.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 5.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 6.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 7.0 | Major | Updated and revised the technical content. |
| 7/12/2012 | 8.0 | Major | Updated and revised the technical content. |
| 10/25/2012 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/31/2013 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 9.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 9.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 9.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 10.0 | Major | Updated and revised the technical content. |
| 6/30/2015 | 11.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 7](#_Toc483457068)

[1.1 Glossary 7](#_Toc483457069)

[1.2 References 7](#_Toc483457070)

[1.2.1 Normative References 8](#_Toc483457071)

[1.2.2 Informative References 8](#_Toc483457072)

[1.3 Overview 8](#_Toc483457073)

[1.3.1 Initialization Sequence 8](#_Toc483457074)

[1.3.2 Data Transfer Sequence 9](#_Toc483457075)

[1.3.3 Format Change Sequence 10](#_Toc483457076)

[1.4 Relationship to Other Protocols 10](#_Toc483457077)

[1.5 Prerequisites/Preconditions 10](#_Toc483457078)

[1.6 Applicability Statement 11](#_Toc483457079)

[1.7 Versioning and Capability Negotiation 11](#_Toc483457080)

[1.8 Vendor-Extensible Fields 11](#_Toc483457081)

[1.9 Standards Assignments 11](#_Toc483457082)

[2 Messages 12](#_Toc483457083)

[2.1 Transport 12](#_Toc483457084)

[2.2 Message Syntax 12](#_Toc483457085)

[2.2.1 SNDIN\_PDU Header 12](#_Toc483457086)

[2.2.2 Initialization Messages 12](#_Toc483457087)

[2.2.2.1 Version PDU (MSG\_SNDIN\_VERSION) 13](#_Toc483457088)

[2.2.2.2 Sound Formats PDU (MSG\_SNDIN\_FORMATS) 13](#_Toc483457089)

[2.2.2.3 Open PDU (MSG\_SNDIN\_OPEN) 14](#_Toc483457090)

[2.2.2.3.1 Extended Wave Format Structure (WAVEFORMAT\_EXTENSIBLE) 15](#_Toc483457091)

[2.2.2.4 Open Reply PDU (MSG\_SNDIN\_OPEN\_REPLY) 16](#_Toc483457092)

[2.2.3 Data Transfer Messages 17](#_Toc483457093)

[2.2.3.1 Incoming Data PDU (MSG\_SNDIN\_DATA\_INCOMING) 17](#_Toc483457094)

[2.2.3.2 Data PDU (MSG\_SNDIN\_DATA) 17](#_Toc483457095)

[2.2.4 Format Change Messages 18](#_Toc483457096)

[2.2.4.1 Format Change PDU (MSG\_SNDIN\_FORMATCHANGE) 18](#_Toc483457097)

[3 Protocol Details 19](#_Toc483457098)

[3.1 Common Details 19](#_Toc483457099)

[3.1.1 Abstract Data Model 19](#_Toc483457100)

[3.1.2 Timers 19](#_Toc483457101)

[3.1.3 Initialization 19](#_Toc483457102)

[3.1.4 Higher-Layer Triggered Events 19](#_Toc483457103)

[3.1.4.1 Recording Audio 19](#_Toc483457104)

[3.1.5 Message Processing Events and Sequencing Rules 20](#_Toc483457105)

[3.1.5.1 Protocol Initialization 20](#_Toc483457106)

[3.1.5.2 Protocol Termination 21](#_Toc483457107)

[3.1.6 Timer Events 21](#_Toc483457108)

[3.1.7 Other Local Events 21](#_Toc483457109)

[3.2 Client Details 21](#_Toc483457110)

[3.2.1 Abstract Data Model 21](#_Toc483457111)

[3.2.2 Timers 21](#_Toc483457112)

[3.2.3 Initialization 21](#_Toc483457113)

[3.2.4 Higher-Layer Triggered Events 21](#_Toc483457114)

[3.2.5 Message Processing Events and Sequencing Rules 21](#_Toc483457115)

[3.2.5.1 Initialization Sequence 21](#_Toc483457116)

[3.2.5.1.1 Processing a Version PDU 21](#_Toc483457117)

[3.2.5.1.2 Sending a Version PDU 21](#_Toc483457118)

[3.2.5.1.3 Processing a Sound Formats PDU 21](#_Toc483457119)

[3.2.5.1.4 Sending an Incoming Data PDU 22](#_Toc483457120)

[3.2.5.1.5 Sending a Sound Formats PDU 22](#_Toc483457121)

[3.2.5.1.6 Processing an Open PDU 22](#_Toc483457122)

[3.2.5.1.7 Sending a Format Change PDU 22](#_Toc483457123)

[3.2.5.1.8 Sending an Open Reply PDU 22](#_Toc483457124)

[3.2.5.2 Data Transfer Sequence 23](#_Toc483457125)

[3.2.5.2.1 Sending an Incoming Data PDU 23](#_Toc483457126)

[3.2.5.2.2 Sending a Data PDU 23](#_Toc483457127)

[3.2.5.3 Format Change Sequence 23](#_Toc483457128)

[3.2.5.3.1 Processing a Format Change PDU 23](#_Toc483457129)

[3.2.5.3.2 Sending a Format Change PDU 23](#_Toc483457130)

[3.2.6 Timer Events 23](#_Toc483457131)

[3.2.7 Other Local Events 23](#_Toc483457132)

[3.3 Server Details 23](#_Toc483457133)

[3.3.1 Abstract Data Model 23](#_Toc483457134)

[3.3.2 Timers 24](#_Toc483457135)

[3.3.3 Initialization 24](#_Toc483457136)

[3.3.4 Higher-Layer Triggered Events 24](#_Toc483457137)

[3.3.5 Message Processing Events and Sequencing Rules 24](#_Toc483457138)

[3.3.5.1 Initialization Sequence 24](#_Toc483457139)

[3.3.5.1.1 Sending a Version PDU 24](#_Toc483457140)

[3.3.5.1.2 Processing a Version PDU 24](#_Toc483457141)

[3.3.5.1.3 Sending a Sound Formats PDU 24](#_Toc483457142)

[3.3.5.1.4 Processing an Incoming Data PDU 24](#_Toc483457143)

[3.3.5.1.5 Processing a Sound Formats PDU 24](#_Toc483457144)

[3.3.5.1.6 Sending an Open PDU 25](#_Toc483457145)

[3.3.5.1.7 Processing a Format Change PDU 25](#_Toc483457146)

[3.3.5.1.8 Processing an Open Reply PDU 25](#_Toc483457147)

[3.3.5.2 Data Transfer Sequence 25](#_Toc483457148)

[3.3.5.2.1 Processing an Incoming Data PDU 25](#_Toc483457149)

[3.3.5.2.2 Processing a Data PDU 25](#_Toc483457150)

[3.3.5.3 Format Change Sequence 25](#_Toc483457151)

[3.3.5.3.1 Sending a Format Change PDU 25](#_Toc483457152)

[3.3.5.3.2 Processing a Format Change PDU 25](#_Toc483457153)

[3.3.6 Termination 26](#_Toc483457154)

[3.3.7 Timer Events 26](#_Toc483457155)

[3.3.8 Other Local Events 26](#_Toc483457156)

[4 Protocol Examples 27](#_Toc483457157)

[4.1 Annotated Initialization Sequence 27](#_Toc483457158)

[4.1.1 Server Version PDU 27](#_Toc483457159)

[4.1.2 Client Version PDU 27](#_Toc483457160)

[4.1.3 Server Sound Formats PDU 27](#_Toc483457161)

[4.1.4 Incoming Data PDU 31](#_Toc483457162)

[4.1.5 Client Sound Formats PDU 31](#_Toc483457163)

[4.1.6 Open PDU 36](#_Toc483457164)

[4.1.7 Format Change PDU 36](#_Toc483457165)

[4.1.8 Open Reply PDU 36](#_Toc483457166)

[4.2 Annotated Data Transfer Sequence 36](#_Toc483457167)

[4.2.1 Incoming Data PDU 36](#_Toc483457168)

[4.2.2 Data PDU 37](#_Toc483457169)

[4.3 Annotated Format Change Sequence 37](#_Toc483457170)

[4.3.1 Server Format Change PDU 37](#_Toc483457171)

[4.3.2 Client Format Change PDU 37](#_Toc483457172)

[5 Security 38](#_Toc483457173)

[5.1 Security Considerations for Implementers 38](#_Toc483457174)

[5.2 Index of Security Parameters 38](#_Toc483457175)

[6 Appendix A: Product Behavior 39](#_Toc483457176)

[7 Change Tracking 41](#_Toc483457177)

[8 Index 42](#_Toc483457178)

# Introduction

The Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension seamlessly transfers audio data from a Remote Desktop Protocol client to a Remote Desktop Protocol server.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

## Glossary

This document uses the following terms:

**audio format**: A data structure that is used to define waveform-audio data. The actual structure of individual formats is opaque to the underlying transport protocol. For more information, see [MSDN-AUDIOFORMAT].

**dynamic virtual channel**: A transport used for lossless communication between an RDP client and a server component over a main data connection, as specified in [MS-RDPEDYC].

**Dynamic Virtual Channel (DVC) Listener (or Listener)**: A named endpoint registered at the TS client during initialization of a DVC. DVC listeners are service providers to the applications that run on a TS server.

**globally unique identifier (GUID)**: A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [[RFC4122]](https://go.microsoft.com/fwlink/?LinkId=90460) or [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) must be used for generating the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). See also universally unique identifier (UUID).

**HRESULT**: An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1 and specific protocol documents for further details.

**little-endian**: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**protocol data unit (PDU)**: Information that is delivered as a unit among peer entities of a network and that may contain control information, address information, or data. For more information on remote procedure call (RPC)-specific PDUs, see [C706] section 12.

**Remote Desktop Protocol (RDP) client**: The client that initiated a remote desktop connection.

**Remote Desktop Protocol (RDP) server**: The server to which a client initiated a remote desktop connection.

**Wave Capture Device**: A device that captures audio to the computer.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](http://msdn.microsoft.com/en-us/library/dn781092.aspx).

### 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](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)".

[MS-RDPBCGR] Microsoft Corporation, "[Remote Desktop Protocol: Basic Connectivity and Graphics Remoting](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c)".

[MS-RDPEA] Microsoft Corporation, "[Remote Desktop Protocol: Audio Output Virtual Channel Extension](%5bMS-RDPEA%5d.pdf#Section_bea2d5cfe3b9441992e50e074ff9bc5b)".

[MS-RDPEDYC] Microsoft Corporation, "[Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, [http://www.rfc-editor.org/rfc/rfc2119.txt](https://go.microsoft.com/fwlink/?LinkId=90317)

[RFC2361] Fleischman, E., "WAVE and AVI Codec Registries", RFC 2361, June 1998, [http://www.ietf.org/rfc/rfc2361.txt](https://go.microsoft.com/fwlink/?LinkId=90337)

### Informative References

[ETSI-GSM] European Telecommunications Standards Organization, "GSM UMTS 3GPP Numbering Cross Reference", March 2008, [http://webapp.etsi.org/key/key.asp?GSMSpecPart1=06&GSMSpecPart2=10&Search=search](https://go.microsoft.com/fwlink/?LinkId=114230)

[MSDN-WAVEFMTEXT] Microsoft Corporation, "WAVEFORMATEXTENSIBLE", [http://msdn.microsoft.com/en-us/library/ms713496.aspx](https://go.microsoft.com/fwlink/?LinkId=128854)

## Overview

This section provides a high-level overview of the operation of the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension. The purpose of this protocol is to transfer audio data from a [**Remote Desktop Protocol (RDP) client**](#gt_b58fec62-3d34-4209-94eb-e933b4cd4513) to a [**Remote Desktop Protocol (RDP) server**](#gt_2041833b-a196-4b8a-b8a8-8ed5424415b9), hereinafter referred to as client and server, respectively. For example, an application running on a server can request to record audio data. This data will be transferred from the client to the server, allowing a server application to record from an audio device installed on the client.

The protocol is divided into three main sequences:

**Initialization sequence:** The server and client exchange versions and audio formats, and begin recording.

**Data transfer sequence:** The client sends audio data to the server.

**Format change sequence:** The server requests a new [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044), and the client confirms this request.

### Initialization Sequence

The initialization sequence has the following goals:

1. To establish the client and server protocol versions and capabilities.
2. To establish a list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) supported by both the client and the server.
3. To begin recording audio data.

Initially, the server sends a [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) to the client within the already established [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3). The client will respond with its own Version PDU. Next, the server will send a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d), which contains a list of the audio formats the server supports. The client sends its own Sound Formats PDU to the server, establishing the common list of audio formats. All audio data will be encoded using one of the formats in this list.

Once the audio formats have been exchanged, the server will indicate that it has requested to begin recording, by sending an [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0). The client will attempt to start recording from an attached audio capture device and return the result to the server in an [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91). At this point, the client will begin sending audio data.

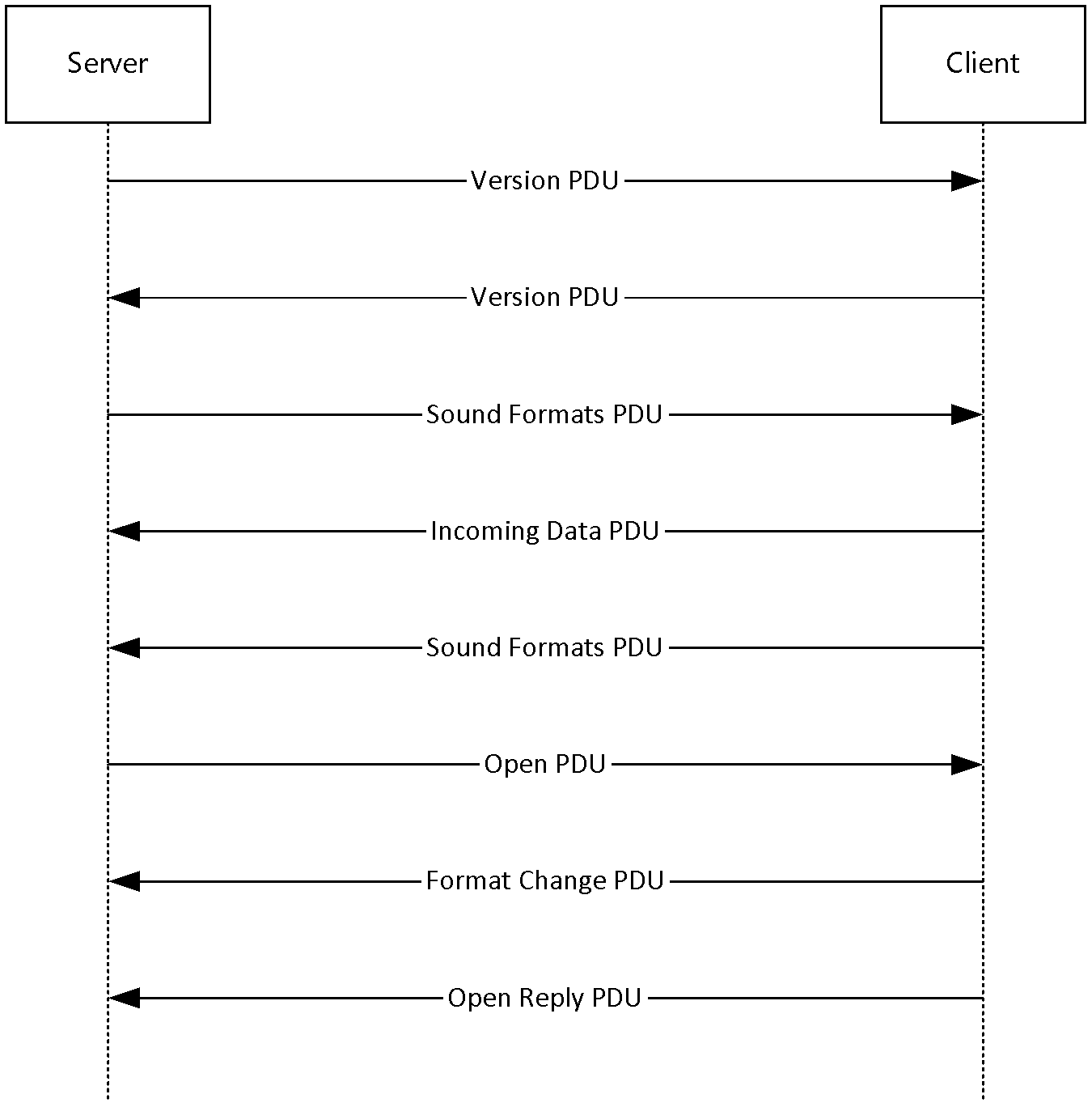


Figure 1: Initialization sequence

### Data Transfer Sequence

The data transfer sequence simply transfers audio data from the client to the server. The client will encode captured audio data using the current [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) agreed on during either the initialization sequence or the format change sequence and send it to the server. The client first sends an [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92), which informs the server that the next packet will contain audio data. The client will then send the audio data in a [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f).

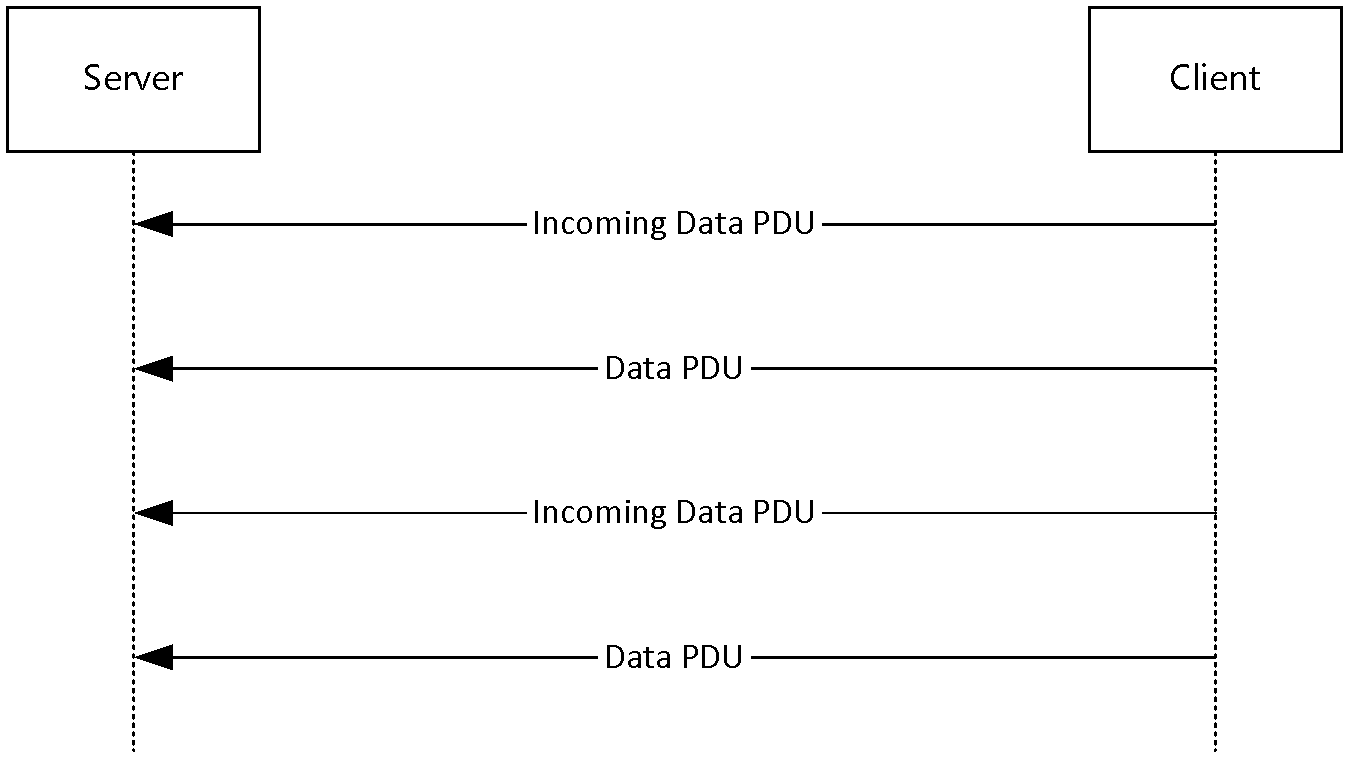


Figure 2: Data transfer sequence

### Format Change Sequence

The format change sequence provides a mechanism for the server to request that the client use a different format for encoding the audio data. The server initiates the sequence by sending a [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866), identifying the server's desired format out of the list that was agreed on during the initialization sequence. The client will then confirm this change of format by sending a Format Change PDU specifying the same format. From this point, the client will encode audio data using the new format.

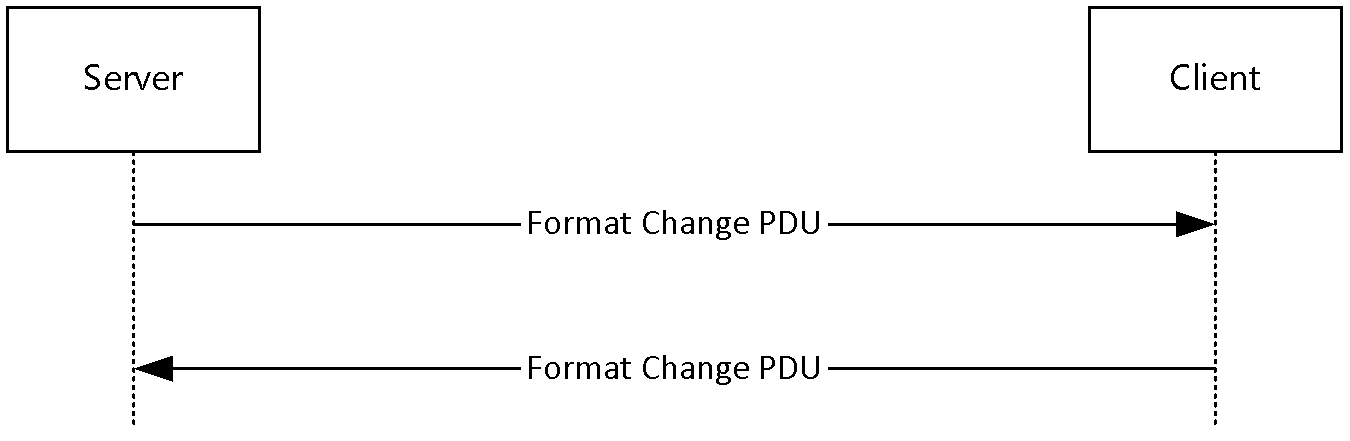


Figure 3: Format change sequence

## Relationship to Other Protocols

The Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension is embedded in a [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3) transport, as specified in [[MS-RDPEDYC]](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06).

## Prerequisites/Preconditions

The Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension operates only after the [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3) transport, as specified in [[MS-RDPEDYC]](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06), is fully established. If the dynamic virtual channel transport is terminated, no other communication occurs over the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension.

All multiple-byte fields within a message are assumed to contain data in [**little-endian**](#gt_079478cb-f4c5-4ce5-b72b-2144da5d2ce7) byte ordering, unless otherwise specified.

## Applicability Statement

The Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension is designed to run within the context of a Remote Desktop Protocol (RDP) [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3) established between an [**RDP client**](#gt_b58fec62-3d34-4209-94eb-e933b4cd4513) and [**RDP server**](#gt_2041833b-a196-4b8a-b8a8-8ed5424415b9). The protocol is applicable when the client is required to record audio and transfer the recorded audio to the server.

## Versioning and Capability Negotiation

The Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension is capability-based. The client and the server exchange capabilities during the protocol initialization sequence (as specified in section [3.2.5.1](#Section_84d6c7b318864157920fc9a5bf68ff64), and section [3.3.5.1](#Section_32478681c8b547e8bff5ed2508bed7ae)). After the capabilities have been exchanged, the client and server do not send [**protocol data units (PDUs)**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) or data formats that cannot be processed by the other.

## Vendor-Extensible Fields

This protocol uses [**HRESULT**](#gt_799103ab-b3cb-4eab-8c55-322821b2b235) values as defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1.1. Vendors can define their own HRESULT values, provided that they set the C bit (0x20000000) for each vendor-defined value, indicating that the value is a customer code.

## Standards Assignments

None.

# Messages

## Transport

This protocol is designed to operate over a [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3), as specified in [[MS-RDPEDYC]](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06) section 1.1. The dynamic virtual channel name is the null-terminated ANSI encoded character string "AUDIO\_INPUT", which is the name of the [**Listener**](#gt_bb000f88-ce99-44ce-bdc2-2e438a2f131f) on the client side. The usage of a channel name when opening a dynamic virtual channel is specified in [MS-RDPEDYC] section 2.2.2.1. The RDP layer manages the creation, setup, and transmission of data over the dynamic virtual channel.

## Message Syntax

The following sections define the syntax for the various [**PDUs**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) in each protocol sequence. The listing is not exclusive; some PDUs can appear in other sequences.

### SNDIN\_PDU Header

The SNDIN\_PDU header MUST be included in all audio capture PDUs. It identifies the type of the [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| MessageId | | | | | | | |

**MessageId (1 byte):** An 8-bit unsigned integer that specifies the type of audio PDU.

| Value | Meaning |
| --- | --- |
| MSG\_SNDIN\_VERSION  0x01 | [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) |
| MSG\_SNDIN\_FORMATS  0x02 | [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) |
| MSG\_SNDIN\_OPEN  0x03 | [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) |
| MSG\_SNDIN\_OPEN\_REPLY  0x04 | [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) |
| MSG\_SNDIN\_DATA\_INCOMING  0x05 | [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) |
| MSG\_SNDIN\_DATA  0x06 | [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f) |
| MSG\_SNDIN\_FORMATCHANGE  0x07 | [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) |

### Initialization Messages

The following sections contain the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension message syntax for exchanging versions and capabilities, establishing a list of audio formats supported by both the client and the server, and starting audio data recording. For more information, see section [3.1.3](#Section_547a89a5425c4fcca548dccba6d89dd6).

#### Version PDU (MSG\_SNDIN\_VERSION)

The Version PDU is sent by the server and the client to negotiate which version of the protocol MUST be used.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | Version | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**Header (1 byte):** A [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_VERSION (0x01).

**Version (4 bytes):** A 32-bit unsigned integer. This field specifies the protocol version used by the client or server that sent this [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5). This field MUST be set to 0x00000001.

#### Sound Formats PDU (MSG\_SNDIN\_FORMATS)

The Sound Formats PDU is sent by the server and the client to negotiate a common list of supported [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | NumFormats | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | cbSizeFormatsPacket | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | SoundFormats (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | ExtraData (variable) (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**Header (1 byte):** An [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_FORMATS (0x02).

**NumFormats (4 bytes):** A 32-bit unsigned integer. This field specifies the number of formats in the **SoundFormats** array.

**cbSizeFormatsPacket (4 bytes):** A 32-bit unsigned integer. This field is reserved in the Sound Formats PDU that is sent from server to client. The value is arbitrary in the PDU sent from server to client, and MUST be ignored upon receipt on the client side. In the PDU sent from client to server, this field MUST be set to the size, in bytes, of the entire PDU minus the size, in bytes, of the **ExtraData** field.

**SoundFormats (16 bytes):** A variable-sized array of audio formats supported by the client or server. The number of formats is **NumFormats**. Each element in the array conforms to the AUDIO\_FORMAT, as specified in [[MS-RDPEA]](%5bMS-RDPEA%5d.pdf#Section_bea2d5cfe3b9441992e50e074ff9bc5b) section 2.2.2.1.1. Any of the audio codecs defined in [[RFC2361]](https://go.microsoft.com/fwlink/?LinkId=90337) are allowed by this protocol. Implementations MUST, at a minimum, support WAVE\_FORMAT\_PCM (0x0001). For more information about this codec, see [MS-RDPEA] section 2.2.2.1.1.[<1>](#Appendix_A_1" \o "Product behavior note 1)

**ExtraData (variable):** An optional field that contains additional data. This data MAY be appended to the end of a Sound Formats PDU. The data is arbitrary and MUST be ignored by the recipient. The recipient finds out where **ExtraData** starts by parsing the PDU. This will tell the total size of all the data fields in this PDU except **ExtraData**. The size of ExtraData can be calculated by subtracting the size of the useful data from the size of the PDU. The **MSG\_SNDIN\_FORMATS** structure is encapsulated in the Dynamic VC Data First PDU ([[MS-RDPEDYC]](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06) section 2.2.3.1) and Dynamic VC Data PDU ([MS-RDPEDYC] section 2.2.3.2). The length of these PDUs is determined as described in [MS-RDPEDYC] sections 3.1.5.1 and 3.1.5.2.

#### Open PDU (MSG\_SNDIN\_OPEN)

The Open PDU is sent by the server and indicates that the server requests to start recording from the client device. The **wFormatTag**, **nChannels**, **nSamplesPerSec**, **nAvgBytesPerSec**, **nBlockAlign**, **wBitsPerSample**, **cbSize**, and **ExtraFormatData** fields specify the [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) that the client SHOULD use to record from the audio input device. The **initialFormat** field contains an index into the list of audio formats agreed to by the client and the server. The **initialFormat** field specifies the format in which audio data MUST be encoded.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | FramesPerPacket | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | initialFormat | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | wFormatTag | | | | | | | | | | | | | | | | nChannels | | | | | | | |
| ... | | | | | | | | nSamplesPerSec | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | nAvgBytesPerSec | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | nBlockAlign | | | | | | | | | | | | | | | | wBitsPerSample | | | | | | | |
| ... | | | | | | | | cbSize | | | | | | | | | | | | | | | | ExtraFormatData (variable) (16 bytes) | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | |

**Header (1 byte):** A [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_OPEN (0x03).

**FramesPerPacket (4 bytes):** A 32-bit unsigned integer. This field specifies the number of audio frames that the client MUST send in each [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f). The product of nChannels, 2, and FramesPerPacket is the size of audio data that the client MUST put in a Data PDU. The audio data size is determined by the server application and is not determined by the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension.

**initialFormat (4 bytes):** A 32-bit unsigned integer. This field is an index into the list of audio formats agreed on by the client and server. This field specifies the initial audio format.

**wFormatTag (2 bytes):** A 16-bit unsigned integer identifying the compression codec of the audio format. For more information on compression formats, see [[RFC2361]](https://go.microsoft.com/fwlink/?LinkId=90337). If this field is set to WAVE\_FORMAT\_EXTENSIBLE (0xFFFE), the **cbSize** field MUST be equal to 22.

**nChannels (2 bytes):** A 16-bit unsigned integer. This field specifies the number of channels in the audio format identified by **wFormatTag**. It is part of the audio format and is not determined by the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension.

**nSamplesPerSec (4 bytes):** A 32-bit unsigned integer. This field specifies the sampling rate, in hertz, for recording audio.

**nAvgBytesPerSec (4 bytes):** A 32-bit unsigned integer. This field specifies average data-transfer rate for the format, in bytes per second.

**nBlockAlign (2 bytes):** A 16-bit unsigned integer. This field specifies the block alignment, in bytes. The block alignment is the minimum atomic unit of data for this audio format.

**wBitsPerSample (2 bytes):** A 16-bit unsigned integer. This field specifies the size, in bits, for each audio sample.

**cbSize (2 bytes):** A 16-bit unsigned integer specifying the size of the **ExtraFormatData** field.

**ExtraFormatData (variable):** Extra data specific to the audio format. The size of **ExtraFormatData**, in bytes, is **cbSize**. If **cbSize** is 0, this field MUST NOT exist. If **wFormatTag** is set to WAVE\_FORMAT\_EXTENSIBLE (0xFFFE), this field MUST be a [WAVEFORMAT\_EXTENSIBLE](#Section_84835e471f12470faacb80775c854b04) structure.

##### Extended Wave Format Structure (WAVEFORMAT\_EXTENSIBLE)

The Extended Wave Format Structure provides detailed [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) information. For more information, see [[MSDN-WAVEFMTEXT]](https://go.microsoft.com/fwlink/?LinkId=128854).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| wValidBitsPerSample/wSamplesPerBlock | | | | | | | | | | | | | | | | dwChannelMask | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | SubFormat (16 bytes) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | |

**wValidBitsPerSample/wSamplesPerBlock (2 bytes):** Depending on the format used, the 16–bit field before **dwChannelMask** can be interpreted as either **wValidBitsPerSample** or **wSamplesPerBlock**. **wValidBitsPerSample** is a 16-bit unsigned integer specifying how many bits are valid for each sample. This value MUST be less than or equal to the value of the **wBitsPerSample** field of the containing [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0). **wSamplesPerBlock** is a 16-bit unsigned integer specifying the number of samples contained in one compressed block of audio data. This value is used with compressed formats that have a fixed number of samples within each block. This value can be set to 0 if a variable number of samples is contained in each block of compressed audio data.

**dwChannelMask (4 bytes):** A 32-bit unsigned integer specifying from which audio channels to record. This field MUST be a bitwise OR of the following values.

| Speaker position | Flag bit |
| --- | --- |
| SPEAKER\_FRONT\_LEFT | 0x00000001 |
| SPEAKER\_FRONT\_RIGHT | 0x00000002 |
| SPEAKER\_FRONT\_CENTER | 0x00000004 |
| SPEAKER\_LOW\_FREQUENCY | 0x00000008 |
| SPEAKER\_BACK\_LEFT | 0x00000010 |
| SPEAKER\_BACK\_RIGHT | 0x00000020 |
| SPEAKER\_FRONT\_LEFT\_OF\_CENTER | 0x00000040 |
| SPEAKER\_FRONT\_RIGHT\_OF\_CENTER | 0x00000080 |
| SPEAKER\_BACK\_CENTER | 0x00000100 |
| SPEAKER\_SIDE\_LEFT | 0x00000200 |
| SPEAKER\_SIDE\_RIGHT | 0x00000400 |
| SPEAKER\_TOP\_CENTER | 0x00000800 |
| SPEAKER\_TOP\_FRONT\_LEFT | 0x00001000 |
| SPEAKER\_TOP\_FRONT\_CENTER | 0x00002000 |
| SPEAKER\_TOP\_FRONT\_RIGHT | 0x00004000 |
| SPEAKER\_TOP\_BACK\_LEFT | 0x00008000 |
| SPEAKER\_TOP\_BACK\_CENTER | 0x00010000 |
| SPEAKER\_TOP\_BACK\_RIGHT | 0x00020000 |

**SubFormat (16 bytes):** A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that specifies the audio format to use for recording data. This field MUST be set to KSDATAFORMAT\_SUBTYPE\_PCM ({00000001-0000-0010-8000-00aa00389b71}).

#### Open Reply PDU (MSG\_SNDIN\_OPEN\_REPLY)

The Open Reply PDU is sent by the client to convey the result of an attempt to open the audio capture device.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | Result | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**Header (1 byte):**  A [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_OPEN\_REPLY (0x04).

**Result (4 bytes):** An [**HRESULT**](#gt_799103ab-b3cb-4eab-8c55-322821b2b235) containing the status of the attempt to open the client-side [**Wave Capture Device**](#gt_7f57363c-bc2b-4055-b344-7c79b1dfe80f).

### Data Transfer Messages

The following sections contain the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension message syntax for transferring audio data from client to server. For more information, see section [3.2.5.2](#Section_3689bf5888cc4e2e91b84cdb60c84d78).

#### Incoming Data PDU (MSG\_SNDIN\_DATA\_INCOMING)

The Incoming Data PDU is sent by the client to indicate that the next packet sent MUST either be a [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f) or a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d). This [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) is used for diagnostic purposes. The time used to transmit a Data PDU/Sound Formats PDU can be found out by comparing the time between receiving a Data PDU/Sound Formats PDU and receiving an Incoming Data PDU. The server MAY use this time and the data size of a Data PDU/Sound Formats PDU to calculate how fast the network is transmitting data. The result of this calculation MAY then be used to determine the audio format to use.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | |

**Header (1 byte):**  An [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_DATA\_INCOMING (0x05).

#### Data PDU (MSG\_SNDIN\_DATA)

The Data PDU is sent by the client and contains a fragment of audio data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | Data (variable) (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**Header (1 byte):** A [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_DATA (0x06).

**Data (variable):** A variable-length field that contains audio data. The recipient can determine the length of the **Data** field by subtracting the length of the **Header** field from the length of the Data PDU. The **MSG\_SNDIN\_DATA** structure is encapsulated in the Dynamic VC Data First PDU ([[MS-RDPEDYC]](%5bMS-RDPEDYC%5d.pdf#Section_3bd530209b644c9a97fc90a79e7e1e06) section 2.2.3.1) and Dynamic VC Data PDU ([MS-RDPEDYC] section 2.2.3.2). The length of these PDUs is determined as described in [MS-RDPEDYC] sections 3.1.5.1 and 3.1.5.2.

### Format Change Messages

The following section specifies the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension message syntax for changing audio data encoding format. For more information, see section [3.2.5.1.7](#Section_4b3fa3699aa84776a96beff10b8c4fbc) and section [3.3.5.1.7](#Section_f745fff733ed475aa9b9f832cf902566).

#### Format Change PDU (MSG\_SNDIN\_FORMATCHANGE)

The Format Change PDU is sent by the server to request that the client change [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) and by the client to confirm a format change.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Header | | | | | | | | NewFormat | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | |

**Header (1 byte):**  An [SNDIN\_PDU](#Section_51b2d5c1294347b1b5948a04a53df7ec) header (section 2.2.1). The **MessageId** field of the SNDIN\_PDU header MUST be set to MSG\_SNDIN\_FORMATCHANGE (0x07).

**NewFormat (4 bytes):** A 32-bit unsigned integer that represents an index into the list of audio formats exchanged between the client and server during the initialization phase, as specified in section [3.1.1](#Section_d3856db25ae44909baba23f8df26fc6c). For more information about changing audio formats, see sections [3.2.5.3](#Section_799885286f8c48b4ab5b7434e7c11a8a) and [3.3.5.3](#Section_e33a314a2e9449a7a97b6d4784acf908).

# Protocol Details

## Common Details

### 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.

**Audio format list:** The client and server maintain a list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) that were agreed on during the initialization sequence throughout the duration of the protocol. The **NewFormat** field of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) and the **InitialFormat** field of the [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) are indices into this list. The index to the **audio format list** is zero-based, where the value zero refers to the first format in the list.

**Current audio format:** The client and server maintain the current audio formats. This is the format in which audio data MUST be encoded when transmitted from the client to the server. The **NewFormat** field of the Format Change PDU or the **InitialFormat** field of the Open PDU indicates the current audio format.

**Audio capture device state:** The audio capture device has two states: **opened** and **not opened**. Initially the state is **not opened**. The server sends an Open PDU to ask the client to open the audio capture device. The client sends an [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) to tell the server the result of opening the capture device. If the capture device is opened successfully, the audio input device state becomes **opened**.

**Audio data ready:** The client sends an [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) to tell the server that it is ready to send data. The server maintains the **audio data ready** state to prepare to receive a [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f).

**Protocol version:** The client and server store the protocol version negotiated during the initialization sequence.

### Timers

None.

### Initialization

Before this protocol begins, the [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3) transport MUST be established as specified in section [2.1](#Section_ff2fa50258574e7996985f17eb1e8a1e). The server and client also need to negotiate the protocol version and a common list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) by exchanging [Version PDUs](#Section_b25cb6b21ab0405698a371b85d9b66b4) and [Sound Formats PDUs](#Section_c4e53155f57743c6ac1dfad49540d02d). Once the protocol version and audio formats list have been exchanged, the server sends an [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) to the client to indicate that the server requests to start recording. The client attempts to start recording and returns the result to the server in an [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) message.

### Higher-Layer Triggered Events

#### Recording Audio

When the server attempts to start recording audio, for example the server runs Sound Recorder, the client MUST redirect the audio data. If the initialization sequence has not happened, the server MUST begin the initialization sequence.

### Message Processing Events and Sequencing Rules

The state transition diagram in the following figure summarizes the message sequencing rules for the Remote Desktop Protocol: Audio Input Redirection Virtual Channel Extension. Further details on sending and processing the messages are included in sections [3.2.5](#Section_24ff9db29efc49c095c2aa2db2a9baf7) and [3.3.5](#Section_9315b145238b4349bf58f4cf693aacde). Malformed packets that do not meet the specifications outlined in section [2.2](#Section_2451e5b379ec4f6a88cf5419737cd1e4), unrecognized packets, and out-of-sequence packets MUST be ignored by the server and the client. There are no time-outs for receiving a reply for any request.

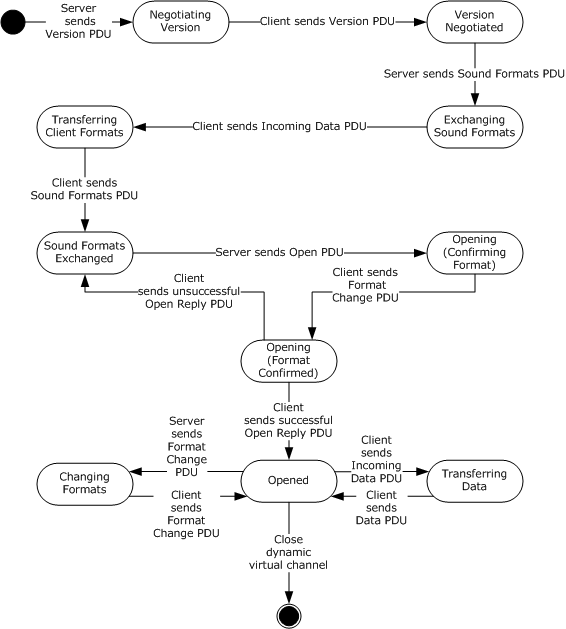


Figure 4: State transition diagram

#### Protocol Initialization

The protocol begins when the server opens the [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3). The first [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) sent is always the [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4).

#### Protocol Termination

The protocol is terminated when the [**dynamic virtual channel**](#gt_fda9b8e9-bec6-4e04-9833-83bee294f3d3) is closed. This can occur at any time during the protocol sequence and not merely on the state transition labeled "Close dynamic virtual channel". The close can be initiated by either the [**RDP server**](#gt_2041833b-a196-4b8a-b8a8-8ed5424415b9) or the [**RDP client**](#gt_b58fec62-3d34-4209-94eb-e933b4cd4513), but it is normally initiated by the RDP client.

### Timer Events

None.

### Other Local Events

None.

## Client Details

### Abstract Data Model

The abstract data model is specified in section [3.1.1](#Section_d3856db25ae44909baba23f8df26fc6c).

### Timers

None.

### Initialization

Initialization is specified in section [3.1.3](#Section_547a89a5425c4fcca548dccba6d89dd6).

### Higher-Layer Triggered Events

See section [3.1.4](#Section_63a3a1487f5b403aa2e350ddd6755fae) for more information.

### Message Processing Events and Sequencing Rules

#### Initialization Sequence

##### Processing a Version PDU

The structure and fields of the [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) are specified in section 2.2.2.1.

The Version PDU MUST be the first [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) received by the client. The client MUST store the protocol version and use this to determine which messages are supported by the server.

##### Sending a Version PDU

The structure and fields of the [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) are specified in section 2.2.2.1.

The client MUST acknowledge the Version PDU sent by the server by sending its own Version PDU to the server.

##### Processing a Sound Formats PDU

The structure and fields of the [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) are specified in section 2.2.2.2.

The client uses the [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) sent by the server when selecting which formats will be used.

##### Sending an Incoming Data PDU

The structure and format of the [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) are specified in section 2.2.3.1.

The client MUST send an Incoming Data PDU before responding to the server with a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d).

##### Sending a Sound Formats PDU

The structure and fields of the [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) are specified in section 2.2.2.2.

The client MUST respond to the Sound Formats PDU sent by the server by replying with its own Sound Formats PDU. The list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) that the client sends back to the server MUST be a subset of the list sent by the server. Formats that are not in the server's list MUST NOT be included in the client's audio format list.

The list sent by the client establishes the allowed audio formats for the current protocol session. These formats MUST be referenced by the **NewFormat** field of the [Format Change PDUs](#Section_8d51c8bc5a7941c897c4b0a9daaba866) and the **initialFormat** field of the [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0). A value of *I* refers to the *I*th format of this list and means that audio data MUST be encoded in the *I*th format in the list.

##### Processing an Open PDU

The structure and fields of the [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) are specified in section 2.2.2.3.

The server sends the Open PDU to indicate that it is ready to start receiving audio data. The Open PDU specifies two [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044). The first format is specified by the **InitialFormat** field and is an index into the negotiated audio formats list. This format is the audio format that the client MUST use to encode audio data and send to the server. The format is used to encode audio data that can be changed by a [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) later.

The second format is specified by the **wFormatTag**, **nChannels**, **nSamplesPerSec**, **nAvgBytesPerSec**, **nBlockAlign**, **nBitsPerSample**, **cbSize**, and **ExtraFormatData** fields. These fields specify the format that SHOULD be used to capture data from the actual audio input device.

##### Sending a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

Client sends Format Change PDU after it receives a Format Change PDU from the server. The **NewFormat** field MUST be set to the same value as the **NewFormat** field in the server’s Format Change PDU.

Client also sends Format Change PDU after it receives [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) from server. Client sends this Format Change PDU before sending [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91), and the **NewFormat** field MUST be set to the same value as the **initialFormat** field in the server’s Open PDU.

##### Sending an Open Reply PDU

The structure and fields of the [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) are specified in section 2.2.2.4.

After processing an [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0), the client MUST attempt to begin recording from a local audio input device. The client MUST inform the server of the result by sending an appropriate [**HRESULT**](#gt_799103ab-b3cb-4eab-8c55-322821b2b235) code in the **Result** field. An HRESULT code is an error code when its first bit is set to 1. See [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1 for a description of HRESULT.

#### Data Transfer Sequence

##### Sending an Incoming Data PDU

The structure and format of the [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) are specified in section 2.2.3.1.

Once the client has captured a packet of audio data to send to the server, the client MUST send an Incoming Data PDU immediately before sending each [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f).

##### Sending a Data PDU

The structure and format of the [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f) are specified in section 2.2.3.2.

This message MUST be sent immediately after sending an [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92). The client MUST fill the **Data** field with an audio packet encoded using the current [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044).

#### Format Change Sequence

##### Processing a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

The Format Change PDU indicates that the server requires the client to use a different [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) to encode audio data. The client MUST change the current audio format to the value contained in the **NewFormat** field, which is an index into the negotiated audio formats list. The client MUST also confirm the format change by replying with a Format Change PDU. After sending a Format Change PDU, all audio data MUST be encoded using the new format until the server requests another format change.

##### Sending a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

After processing a Format Change PDU, the client MUST confirm the format change by replying with its own Format Change PDU. The **NewFormat** field MUST be set to the new value of the current [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044). After sending a Format Change PDU, all audio data MUST be encoded using the new format, until the server requests another format change. The format change operation MUST always succeed.

### Timer Events

None.

### Other Local Events

None.

## Server Details

### Abstract Data Model

The abstract data model is specified in section [3.1.1](#Section_d3856db25ae44909baba23f8df26fc6c).

### Timers

After sending [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4), the server MAY use a time-out when waiting for the client's Version PDU.[<2>](#Appendix_A_2" \o "Product behavior note 2)

After sending [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d), the server MAY use a time-out when waiting for the client's Sound Formats PDU.[<3>](#Appendix_A_3" \o "Product behavior note 3)

After sending [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0), the server MAY use a time-out when waiting for the client's Open Reply PDU.[<4>](#Appendix_A_4" \o "Product behavior note 4)

### Initialization

Initialization is specified in section [3.1.3](#Section_547a89a5425c4fcca548dccba6d89dd6).

### Higher-Layer Triggered Events

See section [3.1.4](#Section_63a3a1487f5b403aa2e350ddd6755fae) for more information.

### Message Processing Events and Sequencing Rules

#### Initialization Sequence

##### Sending a Version PDU

The structure and fields of the [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) are specified in section 2.2.2.1.

The Version PDU MUST be the first [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) sent by the server.

##### Processing a Version PDU

The structure and fields of the [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4) are specified in section 2.2.2.1.

The server MUST store the protocol version sent by the client.

##### Sending a Sound Formats PDU

The structure and fields of the [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) are specified in section 2.2.2.2.

The server MUST send a Sound Formats PDU after processing the client's [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4). The Sound Formats PDU informs the client of the [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) that are supported by the server.

##### Processing an Incoming Data PDU

The structure and format of the [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) are specified in section 2.2.3.1.

The client sends an Incoming Data PDU to inform the server that it is about to respond with its [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d). This [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) is used for diagnostic purposes. An Incoming Data PDU also precedes a Data PDU (see section [3.3.5.2.1](#Section_7e40602348f34f9dbe6c1319b3c656f5)).

##### Processing a Sound Formats PDU

The structure and fields of the [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) are specified in section 2.2.2.2.

The server MUST store the list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) sent by the client. This list is the final list of formats that can be used during this protocol session.

##### Sending an Open PDU

The structure and fields of the [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0) are specified in section 2.2.2.3.

The server sends an Open PDU after the [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) exchange is completed. The fields of the Open PDU MUST be set according to section [3.2.5.1.6](#Section_ae0639e696f64077a436d2ec14918aab).

##### Processing a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

Before sending the [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) and beginning the transmission of audio data, the client sends a Format Change PDU. This [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) confirms the initial format chosen by the server and informs the server that all audio data MUST initially be encoded in the format specified by the **NewFormat** field.

##### Processing an Open Reply PDU

The structure and fields of the [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91) are specified in section 2.2.2.4.

The Open Reply PDU contains the result of the client’s attempt to open the audio capture device. If the **Result** field is a success code, the client will begin sending audio data. If the **Result** field is an error code, the client MUST not send audio data. If the **Result** field is an error code, then the server MAY send an additional [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0).

#### Data Transfer Sequence

##### Processing an Incoming Data PDU

The structure and format of the [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92) are specified in section 2.2.3.1.

The Incoming Data PDU indicates that the client is about to send a packet of audio data. The server uses this information for diagnostic purposes. An Incoming Data PDU also precedes a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d) (see section [3.3.5.1.4](#Section_044e7cdc619a43e7a1f262d37289d332)).

##### Processing a Data PDU

The structure and format of the [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f) are specified in section 2.2.3.2.

After receiving a Data PDU, the server decodes the audio data according to the current [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044). The decoded audio is then provided to server-side applications that are recording audio.

#### Format Change Sequence

##### Sending a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

When the server determines that a new format is necessary, it MUST choose one from the list of [**audio formats**](#gt_119a5832-c03a-4f9a-aa57-b52140751044) negotiated in the initialization sequence. The server MUST set the **NewFormat** field of the Format Change PDU to the index into the list of the new audio formats.

The server MUST continue to decode any audio data that arrives using the old audio format until it receives the confirmation Format Change PDU from the client.

##### Processing a Format Change PDU

The structure and format of the [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866) are specified in section 2.2.4.1.

The client sends a Format Change PDU to confirm the format change. After receiving acknowledgment, server MUST set **Current audio format** as **NewFormat**. From this point, all audio data sent by the client MUST be encoded using the new [**audio format**](#gt_119a5832-c03a-4f9a-aa57-b52140751044). The server MUST decode all audio packets that arrive after this [**PDU**](#gt_34715e6f-1612-4b2d-a4bb-3305c56e96f5) according to the new audio format.

### Termination

Termination is specified in section [3.1.5.2](#Section_28b18dedb035469da2dbf6d17f9f4114).

### Timer Events

None.

### Other Local Events

None.

# Protocol Examples

## Annotated Initialization Sequence

The following is an annotated dump of an initialization sequence, as described in section [1.3.1](#Section_c287fe7796ac42d6a62cae20d23ca2a5).

### Server Version PDU

The following is an annotated dump of a [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4).

1. 00000000 01 01 00 00 00
2. 01 -> MSG\_SNDIN\_PDU\_HEADER::MessageId = MSG\_SNDIN\_VERSION (1)
3. 01 00 00 00 -> MSG\_SNDIN\_VERSION::Version = 1

### Client Version PDU

The following is an annotated dump of a [Version PDU](#Section_b25cb6b21ab0405698a371b85d9b66b4).

1. 00000000 01 01 00 00 00
2. 01 -> MSG\_SNDIN\_PDU\_HEADER::MessageId = MSG\_SNDIN\_VERSION (1)
3. 01 00 00 00 -> MSG\_SNDIN\_VERSION::Version = 1

### Server Sound Formats PDU

The following is an annotated dump of a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d).

1. 00000000 02 15 00 00 00 00 00 00 80 01 00 02 00 44 ac 00
2. 00000010 00 10 b1 02 00 04 00 10 00 00 00 02 00 02 00 44
3. 00000020 ac 00 00 47 ad 00 00 00 08 04 00 20 00 f4 07 07
4. 00000030 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40
5. 00000040 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11 00 02
6. 00000050 00 44 ac 00 00 db ac 00 00 00 08 04 00 02 00 f9
7. 00000060 07 02 00 02 00 22 56 00 00 27 57 00 00 00 04 04
8. 00000070 00 20 00 f4 03 07 00 00 01 00 00 00 02 00 ff 00
9. 00000080 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88
10. 00000090 01 18 ff 11 00 02 00 22 56 00 00 b9 56 00 00 00
11. 000000a0 04 04 00 02 00 f9 03 02 00 01 00 44 ac 00 00 a3
12. 000000b0 56 00 00 00 04 04 00 20 00 f4 07 07 00 00 01 00
13. 000000c0 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00
14. 000000d0 00 cc 01 30 ff 88 01 18 ff 11 00 01 00 44 ac 00
15. 000000e0 00 6d 56 00 00 00 04 04 00 02 00 f9 07 02 00 02
16. 000000f0 00 11 2b 00 00 19 2c 00 00 00 02 04 00 20 00 f4
17. 00000100 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
18. 00000110 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
19. 00000120 00 02 00 11 2b 00 00 a9 2b 00 00 00 02 04 00 02
20. 00000130 00 f9 01 02 00 01 00 22 56 00 00 93 2b 00 00 00
21. 00000140 02 04 00 20 00 f4 03 07 00 00 01 00 00 00 02 00
22. 00000150 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30
23. 00000160 ff 88 01 18 ff 11 00 01 00 22 56 00 00 5c 2b 00
24. 00000170 00 00 02 04 00 02 00 f9 03 31 00 01 00 44 ac 00
25. 00000180 00 fd 22 00 00 41 00 00 00 02 00 40 01 02 00 02
26. 00000190 00 40 1f 00 00 00 20 00 00 00 02 04 00 20 00 f4
27. 000001a0 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
28. 000001b0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
29. 000001c0 00 02 00 40 1f 00 00 ae 1f 00 00 00 02 04 00 02
30. 000001d0 00 f9 01 02 00 01 00 11 2b 00 00 0c 16 00 00 00
31. 000001e0 01 04 00 20 00 f4 01 07 00 00 01 00 00 00 02 00
32. 000001f0 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30
33. 00000200 ff 88 01 18 ff 11 00 01 00 11 2b 00 00 d4 15 00
34. 00000210 00 00 01 04 00 02 00 f9 01 31 00 01 00 22 56 00
35. 00000220 00 7e 11 00 00 41 00 00 00 02 00 40 01 02 00 01
36. 00000230 00 40 1f 00 00 00 10 00 00 00 01 04 00 20 00 f4
37. 00000240 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
38. 00000250 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
39. 00000260 00 01 00 40 1f 00 00 d7 0f 00 00 00 01 04 00 02
40. 00000270 00 f9 01 31 00 01 00 11 2b 00 00 bf 08 00 00 41
41. 00000280 00 00 00 02 00 40 01 31 00 01 00 40 1f 00 00 59
42. 00000290 06 00 00 41 00 00 00 02 00 40 01
43. 02 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_FORMATS (2)
44. 15 00 00 00 -> MSG\_SNDIN\_FORMATS::NumFormats = 0x15 = 21
45. 00 00 00 80 -> MSG\_SNDIN\_FORMATS::cbSizeFormatsPacket = 0x80000000 = 2147483648 bytes
46. 01 00 02 00 44 ac 00 00 10 b1 02 00 04 00 10 00 00 00 -> AUDIO\_FORMAT
47. 01 00 -> AUDIO\_FORMAT::wFormatTag = 0x1 = 1
48. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
49. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
50. 10 b1 02 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b110 = 176400
51. 04 00 -> AUDIO\_FORMAT::nBlockAlign = 0x4 = 4
52. 10 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x10 = 16
53. 00 00 -> AUDIO\_FORMAT::cbSize = 0x0 = 0
54. 02 00 02 00 44 ac 00 00 47 ad 00 00 00 08 04 00 20 00 f4 07 07 00 00 01 00 00 00
55. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
56. AUDIO\_FORMAT
57. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
58. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
59. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
60. 47 ad 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xad47 = 44359
61. 00 08 -> AUDIO\_FORMAT::nBlockAlign = 0x800 = 2048
62. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
63. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
64. f4 07 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
65. 01 30 ff 88 01 18 ff -> data
66. 11 00 02 00 44 ac 00 00 db ac 00 00 00 08 04 00 02 00 f9 07 -> AUDIO\_FORMAT
67. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
68. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
69. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
70. db ac 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xacdb = 44251
71. 00 08 -> AUDIO\_FORMAT::nBlockAlign = 0x800 = 2048
72. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
73. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
74. f9 07 -> data
75. 02 00 02 00 22 56 00 00 27 57 00 00 00 04 04 00 20 00 f4 03 07 00 00 01 00 00 00
76. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
77. AUDIO\_FORMAT
78. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
79. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
80. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
81. 27 57 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x5727 = 22311
82. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
83. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
84. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
85. f4 03 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
86. 01 30 ff 88 01 18 ff -> data
87. 11 00 02 00 22 56 00 00 b9 56 00 00 00 04 04 00 02 00 f9 03 -> AUDIO\_FORMAT
88. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
89. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
90. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
91. b9 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x56b9 = 22201
92. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
93. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
94. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
95. f9 03 -> data
96. 02 00 01 00 44 ac 00 00 a3 56 00 00 00 04 04 00 20 00 f4 07 07 00 00 01 00 00 00
97. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
98. AUDIO\_FORMAT
99. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
100. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
101. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
102. a3 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x56a3 = 22179
103. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
104. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
105. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
106. f4 07 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
107. 01 30 ff 88 01 18 ff -> data
108. 11 00 01 00 44 ac 00 00 6d 56 00 00 00 04 04 00 02 00 f9 07 -> AUDIO\_FORMAT
109. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
110. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
111. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
112. 6d 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x566d = 22125
113. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
114. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
115. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
116. f9 07 -> data
117. 02 00 02 00 11 2b 00 00 19 2c 00 00 00 02 04 00 20 00 f4 01 07 00 00 01 00 00 00
118. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
119. AUDIO\_FORMAT
120. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
121. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
122. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
123. 19 2c 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2c19 = 11289
124. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
125. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
126. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
127. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
128. 01 30 ff 88 01 18 ff -> data
129. 11 00 02 00 11 2b 00 00 a9 2b 00 00 00 02 04 00 02 00 f9 01 -> AUDIO\_FORMAT
130. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
131. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
132. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
133. a9 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2ba9 = 11177
134. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
135. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
136. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
137. f9 01 -> data
138. 02 00 01 00 22 56 00 00 93 2b 00 00 00 02 04 00 20 00 f4 03 07 00 00 01 00 00 00
139. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff -> AUDIO\_FORMAT
140. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
141. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
142. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
143. 93 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b93 = 11155
144. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
145. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
146. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
147. f4 03 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
148. 01 30 ff 88 01 18 ff -> data
149. 11 00 01 00 22 56 00 00 5c 2b 00 00 00 02 04 00 02 00 f9 03 -> AUDIO\_FORMAT
150. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
151. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
152. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
153. 5c 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b5c = 11100
154. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
155. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
156. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
157. f9 03 -> data
158. 31 00 01 00 44 ac 00 00 fd 22 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
159. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
160. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
161. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
162. fd 22 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x22fd = 8957
163. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
164. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
165. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
166. 40 01 -> data
167. 02 00 02 00 40 1f 00 00 00 20 00 00 00 02 04 00 20 00 f4 01 07 00 00 01 00 00 00
168. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff -> AUDIO\_FORMAT
169. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
170. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
171. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
172. 00 20 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2000 = 8192
173. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
174. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
175. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
176. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
177. 01 30 ff 88 01 18 ff -> data
178. 11 00 02 00 40 1f 00 00 ae 1f 00 00 00 02 04 00 02 00 f9 01 -> AUDIO\_FORMAT
179. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
180. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
181. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
182. ae 1f 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x1fae = 8110
183. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
184. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
185. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
186. f9 01 -> data
187. 02 00 01 00 11 2b 00 00 0c 16 00 00 00 01 04 00 20 00 f4 01 07 00 00 01 00 00 00
188. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff -> AUDIO\_FORMAT
189. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
190. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
191. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
192. 0c 16 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x160c = 5644
193. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
194. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
195. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
196. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
197. 01 30 ff 88 01 18 ff -> data
198. 11 00 01 00 11 2b 00 00 d4 15 00 00 00 01 04 00 02 00 f9 01 -> AUDIO\_FORMAT
199. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
200. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
201. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
202. d4 15 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x15d4 = 5588
203. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
204. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
205. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
206. f9 01 -> data
207. 31 00 01 00 22 56 00 00 7e 11 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
208. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
209. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
210. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
211. 7e 11 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x117e = 4478
212. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
213. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
214. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
215. 40 01 -> data
216. 02 00 01 00 40 1f 00 00 00 10 00 00 00 01 04 00 20 00 f4 01 07 00 00 01 00 00 00
217. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff -> AUDIO\_FORMAT
218. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
219. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
220. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
221. 00 10 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x1000 = 4096
222. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
223. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
224. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
225. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
226. 01 30 ff 88 01 18 ff -> data
227. 11 00 01 00 40 1f 00 00 d7 0f 00 00 00 01 04 00 02 00 f9 01 -> AUDIO\_FORMAT
228. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
229. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
230. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
231. d7 0f 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xfd7 = 4055
232. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
233. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
234. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
235. f9 01 -> data
236. 31 00 01 00 11 2b 00 00 bf 08 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
237. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
238. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
239. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
240. bf 08 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x8bf = 2239
241. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
242. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
243. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
244. 40 01 -> data
245. 31 00 01 00 40 1f 00 00 59 06 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
246. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
247. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
248. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
249. 59 06 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x659 = 1625
250. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
251. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
252. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
253. 40 01 -> data

### Incoming Data PDU

The following is an annotated dump of an [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92).

1. 00000000 05
2. 05 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_DATA\_INCOMING (5)

### Client Sound Formats PDU

The following is an annotated dump of a [Sound Formats PDU](#Section_c4e53155f57743c6ac1dfad49540d02d).

1. 00000000 02 15 00 00 00 9b 02 00 00 01 00 02 00 44 ac 00
2. 00000010 00 10 b1 02 00 04 00 10 00 00 00 02 00 02 00 44
3. 00000020 ac 00 00 47 ad 00 00 00 08 04 00 20 00 f4 07 07
4. 00000030 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40
5. 00000040 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11 00 02
6. 00000050 00 44 ac 00 00 db ac 00 00 00 08 04 00 02 00 f9
7. 00000060 07 02 00 02 00 22 56 00 00 27 57 00 00 00 04 04
8. 00000070 00 20 00 f4 03 07 00 00 01 00 00 00 02 00 ff 00
9. 00000080 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88
10. 00000090 01 18 ff 11 00 02 00 22 56 00 00 b9 56 00 00 00
11. 000000a0 04 04 00 02 00 f9 03 02 00 01 00 44 ac 00 00 a3
12. 000000b0 56 00 00 00 04 04 00 20 00 f4 07 07 00 00 01 00
13. 000000c0 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00
14. 000000d0 00 cc 01 30 ff 88 01 18 ff 11 00 01 00 44 ac 00
15. 000000e0 00 6d 56 00 00 00 04 04 00 02 00 f9 07 02 00 02
16. 000000f0 00 11 2b 00 00 19 2c 00 00 00 02 04 00 20 00 f4
17. 00000100 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
18. 00000110 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
19. 00000120 00 02 00 11 2b 00 00 a9 2b 00 00 00 02 04 00 02
20. 00000130 00 f9 01 02 00 01 00 22 56 00 00 93 2b 00 00 00
21. 00000140 02 04 00 20 00 f4 03 07 00 00 01 00 00 00 02 00
22. 00000150 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30
23. 00000160 ff 88 01 18 ff 11 00 01 00 22 56 00 00 5c 2b 00
24. 00000170 00 00 02 04 00 02 00 f9 03 31 00 01 00 44 ac 00
25. 00000180 00 fd 22 00 00 41 00 00 00 02 00 40 01 02 00 02
26. 00000190 00 40 1f 00 00 00 20 00 00 00 02 04 00 20 00 f4
27. 000001a0 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
28. 000001b0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
29. 000001c0 00 02 00 40 1f 00 00 ae 1f 00 00 00 02 04 00 02
30. 000001d0 00 f9 01 02 00 01 00 11 2b 00 00 0c 16 00 00 00
31. 000001e0 01 04 00 20 00 f4 01 07 00 00 01 00 00 00 02 00
32. 000001f0 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30
33. 00000200 ff 88 01 18 ff 11 00 01 00 11 2b 00 00 d4 15 00
34. 00000210 00 00 01 04 00 02 00 f9 01 31 00 01 00 22 56 00
35. 00000220 00 7e 11 00 00 41 00 00 00 02 00 40 01 02 00 01
36. 00000230 00 40 1f 00 00 00 10 00 00 00 01 04 00 20 00 f4
37. 00000240 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0
38. 00000250 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff 11
39. 00000260 00 01 00 40 1f 00 00 d7 0f 00 00 00 01 04 00 02
40. 00000270 00 f9 01 31 00 01 00 11 2b 00 00 bf 08 00 00 41
41. 00000280 00 00 00 02 00 40 01 31 00 01 00 40 1f 00 00 59
42. 00000290 06 00 00 41 00 00 00 02 00 40 01 00 00 00 00 00
43. 000002a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
44. ...
45. 000003f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
46. 02 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_FORMATS (2)
47. 15 00 00 00 -> MSG\_SNDIN\_FORMATS::NumFormats = 0x15 = 21
48. 9b 02 00 00 -> MSG\_SNDIN\_FORMATS::cbSizeFormatsPacket = 0x29b = 667 bytes
49. 01 00 02 00 44 ac 00 00 10 b1 02 00 04 00 10 00 00 00 -> AUDIO\_FORMAT
50. 01 00 -> AUDIO\_FORMAT::wFormatTag = 0x1 = 1
51. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
52. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
53. 10 b1 02 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b110 = 176400
54. 04 00 -> AUDIO\_FORMAT::nBlockAlign = 0x4 = 4
55. 10 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x10 = 16
56. 00 00 -> AUDIO\_FORMAT::cbSize = 0x0 = 0
57. 02 00 02 00 44 ac 00 00 47 ad 00 00 00 08 04 00 20 00 f4 07 07 00 00 01 00 00 00
58. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
59. AUDIO\_FORMAT
60. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
61. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
62. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
63. 47 ad 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xad47 = 44359
64. 00 08 -> AUDIO\_FORMAT::nBlockAlign = 0x800 = 2048
65. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
66. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
67. f4 07 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
68. 01 30 ff 88 01 18 ff -> data
69. 11 00 02 00 44 ac 00 00 db ac 00 00 00 08 04 00 02 00 f9 07 -> AUDIO\_FORMAT
70. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
71. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
72. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
73. db ac 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xacdb = 44251
74. 00 08 -> AUDIO\_FORMAT::nBlockAlign = 0x800 = 2048
75. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
76. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
77. f9 07 -> data
78. 02 00 02 00 22 56 00 00 27 57 00 00 00 04 04 00 20 00 f4 03 07 00 00 01 00 00 00
79. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff
80. -> AUDIO\_FORMAT
81. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
82. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
83. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
84. 27 57 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x5727 = 22311
85. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
86. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
87. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
88. f4 03 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
89. 01 30 ff 88 01 18 ff -> data
90. 11 00 02 00 22 56 00 00 b9 56 00 00 00 04 04 00 02 00 f9 03 -> AUDIO\_FORMAT
91. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
92. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
93. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
94. b9 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x56b9 = 22201
95. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
96. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
97. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
98. f9 03 -> data
99. 02 00 01 00 44 ac 00 00 a3 56 00 00 00 04 04 00 20 00 f4 07 07 00 00 01 00 00 00
100. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
101. AUDIO\_FORMAT
102. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
103. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
104. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
105. a3 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x56a3 = 22179
106. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
107. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
108. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
109. f4 07 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
110. 01 30 ff 88 01 18 ff -> data
111. 11 00 01 00 44 ac 00 00 6d 56 00 00 00 04 04 00 02 00 f9 07 -> AUDIO\_FORMAT
112. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
113. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
114. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
115. 6d 56 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x566d = 22125
116. 00 04 -> AUDIO\_FORMAT::nBlockAlign = 0x400 = 1024
117. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
118. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
119. f9 07 -> data
120. 02 00 02 00 11 2b 00 00 19 2c 00 00 00 02 04 00 20 00 f4 01 07 00 00 01 00 00 00
121. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
122. AUDIO\_FORMAT
123. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
124. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
125. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
126. 19 2c 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2c19 = 11289
127. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
128. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
129. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
130. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
131. 01 30 ff 88 01 18 ff -> data
132. 11 00 02 00 11 2b 00 00 a9 2b 00 00 00 02 04 00 02 00 f9 01 -> AUDIO\_FORMAT
133. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
134. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
135. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
136. a9 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2ba9 = 11177
137. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
138. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
139. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
140. f9 01 -> data
141. 02 00 01 00 22 56 00 00 93 2b 00 00 00 02 04 00 20 00 f4 03 07 00 00 01 00 00 00
142. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
143. AUDIO\_FORMAT
144. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
145. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
146. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
147. 93 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b93 = 11155
148. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
149. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
150. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
151. f4 03 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
152. 01 30 ff 88 01 18 ff -> data
153. 11 00 01 00 22 56 00 00 5c 2b 00 00 00 02 04 00 02 00 f9 03 -> AUDIO\_FORMAT
154. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
155. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
156. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
157. 5c 2b 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2b5c = 11100
158. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
159. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
160. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
161. f9 03 -> data
162. 31 00 01 00 44 ac 00 00 fd 22 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
163. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
164. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
165. 44 ac 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0xac44 = 44100
166. fd 22 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x22fd = 8957
167. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
168. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
169. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
170. 40 01 -> data
171. 02 00 02 00 40 1f 00 00 00 20 00 00 00 02 04 00 20 00 f4 01 07 00 00 01 00 00 00
172. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
173. AUDIO\_FORMAT
174. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
175. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
176. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
177. 00 20 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x2000 = 8192
178. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
179. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
180. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
181. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
182. 01 30 ff 88 01 18 ff -> data
183. 11 00 02 00 40 1f 00 00 ae 1f 00 00 00 02 04 00 02 00 f9 01 -> AUDIO\_FORMAT
184. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
185. 02 00 -> AUDIO\_FORMAT::nChannels = 0x2 = 2
186. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
187. ae 1f 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x1fae = 8110
188. 00 02 -> AUDIO\_FORMAT::nBlockAlign = 0x200 = 512
189. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
190. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
191. f9 01 -> data
192. 02 00 01 00 11 2b 00 00 0c 16 00 00 00 01 04 00 20 00 f4 01 07 00 00 01 00 00 00
193. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
194. AUDIO\_FORMAT
195. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
196. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
197. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
198. 0c 16 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x160c = 5644
199. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
200. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
201. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
202. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
203. 01 30 ff 88 01 18 ff -> data
204. 11 00 01 00 11 2b 00 00 d4 15 00 00 00 01 04 00 02 00 f9 01 -> AUDIO\_FORMAT
205. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
206. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
207. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
208. d4 15 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x15d4 = 5588
209. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
210. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
211. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
212. f9 01 -> data
213. 31 00 01 00 22 56 00 00 7e 11 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
214. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
215. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
216. 22 56 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x5622 = 22050
217. 7e 11 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x117e = 4478
218. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
219. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
220. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
221. 40 01 -> data
222. 02 00 01 00 40 1f 00 00 00 10 00 00 00 01 04 00 20 00 f4 01 07 00 00 01 00 00 00
223. 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff ->
224. AUDIO\_FORMAT
225. 02 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_ADPCM (2)
226. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
227. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
228. 00 10 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x1000 = 4096
229. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
230. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
231. 20 00 -> AUDIO\_FORMAT::cbSize = 0x20 = 32
232. f4 01 07 00 00 01 00 00 00 02 00 ff 00 00 00 00 c0 00 40 00 f0 00 00 00 cc
233. 01 30 ff 88 01 18 ff -> data
234. 11 00 01 00 40 1f 00 00 d7 0f 00 00 00 01 04 00 02 00 f9 01 -> AUDIO\_FORMAT
235. 11 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_DVI\_ADPCM (17)
236. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
237. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
238. d7 0f 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0xfd7 = 4055
239. 00 01 -> AUDIO\_FORMAT::nBlockAlign = 0x100 = 256
240. 04 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x4 = 4
241. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
242. f9 01 -> data
243. 31 00 01 00 11 2b 00 00 bf 08 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
244. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
245. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
246. 11 2b 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x2b11 = 11025
247. bf 08 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x8bf = 2239
248. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
249. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
250. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
251. 40 01 -> data
252. 31 00 01 00 40 1f 00 00 59 06 00 00 41 00 00 00 02 00 40 01 -> AUDIO\_FORMAT
253. 31 00 -> AUDIO\_FORMAT::wFormatTag = WAVE\_FORMAT\_GSM610 (49)
254. 01 00 -> AUDIO\_FORMAT::nChannels = 0x1 = 1
255. 40 1f 00 00 -> AUDIO\_FORMAT::nSamplesPerSecond = 0x1f40 = 8000
256. 59 06 00 00 -> AUDIO\_FORMAT::nAvgBytesPerSecond = 0x659 = 1625
257. 41 00 -> AUDIO\_FORMAT::nBlockAlign = 0x41 = 65
258. 00 00 -> AUDIO\_FORMAT::wBitsPerSample = 0x0 = 0
259. 02 00 -> AUDIO\_FORMAT::cbSize = 0x2 = 2
260. 40 01 -> data
261. 00... 00 00 00 -> MSG\_SNDIN\_OPEN::ExtraData

### Open PDU

The following is an annotated dump of an [Open PDU](#Section_97db7244249b4a79ad507e45e2f760a0).

1. 00000000 03 9d 08 00 00 0b 00 00 00 fe ff 02 00 44 ac 00
2. 00000010 00 10 b1 02 00 04 00 10 00 16 00 10 00 03 00 00
3. 00000020 00 01 00 00 00 00 00 10 00 80 00 00 aa 00 38 9b
4. 00000030 71
5. 03 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_OPEN (3)
6. 9d 08 00 00 -> MSG\_SNDIN\_OPEN::FramesPerPacket = 0x89d = 2205
7. 0b 00 00 00 -> MSG\_SNDIN\_OPEN::InitialFormat = 0xb = 11
8. fe ff -> MSG\_SNDIN\_OPEN::wFormatTag = WAVE\_FORMAT\_EXTENSIBLE (65534)
9. 02 00 -> MSG\_SNDIN\_OPEN::nChannels = 0x2 = 2
10. 44 ac 00 00 -> MSG\_SNDIN\_OPEN::nSamplesPerSecond = 0xac44 = 44100
11. 10 b1 02 00 -> MSG\_SNDIN\_OPEN::nAvgBytesPerSecond = 0x2b110 = 176400
12. 04 00 -> MSG\_SNDIN\_OPEN::nBlockAlign = 0x4 = 4
13. 10 00 -> MSG\_SNDIN\_OPEN::wBitsPerSample = 0x10 = 16
14. 16 00 -> MSG\_SNDIN\_OPEN::cbSize = 0x16 = 22
15. 10 00 03 00 00 00 01 00 00 00 00 00 10 00 80 00 00 aa 00 38 9b 71 -> ExtraFormatData
16. 10 00 -> WAVEFORMAT\_EXTENSIBLE::wValidBitsPerSample = 0x10 = 16
17. 03 00 00 00 -> WAVEFORMAT\_EXTENSIBLE::dwChannelMask = 0x3
18. = SPEAKER\_FRONT\_LEFT | SPEAKER\_FRONT\_RIGHT
19. 01 00 00 00 00 00 10 00 80 00 00 aa 00 38 9b 71 -> WAVEFORMAT\_EXTENSIBLE::SubFormat
20. = KSDATAFORMAT\_SUBTYPE\_PCM {00000001-0000-0010-8000-00aa00389b71}

### Format Change PDU

The following is an annotated [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866).

1. 00000000 07 0b 00 00 00
2. 07 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_FORMATCHANGE (7)
3. 0b 00 00 00 -> MSG\_SNDIN\_FORMATCHANGE::NewFormat = 0xb = 11

### Open Reply PDU

The following is an annotated dump of an [Open Reply PDU](#Section_9de0de005de2469ea7c94865299c8d91).

1. 00000000 04 00 00 00 00
2. 04 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_OPEN\_REPLY (4)
3. 00 00 00 00 -> MSG\_SNDIN\_OPEN\_REPLY::Result = S\_OK (0)

## Annotated Data Transfer Sequence

The following is an annotated dump of a data transfer sequence as described in section [1.3.2](#Section_9c8c2dc562c844109f12f53705b20e5a).

### Incoming Data PDU

The following is an annotated dump of an [Incoming Data PDU](#Section_839232173dfa427dbda2e146ae3cfc92).

1. 00000000 05
2. 05 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_DATA\_INCOMING (5)

### Data PDU

The following is an annotated dump of a [Data PDU](#Section_7e84ce6652a8422cbbb36d383dac516f).

1. 00000000 06 d6 38 99 59 05 ac 56 93 24 05 d4 df 13 5a c6
2. 00000010 7c b6 6e 7b 0b bb 2d d9 e9 5c 04 2e fc 64 d9 89
3. 00000020 1d 59 89 71 1d 2d b1 b2 b1 e5 1c b9 4c 78 c1 f7
4. 00000030 6d 13 3b b6 77 26 ed b6 bd eb b6 d9 60 df 4e 2d
5. 00000040 59 a7 97 09 93 11 d3 76 9a 4d b9 60 5b c6 ca 3b
6. 00000050 2d 2b 75 d9 a3 c4 18 99 36 49 d3 44 2e f6 c9 34
7. 00000060 11 48 56 95 91 24 28 f5 c0 8f 76 24 17 6e 51 c2
8. 00000070 ac 00 00 10 04 74 16 25 49 92 24 a1 61 da 44 44
9. 00000080 49 92 24 d3 07 d7 cd a2 bd 95 e4 48 ae 06 07 4b
10. 00000090 7d fb 38 52 e2 a8 24 2d 09 12 40 49 62 c3 6c 00
11. 000000a0 12 24 49 f2 6c 8e ee 40 5e ce 86 20 49 22 45 cf
12. 000000b0 8f 49 92 24 c9 37 76 35 61 78 dc b6 8d 2b 9c 38
13. 000000c0 0c 49 92 24 d3 08 d7 d5 14 65 4c 10 80 04 00 f2
14. 000000d0 db 7b db c6 53 e1 bc 23 99 64 73 92 24 49 55 39
15. 000000e0 22 c9 7d 7d bb 80 33 19 4c 52 ee fe fb b4 db 96
16. 000000f0 3f c6 8a 96 64 db 96 e5 88 51 52 1b b7 8d 78 5c
17. 00000100 fe ff ff ff ff 51 e8 16 11 63 5f ac 22 cd 92 65
18. 00000110 51 0f 98 1f 49 92 24 c9 cc 96 24 49 92 0c 3c 6d
19. 00000120 f6 dc 48 92 94 77 8d 91 9d 51 d0 6f c9 91 24 89
20. 00000130 95 67 b1 73 46 db 4c 99 a9 c0 25 77 2e 61 92 3d
21. 00000140 42 6d df fe 5d df 93 f8 d6 95 53 a3 d1 ff 6f db
22. 00000150 b6 bd 66 b2 ed d9 b6 6d 5b 77 7c db b6 6d 60 67
23. 00000160 cf 14 27 49 92 24 6b a1 b0 19 31 57 67 99 eb 25
24. 00000170 dd ea e2 9a 71 e1 25 49 92 be 9c 63 92 65 07 c5
25. 00000180 c9 c2 c7 92 23 b4 4d
26. 06 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_DATA (6)
27. d6... 23 b4 4d -> Data

## Annotated Format Change Sequence

The following is an annotated format change sequence as described in section [1.3.3](#Section_6ee6dc011aec40478167dc07de703d1d).

### Server Format Change PDU

The following is an annotated [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866).

1. 00000000 07 0b 00 00 00
2. 07 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_FORMATCHANGE (7)
3. 0b 00 00 00 -> MSG\_SNDIN\_FORMATCHANGE::NewFormat = 0xb = 11

### Client Format Change PDU

The following is an annotated [Format Change PDU](#Section_8d51c8bc5a7941c897c4b0a9daaba866).

1. 00000000 07 0b 00 00 00
2. 07 -> MSG\_SNDIN\_PDU\_HEADER::Version = MSG\_SNDIN\_FORMATCHANGE (7)
3. 0b 00 00 00 -> MSG\_SNDIN\_FORMATCHANGE::NewFormat = 0xb = 11

# Security

## Security Considerations for Implementers

All dynamic virtual traffic is secured by the underlying core RDP. An overview of the implemented security-related mechanisms is as described in [[MS-RDPBCGR]](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c) section 5.

## Index of Security Parameters

None.

# Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

* Windows 7 operating system
* Windows Server 2008 R2 operating system
* Windows 8 operating system
* Windows Server 2012 operating system
* Windows 8.1 operating system
* Windows Server 2012 R2 operating system
* Windows 10 operating system
* Windows Server 2016 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

[<1> Section 2.2.2.2](#Appendix_A_Target_1): The following codecs are supported by default on Windows, which includes support for audio input redirection. Unless otherwise specified, information about these codecs can be found in [[RFC2361]](https://go.microsoft.com/fwlink/?LinkId=90337).

| Codec name | Format tag |
| --- | --- |
| Microsoft PCM | WAVE\_FORMAT\_PCM  0x0001 |
| Microsoft Adaptive PCM | WAVE\_FORMAT\_ADPCM  0x0002 |
| Microsoft ALAW | WAVE\_FORMAT\_ALAW  0x0006 |
| Microsoft G.723 | WAVE\_FORMAT\_MSG723  0x0042 |
| Microsoft GSM | WAVE\_FORMAT\_GSM610  0x0031 |
| Microsoft MULAW | WAVE\_FORMAT\_MULAW  0x0007 |
| Microsoft AAC  Not supported on Windows 7 and Windows Server 2008 R2 | WAVE\_FORMAT\_AAC  0xa106 |

Microsoft Implementations deviate from the GSM standard [[ETSI-GSM]](https://go.microsoft.com/fwlink/?LinkId=114230) while bitpacking coefficients, as described by the following example:

1. ITU-T Standard Implementation of GSM 610
2. if (((\*c >> 4) & 0x0F) != GSM\_MAGIC) return -1;
3. LARc[1] = (\*c++ & 0xF) << 2; /\* 1 \*/
4. LARc[1] |= (\*c >> 6) & 0x3;
5. LARc[2] = \*c++ & 0x3F;
6. LARc[3] = (\*c >> 3) & 0x1F;
7. LARc[4] = (\*c++ & 0x7) << 2;
8. LARc[4] |= (\*c >> 6) & 0x3;
9. LARc[5] = (\*c >> 2) & 0xF;
10. LARc[6] = (\*c++ & 0x3) << 2;
11. LARc[6] |= (\*c >> 6) & 0x3;
12. LARc[7] = (\*c >> 3) & 0x7;
13. LARc[8] = \*c++ & 0x7;
15. Microsoft Implementaion of GSM 610
16. LAR[1] = (ab[0] & 0x3F);
17. LAR[2] = ((ab[0] & 0xC0) >> 6) | ((ab[1] & 0x0F) << 2);
18. LAR[3] = ((ab[1] & 0xF0) >> 4) | ((ab[2] & 0x01) << 4);
19. LAR[4] = ((ab[2] & 0x3E) >> 1);
20. LAR[5] = ((ab[2] & 0xC0) >> 6) | ((ab[3] & 0x03) << 2);
21. LAR[6] = ((ab[3] & 0x3C) >> 2);
22. LAR[7] = ((ab[3] & 0xC0) >> 6) | ((ab[4] & 0x01) << 2);
23. LAR[8] = ((ab[4] & 0x0E) >> 1);

The ITU implementation is Most Significant Bit (MSB) to Least Significant Bit (LSB). The Microsoft Implementation is LSB to MSB. The following coefficients are represented as MSB to LSB. The first line is the bit position in the actual byte, the second line is the coefficient index, and the third line is the bit position of the coefficients.

1. ITU:
3. BYTE 0 BYTE 1 BYTE 2 BYTE 3
4. 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0
5. 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 ...
6. 5 4 3 2 1 0 5 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 ...
8. MS:
10. BYTE 3 BYTE 2 BYTE 1 BYTE 0
11. 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0
12. ... 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1
13. ... 4 3 2 1 0 4 3 2 1 0 5 4 3 2 1 0 5 4 3 2 1 0

[<2> Section 3.3.2](#Appendix_A_Target_2): Windows servers will wait 5 seconds after sending a Version PDU. If the Version PDU from the client is not received within this time, the server terminates the protocol.

[<3> Section 3.3.2](#Appendix_A_Target_3): Windows servers will wait 5 seconds after sending a Sound Formats PDU. If the Sound Formats PDU from the client is not received within this time, the server terminates the protocol.

[<4> Section 3.3.2](#Appendix_A_Target_4): Windows servers will wait 5 seconds after sending an Open PDU. If the Open Reply PDU from the client is not received within this time, the server terminates the protocol.

# Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

# Index

A

Abstract data model

client ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.2.1](#section_bf040af8795d4742949fd6886b3f91c5) 21)

server ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.3.1](#section_afe798c3d53842a2881675859b93649b) 23)

Annotated data transfer sequence examples

[Data PDU](#section_39a3b8977a2147a88ad407aaef75b60b) 37

[Incoming Data PDU](#section_bb213d1b5e4d4b7b83e050f96fd93b27) 36

[overview](#section_9240df7058fd41118db295390207c612) 36

Annotated format change sequence examples

[client Format Change PDU](#section_cb6b43fc239b426cb1a13bac8e984705) 37

[overview](#section_e6c33a50bcae46cf96914eb51d28c8ae) 37

[server Format Change PDU](#section_848680ba8f6d41cfbe00e0e7f824ffb8) 37

Annotated initialization sequence examples

[client Sound Formats PDU](#section_d4f07a7181fc42c4a22c953d4a127cc3) 31

[client Version PDU](#section_d32f2f16851b4caaaf387891d7074175) 27

[Format Change PDU](#section_c2ee9379d6e944029895299c0c0eee75) 36

[Incoming Data PDU](#section_6fbf5342f1b948549665816e41b0350f) 31

[Open PDU](#section_ce8a340f305e4238ae946ed508d93b76) 36

[Open Reply PDU](#section_c3236eec1e024a00aa6f06a4b39ebe6f) 36

[overview](#section_4c9ed751c73b4473b398897ab411d714) 27

[server Sound Formats PDU](#section_5060513ba68e4dbebb64232b702fa333) 27

[server Version PDU](#section_6f2dd2877c4047c6b2c8f010a766d2a3) 27

[Applicability](#section_83c5981406fb4c35a26e190387c8ac06) 11

[Audio recording](#section_b0422aa763d842bd88e662bd96830bf3) 19

C

[Capability negotiation](#section_495396eba7c9464a80814529df5feb25) 11

[Change tracking](#section_1bcbf32274f9450da9a8470285092cb4) 41

Client ([section 3.1](#section_2ffd5eca15454158896a48de5aceae18) 19, [section 3.2](#section_0b85f4652e3e44da8dee6de1aee4ca53) 21)

abstract data model ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.2.1](#section_bf040af8795d4742949fd6886b3f91c5) 21)

higher-layer triggered events ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.2.4](#section_644087cc6a3c4039aa45d169c04af381) 21)

initialization ([section 3.1.3](#section_547a89a5425c4fcca548dccba6d89dd6) 19, [section 3.2.3](#section_84228b714c104edf8b28fa08dd9946d9) 21)

local events ([section 3.1.7](#section_f6de49abde7848a6bf309b569117823a) 21, [section 3.2.7](#section_a83549ff4d5c477b83630ec154f453b2) 23)

[message processing](#section_24ff9db29efc49c095c2aa2db2a9baf7) 21

[data transfer sequence](#section_3689bf5888cc4e2e91b84cdb60c84d78) 23

[format change sequence](#section_799885286f8c48b4ab5b7434e7c11a8a) 23

[initialization sequence](#section_84d6c7b318864157920fc9a5bf68ff64) 21

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

[other local events](#section_a83549ff4d5c477b83630ec154f453b2) 23

[sequencing rules](#section_24ff9db29efc49c095c2aa2db2a9baf7) 21

[data transfer sequence](#section_3689bf5888cc4e2e91b84cdb60c84d78) 23

[format change sequence](#section_799885286f8c48b4ab5b7434e7c11a8a) 23

[initialization sequence](#section_84d6c7b318864157920fc9a5bf68ff64) 21

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

timer events ([section 3.1.6](#section_9af8692d07004484b1b7f6b2a060a28b) 21, [section 3.2.6](#section_e4b25061ff6846a79c21aa2795c93747) 23)

timers ([section 3.1.2](#section_ff4ad6a7c44845ef8cfa53062e9cd313) 19, [section 3.2.2](#section_17edfb8435fc4d5fbb4ed75daf470bda) 21)

D

Data model - abstract

client ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.2.1](#section_bf040af8795d4742949fd6886b3f91c5) 21)

server ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.3.1](#section_afe798c3d53842a2881675859b93649b) 23)

Data PDU

[processing](#section_b1c47ef62d0e4f148eb5a55d18853565) 25

[sending](#section_9d63223f2ed64b47abdf6e53d2edf643) 23

[Data Transfer Messages message](#section_7b671d5ee1534bd79bbb7a3cb27cfb1b) 17

Data transfer sequence

annotated examples

[Data PDU](#section_39a3b8977a2147a88ad407aaef75b60b) 37

[Incoming Data PDU](#section_bb213d1b5e4d4b7b83e050f96fd93b27) 36

[overview](#section_9240df7058fd41118db295390207c612) 36

[client](#section_3689bf5888cc4e2e91b84cdb60c84d78) 23

[sending Data PDU](#section_9d63223f2ed64b47abdf6e53d2edf643) 23

[sending Incoming Data PDU](#section_9010a07c0cfe45d8bcb6062968991d55) 23

[server](#section_e870ec238f0b4c27ab2db4ab5cdb1f24) 25

[processing Data PDU](#section_b1c47ef62d0e4f148eb5a55d18853565) 25

[processing Incoming Data PDU](#section_7e40602348f34f9dbe6c1319b3c656f5) 25

E

[Examples](#section_17589946b0e546518febe79a18df8914) 27

annotated data transfer sequence

[Data PDU](#section_39a3b8977a2147a88ad407aaef75b60b) 37

[Incoming Data PDU](#section_bb213d1b5e4d4b7b83e050f96fd93b27) 36

[overview](#section_9240df7058fd41118db295390207c612) 36

annotated format change sequence

[client Format Change PDU](#section_cb6b43fc239b426cb1a13bac8e984705) 37

[overview](#section_e6c33a50bcae46cf96914eb51d28c8ae) 37

[server Format Change PDU](#section_848680ba8f6d41cfbe00e0e7f824ffb8) 37

annotated initialization sequence

[client Sound Formats PDU](#section_d4f07a7181fc42c4a22c953d4a127cc3) 31

[client Version PDU](#section_d32f2f16851b4caaaf387891d7074175) 27

[Format Change PDU](#section_c2ee9379d6e944029895299c0c0eee75) 36

[Incoming Data PDU](#section_6fbf5342f1b948549665816e41b0350f) 31

[Open PDU](#section_ce8a340f305e4238ae946ed508d93b76) 36

[Open Reply PDU](#section_c3236eec1e024a00aa6f06a4b39ebe6f) 36

[overview](#section_4c9ed751c73b4473b398897ab411d714) 27

[server Sound Formats PDU](#section_5060513ba68e4dbebb64232b702fa333) 27

[server Version PDU](#section_6f2dd2877c4047c6b2c8f010a766d2a3) 27

F

[Fields - vendor-extensible](#section_0e5e21e7ba194c1faecd56bb9fdcb5f0) 11

[Format Change Messages message](#section_9645b7027ac247e18c4f15036d43c4ff) 18

Format Change PDU

processing

[client](#section_bd3b5bb59b8141cbbff0ce4eaabd2dad) 23

server ([section 3.3.5.1.7](#section_f745fff733ed475aa9b9f832cf902566) 25, [section 3.3.5.3.2](#section_566ca8eeff8f4951a17c6a68acfd8213) 25)

sending

client ([section 3.2.5.1.7](#section_4b3fa3699aa84776a96beff10b8c4fbc) 22, [section 3.2.5.3.2](#section_74e22a5c535c481ba99bca30cbd40fd2) 23)

[server](#section_731132c698af442386d00e70be029bf0) 25

Format change sequence

annotated examples

[client Format Change PDU](#section_cb6b43fc239b426cb1a13bac8e984705) 37

[overview](#section_e6c33a50bcae46cf96914eb51d28c8ae) 37

[server Format Change PDU](#section_848680ba8f6d41cfbe00e0e7f824ffb8) 37

[client](#section_799885286f8c48b4ab5b7434e7c11a8a) 23

[processing Format Change PDU](#section_bd3b5bb59b8141cbbff0ce4eaabd2dad) 23

[sending Format Change PDU](#section_74e22a5c535c481ba99bca30cbd40fd2) 23

[server](#section_e33a314a2e9449a7a97b6d4784acf908) 25

[processing Format Change PDU](#section_566ca8eeff8f4951a17c6a68acfd8213) 25

[sending Format Change PDU](#section_731132c698af442386d00e70be029bf0) 25

G

[Glossary](#section_43c01ffc3faf437ba162d21d1392cc48) 7

H

Higher-layer triggered events

client ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.2.4](#section_644087cc6a3c4039aa45d169c04af381) 21)

server ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.3.4](#section_dda2f811da1b4172b72f430345452e14) 24)

I

[Implementer - security considerations](#section_034f3438c6104e48a8207cfe872bc4c1) 38

Incoming Data PDU

processing ([section 3.3.5.1.4](#section_044e7cdc619a43e7a1f262d37289d332) 24, [section 3.3.5.2.1](#section_7e40602348f34f9dbe6c1319b3c656f5) 25)

sending ([section 3.2.5.1.4](#section_8ebad22ffded4981b40a6660115981ac) 22, [section 3.2.5.2.1](#section_9010a07c0cfe45d8bcb6062968991d55) 23)

[Index of security parameters](#section_70e85667414f4b819b2e0b7e12ab928e) 38

[Informative references](#section_8ebecde4805c48039c3b47f8b8514605) 8

Initialization

client ([section 3.1.3](#section_547a89a5425c4fcca548dccba6d89dd6) 19, [section 3.2.3](#section_84228b714c104edf8b28fa08dd9946d9) 21)

server ([section 3.1.3](#section_547a89a5425c4fcca548dccba6d89dd6) 19, [section 3.3.3](#section_f8c01b2d38634554a9b9eaab42c2dd02) 24)

[Initialization Messages message](#section_43f5785f56fa477ba98262e43091c6ec) 12

Initialization sequence

annotated examples

[client Sound Formats PDU](#section_d4f07a7181fc42c4a22c953d4a127cc3) 31

[client Version PDU](#section_d32f2f16851b4caaaf387891d7074175) 27

[Format Change PDU](#section_c2ee9379d6e944029895299c0c0eee75) 36

[Incoming Data PDU](#section_6fbf5342f1b948549665816e41b0350f) 31

[Open PDU](#section_ce8a340f305e4238ae946ed508d93b76) 36

[Open Reply PDU](#section_c3236eec1e024a00aa6f06a4b39ebe6f) 36

[overview](#section_4c9ed751c73b4473b398897ab411d714) 27

[server Sound Formats PDU](#section_5060513ba68e4dbebb64232b702fa333) 27

[server Version PDU](#section_6f2dd2877c4047c6b2c8f010a766d2a3) 27

[client](#section_84d6c7b318864157920fc9a5bf68ff64) 21

[processing Open PDU](#section_ae0639e696f64077a436d2ec14918aab) 22

[processing Sound Formats PDU](#section_02823a3fd7bf40d2ac5e82f0d605c120) 21

[processing Version PDU](#section_6b0691f0ffdd48df85924db86db139e8) 21

[sending Format Change PDU](#section_4b3fa3699aa84776a96beff10b8c4fbc) 22

[sending Incoming Data PDU](#section_8ebad22ffded4981b40a6660115981ac) 22

[sending Open Reply PDU](#section_2baf760db6ce49c6b1d2b76a106e7072) 22

[sending Sound Formats PDU](#section_d3db92728e0a44908001f5519641373d) 22

[sending Version PDU](#section_ff28f8737e1b40399680115abdc765be) 21

[server](#section_32478681c8b547e8bff5ed2508bed7ae) 24

[processing Format Change PDU](#section_f745fff733ed475aa9b9f832cf902566) 25

[processing Incoming Data PDU](#section_044e7cdc619a43e7a1f262d37289d332) 24

[processing Open Reply PDU](#section_5b8e7a065c9d48c08c89006019007258) 25

[processing Sound Formats PDU](#section_2879cff8a11a470ba5d110610f616b9b) 24

[processing Version PDU](#section_23ea3d3c80bf4b3c9212ded42aa3d2b4) 24

[sending Open PDU](#section_5b3d11b617404986830236a967cc22dc) 25

[sending Sound Formats PDU](#section_e82d1fcd58c7453eb72e8967cd240426) 24

[sending Version PDU](#section_8e1b17e45e90445bbd412f657fd83ff9) 24

[Introduction](#section_80ea2da676164a878b9e5a3c173c256d) 7

L

Local events

client ([section 3.1.7](#section_f6de49abde7848a6bf309b569117823a) 21, [section 3.2.7](#section_a83549ff4d5c477b83630ec154f453b2) 23)

server ([section 3.1.7](#section_f6de49abde7848a6bf309b569117823a) 21, [section 3.3.8](#section_7d23d913e08647eda605a4eb96541f9f) 26)

M

Message processing

[client](#section_24ff9db29efc49c095c2aa2db2a9baf7) 21

[data transfer sequence](#section_3689bf5888cc4e2e91b84cdb60c84d78) 23

[format change sequence](#section_799885286f8c48b4ab5b7434e7c11a8a) 23

[initialization sequence](#section_84d6c7b318864157920fc9a5bf68ff64) 21

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

[server](#section_9315b145238b4349bf58f4cf693aacde) 24

[data transfer sequence](#section_e870ec238f0b4c27ab2db4ab5cdb1f24) 25

[format change sequence](#section_e33a314a2e9449a7a97b6d4784acf908) 25

[initialization sequence](#section_32478681c8b547e8bff5ed2508bed7ae) 24

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

[Messages](#section_d7d959ef89304a4a98d74ae00d059f90) 12

[Data Transfer Messages](#section_7b671d5ee1534bd79bbb7a3cb27cfb1b) 17

[Format Change Messages](#section_9645b7027ac247e18c4f15036d43c4ff) 18

[Initialization Messages](#section_43f5785f56fa477ba98262e43091c6ec) 12

[SNDIN\_PDU Header](#section_51b2d5c1294347b1b5948a04a53df7ec) 12

syntax

[data transfer messages](#section_7b671d5ee1534bd79bbb7a3cb27cfb1b) 17

[format change messages](#section_9645b7027ac247e18c4f15036d43c4ff) 18

[initialization messages](#section_43f5785f56fa477ba98262e43091c6ec) 12

[overview](#section_2451e5b379ec4f6a88cf5419737cd1e4) 12

[transport](#section_ff2fa50258574e7996985f17eb1e8a1e) 12

[MSG\_SNDIN\_DATA packet](#section_7e84ce6652a8422cbbb36d383dac516f) 17

[MSG\_SNDIN\_DATA\_INCOMING packet](#section_839232173dfa427dbda2e146ae3cfc92) 17

[MSG\_SNDIN\_FORMATS packet](#section_c4e53155f57743c6ac1dfad49540d02d) 13

[MSG\_SNDIN\_OPEN packet](#section_97db7244249b4a79ad507e45e2f760a0) 14

[MSG\_SNDIN\_OPEN\_REPLY packet](#section_9de0de005de2469ea7c94865299c8d91) 16

MSG\_SNDIN\_VERSION packet ([section 2.2.2.1](#section_b25cb6b21ab0405698a371b85d9b66b4) 13, [section 2.2.4.1](#section_8d51c8bc5a7941c897c4b0a9daaba866) 18)

N

[Normative references](#section_05001dc878fa4dff94bd241c67ad7af3) 8

O

Open PDU

[processing](#section_ae0639e696f64077a436d2ec14918aab) 22

[sending](#section_5b3d11b617404986830236a967cc22dc) 25

Open Reply PDU

[processing](#section_5b8e7a065c9d48c08c89006019007258) 25

[sending](#section_2baf760db6ce49c6b1d2b76a106e7072) 22

Other local events

[client](#section_a83549ff4d5c477b83630ec154f453b2) 23

[server](#section_7d23d913e08647eda605a4eb96541f9f) 26

[Overview (synopsis)](#section_f7ddb98c85c44a578feebaa4e8d56cf2) 8

[data transfer sequence](#section_9c8c2dc562c844109f12f53705b20e5a) 9

[format change sequence](#section_6ee6dc011aec40478167dc07de703d1d) 10

[initialization sequence](#section_c287fe7796ac42d6a62cae20d23ca2a5) 8

[overview](#section_f7ddb98c85c44a578feebaa4e8d56cf2) 8

P

[Parameters - security index](#section_70e85667414f4b819b2e0b7e12ab928e) 38

PDU (protocol data unit)

[processing Data](#section_b1c47ef62d0e4f148eb5a55d18853565) 25

processing Format Change

[client](#section_bd3b5bb59b8141cbbff0ce4eaabd2dad) 23

server ([section 3.3.5.1.7](#section_f745fff733ed475aa9b9f832cf902566) 25, [section 3.3.5.3.2](#section_566ca8eeff8f4951a17c6a68acfd8213) 25)

processing Incoming Data ([section 3.3.5.1.4](#section_044e7cdc619a43e7a1f262d37289d332) 24, [section 3.3.5.2.1](#section_7e40602348f34f9dbe6c1319b3c656f5) 25)

[processing Open](#section_ae0639e696f64077a436d2ec14918aab) 22

[processing Open Reply](#section_5b8e7a065c9d48c08c89006019007258) 25

processing Sound Formats

[client](#section_02823a3fd7bf40d2ac5e82f0d605c120) 21

[server](#section_2879cff8a11a470ba5d110610f616b9b) 24

processing Version

[client](#section_6b0691f0ffdd48df85924db86db139e8) 21

[server](#section_23ea3d3c80bf4b3c9212ded42aa3d2b4) 24

[sending Data](#section_9d63223f2ed64b47abdf6e53d2edf643) 23

sending Format Change

client ([section 3.2.5.1.7](#section_4b3fa3699aa84776a96beff10b8c4fbc) 22, [section 3.2.5.3.2](#section_74e22a5c535c481ba99bca30cbd40fd2) 23)

[server](#section_731132c698af442386d00e70be029bf0) 25

sending Incoming Data ([section 3.2.5.1.4](#section_8ebad22ffded4981b40a6660115981ac) 22, [section 3.2.5.2.1](#section_9010a07c0cfe45d8bcb6062968991d55) 23)

[sending Open](#section_5b3d11b617404986830236a967cc22dc) 25

[sending Open Reply](#section_2baf760db6ce49c6b1d2b76a106e7072) 22

sending Sound Formats

[client](#section_d3db92728e0a44908001f5519641373d) 22

[server](#section_e82d1fcd58c7453eb72e8967cd240426) 24

sending Version

[client](#section_ff28f8737e1b40399680115abdc765be) 21

[server](#section_8e1b17e45e90445bbd412f657fd83ff9) 24

[Preconditions](#section_dc7d35ae008d4329a88d1b10d85dd152) 10

[Prerequisites](#section_dc7d35ae008d4329a88d1b10d85dd152) 10

[Product behavior](#section_8ea102e2d7ee49cdbe17e49313533949) 39

Protocol data unit (PDU)

[processing Data](#section_b1c47ef62d0e4f148eb5a55d18853565) 25

processing Format Change

[client](#section_bd3b5bb59b8141cbbff0ce4eaabd2dad) 23

server ([section 3.3.5.1.7](#section_f745fff733ed475aa9b9f832cf902566) 25, [section 3.3.5.3.2](#section_566ca8eeff8f4951a17c6a68acfd8213) 25)

processing Incoming Data ([section 3.3.5.1.4](#section_044e7cdc619a43e7a1f262d37289d332) 24, [section 3.3.5.2.1](#section_7e40602348f34f9dbe6c1319b3c656f5) 25)

[processing Open](#section_ae0639e696f64077a436d2ec14918aab) 22

[processing Open Reply](#section_5b8e7a065c9d48c08c89006019007258) 25

processing Sound Formats

[client](#section_02823a3fd7bf40d2ac5e82f0d605c120) 21

[server](#section_2879cff8a11a470ba5d110610f616b9b) 24

processing Version

[client](#section_6b0691f0ffdd48df85924db86db139e8) 21

[server](#section_23ea3d3c80bf4b3c9212ded42aa3d2b4) 24

[sending Data](#section_9d63223f2ed64b47abdf6e53d2edf643) 23

sending Format Change

client ([section 3.2.5.1.7](#section_4b3fa3699aa84776a96beff10b8c4fbc) 22, [section 3.2.5.3.2](#section_74e22a5c535c481ba99bca30cbd40fd2) 23)

[server](#section_731132c698af442386d00e70be029bf0) 25

sending Incoming Data ([section 3.2.5.1.4](#section_8ebad22ffded4981b40a6660115981ac) 22, [section 3.2.5.2.1](#section_9010a07c0cfe45d8bcb6062968991d55) 23)

[sending Open](#section_5b3d11b617404986830236a967cc22dc) 25

[sending Open Reply](#section_2baf760db6ce49c6b1d2b76a106e7072) 22

sending Sound Formats

[client](#section_d3db92728e0a44908001f5519641373d) 22

[server](#section_e82d1fcd58c7453eb72e8967cd240426) 24

sending Version

[client](#section_ff28f8737e1b40399680115abdc765be) 21

[server](#section_8e1b17e45e90445bbd412f657fd83ff9) 24

R

[Recording audio](#section_b0422aa763d842bd88e662bd96830bf3) 19

[References](#section_f4b4f85c69fc4089a89250876dbfe624) 7

[informative](#section_8ebecde4805c48039c3b47f8b8514605) 8

[normative](#section_05001dc878fa4dff94bd241c67ad7af3) 8

[Relationship to other protocols](#section_842af31c34584b038994db536e18e8fc) 10

S

[Security](#section_e2a729feb54b4dda961e2411bf876731) 38

[implementer considerations](#section_034f3438c6104e48a8207cfe872bc4c1) 38

[parameter index](#section_70e85667414f4b819b2e0b7e12ab928e) 38

Sequencing rules

[client](#section_24ff9db29efc49c095c2aa2db2a9baf7) 21

[data transfer sequence](#section_3689bf5888cc4e2e91b84cdb60c84d78) 23

[format change sequence](#section_799885286f8c48b4ab5b7434e7c11a8a) 23

[initialization sequence](#section_84d6c7b318864157920fc9a5bf68ff64) 21

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

[server](#section_9315b145238b4349bf58f4cf693aacde) 24

[data transfer sequence](#section_e870ec238f0b4c27ab2db4ab5cdb1f24) 25

[format change sequence](#section_e33a314a2e9449a7a97b6d4784acf908) 25

[initialization sequence](#section_32478681c8b547e8bff5ed2508bed7ae) 24

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

Server ([section 3.1](#section_2ffd5eca15454158896a48de5aceae18) 19, [section 3.3](#section_cb120a30bd5a41b69877398855870ec1) 23)

abstract data model ([section 3.1.1](#section_d3856db25ae44909baba23f8df26fc6c) 19, [section 3.3.1](#section_afe798c3d53842a2881675859b93649b) 23)

higher-layer triggered events ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.3.4](#section_dda2f811da1b4172b72f430345452e14) 24)

initialization ([section 3.1.3](#section_547a89a5425c4fcca548dccba6d89dd6) 19, [section 3.3.3](#section_f8c01b2d38634554a9b9eaab42c2dd02) 24)

local events ([section 3.1.7](#section_f6de49abde7848a6bf309b569117823a) 21, [section 3.3.8](#section_7d23d913e08647eda605a4eb96541f9f) 26)

[message processing](#section_9315b145238b4349bf58f4cf693aacde) 24

[data transfer sequence](#section_e870ec238f0b4c27ab2db4ab5cdb1f24) 25

[format change sequence](#section_e33a314a2e9449a7a97b6d4784acf908) 25

[initialization sequence](#section_32478681c8b547e8bff5ed2508bed7ae) 24

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

[other local events](#section_7d23d913e08647eda605a4eb96541f9f) 26

[sequencing rules](#section_9315b145238b4349bf58f4cf693aacde) 24

[data transfer sequence](#section_e870ec238f0b4c27ab2db4ab5cdb1f24) 25

[format change sequence](#section_e33a314a2e9449a7a97b6d4784acf908) 25

[initialization sequence](#section_32478681c8b547e8bff5ed2508bed7ae) 24

[overview](#section_6a9d4415b7f444b1a8031ecc34195761) 20

timer events ([section 3.1.6](#section_9af8692d07004484b1b7f6b2a060a28b) 21, [section 3.3.7](#section_b28d0764644d4cafba37dae4745865fe) 26)

timers ([section 3.1.2](#section_ff4ad6a7c44845ef8cfa53062e9cd313) 19, [section 3.3.2](#section_0efa5fbf628044c0b7a4ae20f6216eba) 24)

[SNDIN\_PDU Header message](#section_51b2d5c1294347b1b5948a04a53df7ec) 12

[SNDIN\_PDU packet](#section_51b2d5c1294347b1b5948a04a53df7ec) 12

Sound Formats PDU

processing

[client](#section_02823a3fd7bf40d2ac5e82f0d605c120) 21

[server](#section_2879cff8a11a470ba5d110610f616b9b) 24

sending

[client](#section_d3db92728e0a44908001f5519641373d) 22

[server](#section_e82d1fcd58c7453eb72e8967cd240426) 24

[Standards assignments](#section_9a861185fcdf43a18f69f7797babbfbd) 11

Syntax

[data transfer messages](#section_7b671d5ee1534bd79bbb7a3cb27cfb1b) 17

[format change messages](#section_9645b7027ac247e18c4f15036d43c4ff) 18

[initialization messages](#section_43f5785f56fa477ba98262e43091c6ec) 12

[overview](#section_2451e5b379ec4f6a88cf5419737cd1e4) 12

T

Timer events

client ([section 3.1.6](#section_9af8692d07004484b1b7f6b2a060a28b) 21, [section 3.2.6](#section_e4b25061ff6846a79c21aa2795c93747) 23)

server ([section 3.1.6](#section_9af8692d07004484b1b7f6b2a060a28b) 21, [section 3.3.7](#section_b28d0764644d4cafba37dae4745865fe) 26)

Timers

client ([section 3.1.2](#section_ff4ad6a7c44845ef8cfa53062e9cd313) 19, [section 3.2.2](#section_17edfb8435fc4d5fbb4ed75daf470bda) 21)

server ([section 3.1.2](#section_ff4ad6a7c44845ef8cfa53062e9cd313) 19, [section 3.3.2](#section_0efa5fbf628044c0b7a4ae20f6216eba) 24)

[Tracking changes](#section_1bcbf32274f9450da9a8470285092cb4) 41

[Transport](#section_ff2fa50258574e7996985f17eb1e8a1e) 12

Triggered events - higher-layer

client ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.2.4](#section_644087cc6a3c4039aa45d169c04af381) 21)

server ([section 3.1.4](#section_63a3a1487f5b403aa2e350ddd6755fae) 19, [section 3.3.4](#section_dda2f811da1b4172b72f430345452e14) 24)

V

[Vendor-extensible fields](#section_0e5e21e7ba194c1faecd56bb9fdcb5f0) 11

Version PDU

processing

[client](#section_6b0691f0ffdd48df85924db86db139e8) 21

[server](#section_23ea3d3c80bf4b3c9212ded42aa3d2b4) 24

sending

[client](#section_ff28f8737e1b40399680115abdc765be) 21

[server](#section_8e1b17e45e90445bbd412f657fd83ff9) 24

[Versioning](#section_495396eba7c9464a80814529df5feb25) 11

W

[WAVEFORMAT\_EXTENSIBLE packet](#section_84835e471f12470faacb80775c854b04) 15