[MS-RDPEDYC]:
Remote Desktop Protocol:
Dynamic Channel Virtual Channel Extension

Revision Summary

Date	Revision History	Revision Class	Comments
02/22/2007	0.01		MCPP Milestone 3 Initial Availability
06/01/2007	1.0	Major	Updated and revised the technical content.
07/03/2007	1.0.1	Minor	Editorial fixes only.
07/20/2007	1.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
08/10/2007	1.0.3	Editorial	Revised and edited the technical content.
09/28/2007	1.0.4	Editorial	Revised and edited the technical content.
10/23/2007	1.0.5	Minor	Revised a figure; revised size of cmd field in several packets.
11/30/2007	1.1	Minor	Updated the technical content.
01/25/2008	1.1.1	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document describes the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension available in Windows Vista. This protocol is an extension and refinement of the **virtual channel** protocol, as specified in [MS-RDPBCGR]. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension supports features such as classes of priority (that may be used to implement bandwidth allocation) and individually connected endpoints using **dynamic virtual channel (DVC) listeners**.

1.1 Glossary

The following terms are specific to this document:

Data Message (or Message): Data exchanged between an application running on a **Terminal Services (TS)** server and a **dynamic virtual channel (DVC) listeners** running on a **TS** client. The maximum length of a **data message** is 2^32 bytes.

Dynamic Virtual Channel (DVC) Manager: An application that runs on the **Terminal Services (TS)** servers and clients. They manage the initialization, creation, and closing of dynamic **virtual channels**. They are responsible for maintaining established channels and for transferring **messages** between the applications on the **TS** servers and the **dynamic virtual channel (DVC) listeners** that run on the **TS** clients.

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

Static Virtual Channel: The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is designed to operate over static **virtual channels**, as specified in [MS-RDPBCGR], using the acronym DRDYNVC. The Remote Desktop Protocol (RDP) layer manages the creation, setup, and data transmission over the **virtual channel**.

Terminal Services (TS): The capability to host multiple, simultaneous client sessions on Windows servers. Remote users establish a session on a machine, log in, and run applications on a server. The server transmits the graphical user interface (GUI) of the program to the client. The client then returns keyboard and mouse clicks to be processed by the server.

Virtual Channel: A communication channel available in a **Terminal Services (TS)** server session between applications running at the server and applications running on the **TS** client.

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

1.2 References

1.2.1 Normative References

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

[MS-RDPBCGR] Microsoft Corporation, "Remote Desktop Protocol: Basic Connectivity and Graphics Remoting Specification", June 2007.

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

1.2.2 Informative References

[MSDN-TSVC] Microsoft Corporation, "Using Terminal Services Virtual Channels", http://msdn2.microsoft.com/en-us/library/aa383546.aspx

1.3 Protocol Overview (Synopsis)

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension implements a generic connection-oriented communication channel on top of the virtual channel protocol. A dynamic virtual channel is established over an existing **static virtual channel**. A static virtual channel session is a typical client/server relationship.

A dynamic virtual channel consists of two endpoints logically connected over a network. One endpoint is an application running on a **Terminal Services (TS)** server, and the other endpoint is an application running on a TS client.

Dynamic virtual channels are created and maintained by **DVC managers**. There is a DVC manager running on both the TS server and the TS client. The DVC server manager is responsible for initializing the dynamic virtual channel environment and for creating individual dynamic virtual channels. The DVC client manager is responsible for creating and maintaining connections to client-side DVC manager applications.

After the DVC managers are initialized, the DVC server manager can create individual dynamic virtual channels. These channels are used to exchange **messages** between applications running on the TS server and DVC listeners running on the TS client. Sending and receiving messages is symmetrical between the client and server, and either side can initiate sending a message.

1.3.1 Relationship to Static Virtual Channels

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is implemented on top of a static virtual channel named DRDYNVC. The following diagram illustrates the wire-level encapsulation.

Figure 1: Static virtual channel objects

This is a Windows implementation detail and does not limit the definition and the description of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension. Any transport that has similar characteristics can be used to support a dynamic virtual channel implementation. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension makes use of the following features of a static virtual channel:

- Capability to indicate the reception of a complete message to the dynamic virtual channel handler.
- Capability to support a minimum message size that is sufficient for the complete reception of the PDUs used for version negotiation and channel open/close functionality.

1.3.2 Dynamic Virtual Channel Setup

The following diagram illustrates the sequence of operations involved in initializing the client and server environments.

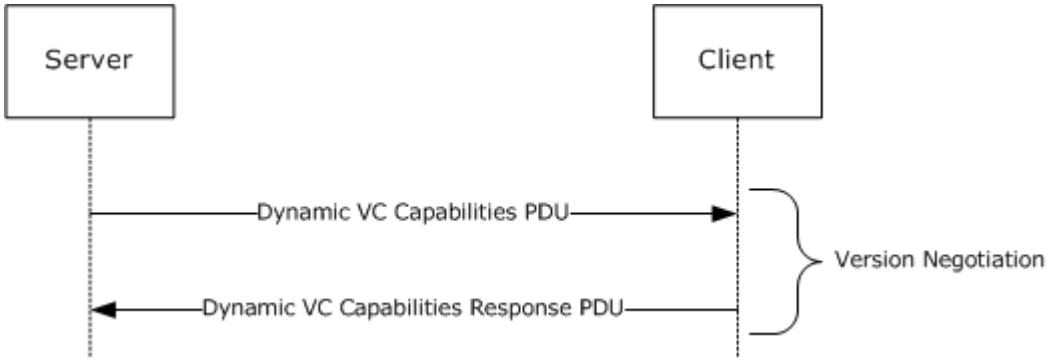


Figure 2: Dynamic virtual channel initialization sequence

The initialization is performed immediately following the establishment of a static virtual channel session, as specified in [MS-RDPBCGR].

The initialization MUST be performed once per connection. At startup and initialization, a DVC server manager performs a version negotiation with a DVC client manager over the existing static virtual channel.

The client and server initialize their environments by exchanging a capability message. The DVC server manager sends a capabilities protocol data unit (PDU) that indicates the maximum supported version level as well as any capability information that is relevant for the supported version. The capability information describes the features supported by the server.

The DVC client manager MUST respond with a capabilities response PDU that states the maximum version level that it supports. If the highest version level that the server or the client supports is version level one, no response is required by the client. The server should adjust the protocol features to match the client capabilities. After this negotiation, the DVC server manager and DVC client manager are ready to establish individual dynamic virtual channels.

1.3.3 Message Flows

1.3.3.1 Opening a Dynamic Virtual Channel

The following diagram illustrates the sequence of operations involved in the creation of a dynamic virtual channel.

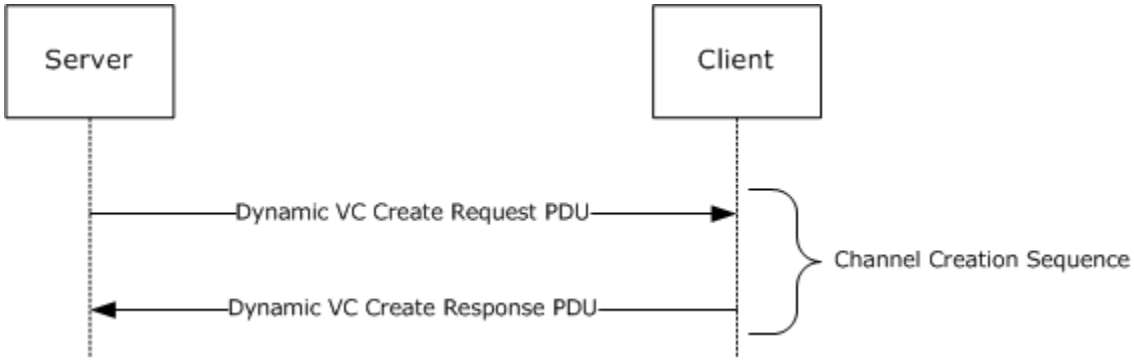


Figure 3: Dynamic virtual channel open sequence

A dynamic virtual channel consists of two endpoints logically connected over a network. One endpoint is an application running on a TS server, and the other endpoint is an application running on a TS client. The applications running on the TS client are referred to as DVC listeners. These listeners are service providers to the applications running on the TS server.

Channels are established by the DVC managers exchanging Create Request and Create Response PDUs. Channels are created by a DVC server manager in response to a channel-create request by an application. When an application makes a request to a DVC server manager to create a channel, the server generates a channel identifier (that is, a unique number for the requested session), and sends this identifier (and the listener name the application is requesting a connection to) in a Create Request PDU to the DVC client manager. The DVC client manager locates the requested listener, and the listener creates an endpoint using the **ChannelId**. The DVC client manager binds the endpoint to the **ChannelId**. The client then sends a Create Response message to the server indicating the endpoint creation status. If the creation is successful, the DVC server manager indicates to the application that the session is established and is ready for sending and receiving data. The client and server maintain the endpoints for the life of the channel.

1.3.3.2 Sending and Receiving Data

The maximum size of a message that a sender can pass to a DVC manager is 2^32 bytes. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension specifies a maximum PDU size of 1,600 bytes. A data PDU consists of a header and message data. The DVC manager is responsible for fragmenting and reassembling large messages. This is provided as a service to the sender and receiver.

The sending and receiving of messages is symmetrical between the client and server, and either side can initiate sending a message. Depending on the size of the message being sent, there are two different message sequences for sending and receiving data, as illustrated in the following diagram.

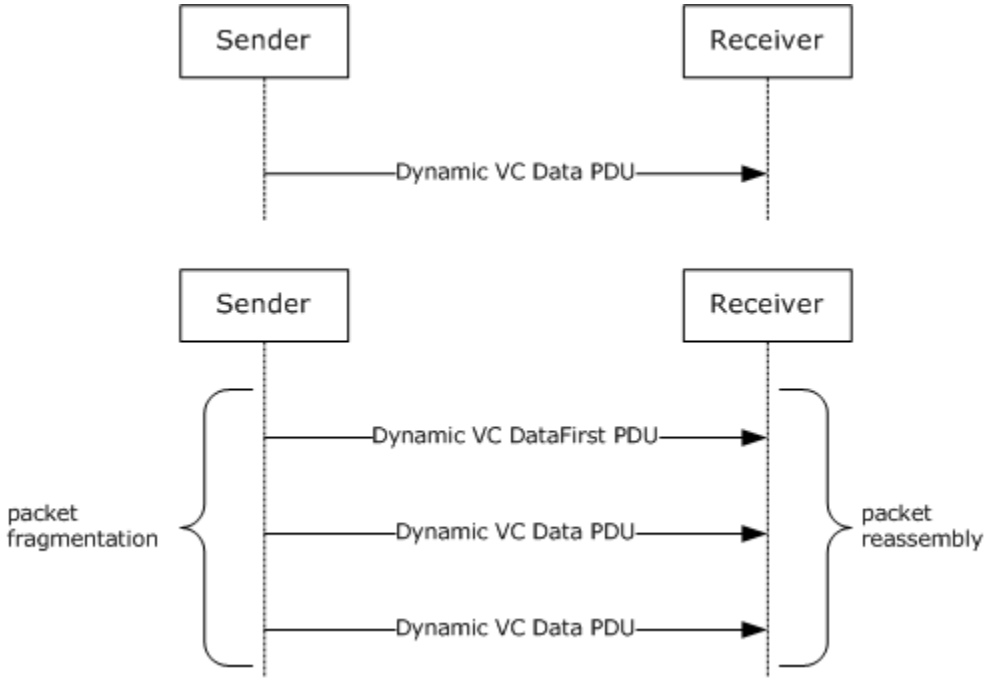


Figure 4: Send data sequence via dynamic virtual channel PDUs

1.3.3.2.1 Sending Data

If the sender requests to send a message that would not cause the PDU to exceed 1,600 bytes, then a single PDU is sent that contains the message and a header with the **Cmd** field set to indicate that the PDU type is Data.

If the sender requests to send a message that would cause a PDU to exceed 1,600 bytes, then the DVC manager fragments the message into blocks and send multiple PDUs. The first PDU contains the first fragment of message data and a header with the **Cmd** field set to indicate that the PDU type is Data First and the **Length** field set to the total length of the message the sender is sending. Subsequent PDUs of type Data are sent until the entire message is transmitted.

The receiver does not acknowledge receipt of the data.

1.3.3.2.2 Receiving Data

If a message has been fragmented, the first data PDU received will be of type Data First. If the message has not been fragmented, the first and only PDU for this message will be of type Data.

When a DVC manager receives a Data First PDU, it saves the message data and continues receiving Data PDUs until all the data is received. The DVC manager then reassembles the data and passes the data to the receiver associated with this channel.

When a DVC manager receives a Data PDU that has not been preceded by a Data First PDU, it passes the message data directly to the receiver without any additional processing.

The DVC manager does not acknowledge receipt of the data.

1.3.4 Closing a Dynamic Virtual Channel

Either an application running on the TS server or a listener running on the TS client can request that a channel be closed. The following diagram illustrates the sequence of operations involved in closing a dynamic virtual channel.

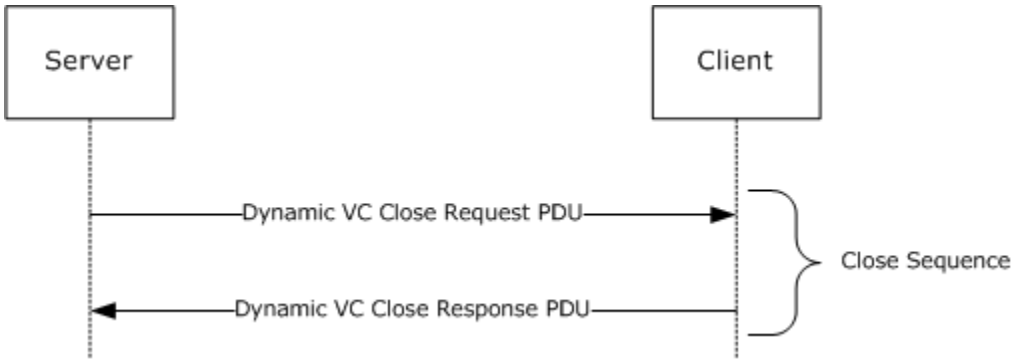


Figure 5: Dynamic virtual channel close sequence

When the DVC server manager initiates closing a channel, it sends a Close Request PDU that specifyies the **ChannelId** to the DVC client manager. The client responds with a Close Response PDU that specifies the **ChannelId**.

When a client initiates a channel-close, it sends an unsolicited Close Response PDU that specifies the **ChannelId** to the server. The server does not respond to the Client Close Response PDU.

1.3.5 Tunneling Dynamic Virtual Channel Data

A dynamic virtual channel tunnels data from multiple listeners over the same transport. The tunneling provides a basic mechanism to create and destroy dynamic virtual channels and to tag all the data flowing through as belonging to a specific dynamic virtual channel.

1.4 Relationship to Other Protocols

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is embedded in a static virtual channel transport, as specified in [MS-RDPBCGR].

1.5 Prerequisites/Preconditions

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension operates only after the static virtual channel transport (as specified in [MS-RDPBCGR]) is fully established. If the static virtual channel transport is terminated, no other communication over the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension occurs.

1.6 Applicability Statement

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is designed to be run within the context of an RDP virtual channel established between a client and a server. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is applicable when creating applications such as Plug and Play device redirection and media infrastructure layer composition engine commands.

1.7 Versioning and Capability Negotiation

The version of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension is negotiated by a DVC client manager in response to a Capabilities PDU sent by a DVC server manager. The server indicates the maximum capability it supports, and the client MUST respond with a Capabilities Response PDU that indicates the maximum capability that it can support.

1.8 Vendor-Extensible Fields

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension contains no vendor-extensible fields.

1.9 Standards Assignments

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension does not use any standards assignments.

2 Messages

The following sections specify how Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension messages are encapsulated on the wire and common data types.

2.1 Transport

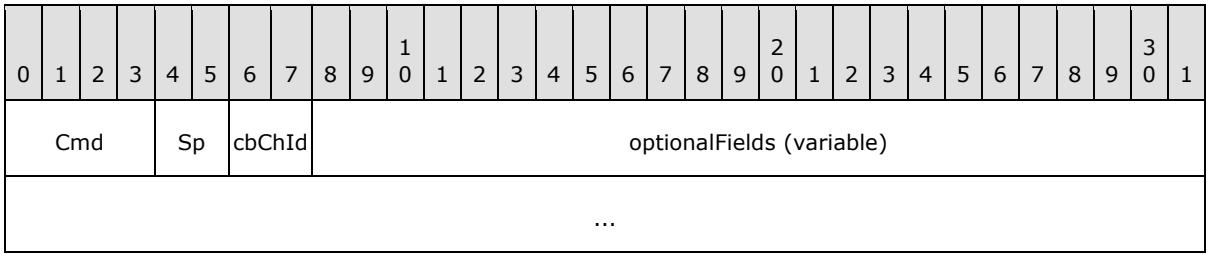
Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension messages are passed between a DVC server manager and a DVC client manager embedded within an RDP connection as a static virtual channel. The name of the channel is DRDYNVC and the bulk compression is turned on. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension does not establish any transport connections.

2.2 Message Syntax

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension consists of the following five types of messages exchanged between the server and the client.

- Capability Negotiation message
- Channel Open message
- Channel DataFirst message
- Channel Data message
- Channel Close message

Each PDU has the same 1-byte header with the **optionalFields** field following it. The **cbChId** and **Cmd** fields are common to all PDUs. The data following the PDU header depends on the type of the message and is addressed in the following sections.



Cmd (4 bits): Indicates the PDU type and MUST be set to one of the following values.

Value	Meaning
0x01	The message contained in the **optionalFields** field is a Create Request PDU (section 2.2.2.1) or a Create Response PDU (section 2.2.2.2).
0x02	The message contained in the **optionalFields** field is a Data First PDU (section 2.2.3.1).
0x03	The message contained in the **optionalFields** field is a Data PDU (section 2.2.3.2).
0x04	The message contained in the **optionalFields** field is a Close Request PDU (section 2.2.4) or a Close Response PDU (section 2.2.4).
0x05	The message contained in the **optionalFields** field is a Capability Request PDU (section

Value	Meaning
	2.2.1.1) or a Capabilities Response PDU (section 2.2.1.2).

Sp (2 bits): The value and meaning depend on the **Cmd** field.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

Value	Meaning
0x00	The **ChannelId** is 1 byte wide.
0x01	The **ChannelId** is 2 bytes wide.
0x02	The **ChannelId** is 4 bytes wide.
0x03	The **ChannelId** is 4 bytes wide.

optionalFields (variable): The data following the message header depends on the type of the message and is addressed in the following sections.

2.2.1 Initializing Dynamic Virtual Channels (DVCs)

Capabilities PDUs are exchanged to negotiate the version level of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension that is supported. Three different Capabilities PDUs are used to negotiate version-level support.

- DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU is sent by a DVC server manager to indicate it supports version 1 of the protocol.
- DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU is sent by a DVC server manager to indicate it supports version 2 of the protocol.
- DYNVC_CAPS_RSP (section 2.2.1.2) PDU is sent by a DVC client manager to acknowledge the version level it supports.

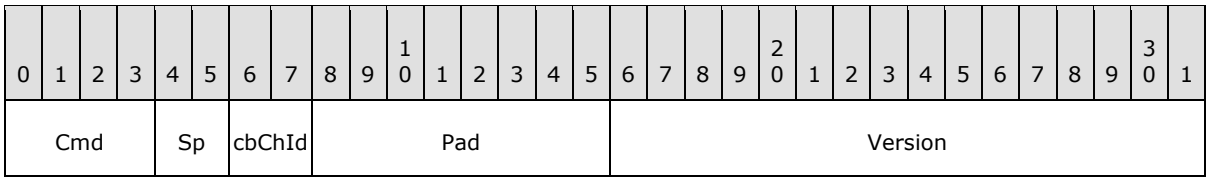
A DVC server manager initializes a dynamic virtual channel environment by sending a DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) or a DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU to the DVC client manager to indicate the highest version level supported by the server. The client MUST respond with a DYNVC_CAPS_RSP (section 2.2.1.2) PDU that indicates the highest version level supported by the client.

The DVC server manager MUST send a Capabilities message prior to creating a dynamic virtual channel and wait for a response from the client. This happens just once; if capability exchange has already been completed, the channel creation continues. The DVC client manager MUST reply with a DYNVC_CAPS_RSP (section 2.2.1.2) PDU as soon as it receives the server request.

2.2.1.1 Dynamic VC Capabilities Request PDU

2.2.1.1.1 Version 1 (DYNVC_CAPS_VERSION1)

The DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU is sent by the DVC server manager to indicate that it supports version 1 of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.



Cmd (4 bits): MUST be set to 0x05 (Capabilities).

Sp (2 bits): Unused. SHOULD be initialized to 0x00.

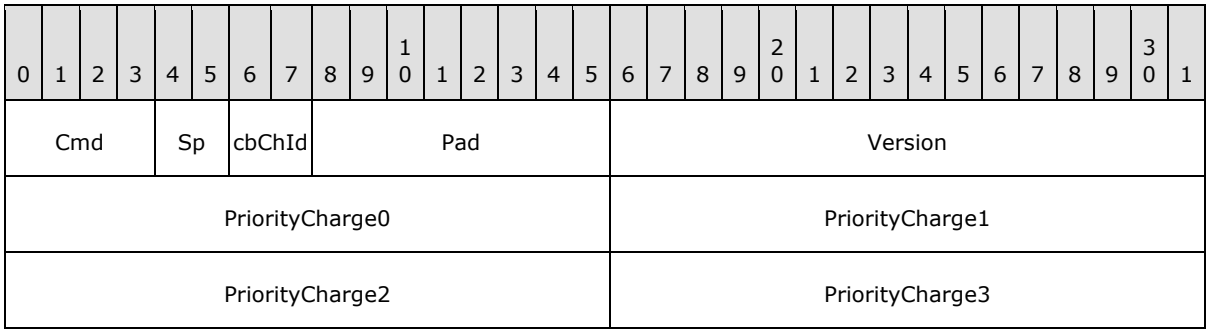
cbChId (2 bits): Unused. MUST be set to 0x00.

Pad (1 byte): An 8-bit unsigned integer. Unused. MUST be set to 0x00.

Version (2 bytes): A 16-bit unsigned integer. MUST be set to 0x0001.

2.2.1.1.2 Version 2 (DYNVC_CAPS_VERSION2)

The DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU is sent by the DVC server manager to indicate that it supports version 2 of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.



Cmd (4 bits): MUST be set to 0x05 (Capabilities).

Sp (2 bits): Unused. SHOULD be set to 0x00.

cbChId (2 bits): Unused. MUST be set to 0x00.

Pad (1 byte): An 8-bit unsigned integer. Unused. MUST be set to 0x00.

Version (2 bytes): A 16-bit unsigned integer. MUST be set to 0x0002.

PriorityCharge0 (2 bytes): A 16-bit unsigned integer. Specifies the amount of bandwidth that is allotted for each priority class, in accordance with the following algorithm.

PriorityCharge1 (2 bytes): A 16-bit unsigned integer. Specifies the amount of bandwidth that is allotted for each priority class, in accordance with the following algorithm.

PriorityCharge2 (2 bytes): A 16-bit unsigned integer. Specifies the amount of bandwidth that is allotted for each priority class, in accordance with the following algorithm.

PriorityCharge3 (2 bytes): A 16-bit unsigned integer. Specifies the amount of bandwidth that is allotted for each priority class, in accordance with the following algorithm.

The **PriorityCharge** fields determine how much bandwidth is allocated for each priority class. The percentage is calculated using the following formula.

```
Base = PriorityCharge0 * PriorityCharge1 *
    PriorityCharge2 * PriorityCharge3 /
    (PriorityCharge1 * PriorityCharge2 *
    PriorityCharge3 + PriorityCharge0 *
    PriorityCharge2 * PriorityCharge3 +
   PriorityCharge0 * PriorityCharge1 * PriorityCharge3 +
    PriorityCharge0 * PriorityCharge1 *
    PriorityCharge2)

BandwidthPriority0 = Base / PriorityCharge0
BandwidthPriority1 = Base / PriorityCharge1
BandwidthPriority2 = Base / PriorityCharge2
BandwidthPriority3 = Base / PriorityCharge3
```

Where BandwidthPriorityX is a number between 0 and 1, and the total sum of all BandwidthPriorityX values is equal to 1.

To calculate priority charges from given priorities the formula is as follows.

```
PriorityCharge0 = 65536 / (BandwidthPriority0 * 100)
PriorityCharge1 = 65536 / (BandwidthPriority1 * 100)
PriorityCharge2 = 65536 / (BandwidthPriority2 * 100)
PriorityCharge3 = 65536 / (BandwidthPriority3 * 100)
```

Where BandwidthPriorityX is a number between 0 and 1, and the total sum of all BandwidthPriorityX values is equal to 1.

For example, to have distribution for priority 0 to 3 be 70%, 20%, 7%, and 3%, the priority charges numbers are as follows.

```
PriorityCharge0 = 65536 / (0.70*100) = 936
PriorityCharge1 = 65536 / (0.20*100) = 3276
PriorityCharge2 = 65536 / (0.07*100) = 9362
PriorityCharge3 = 65536 / (0.03*100) = 21845
```

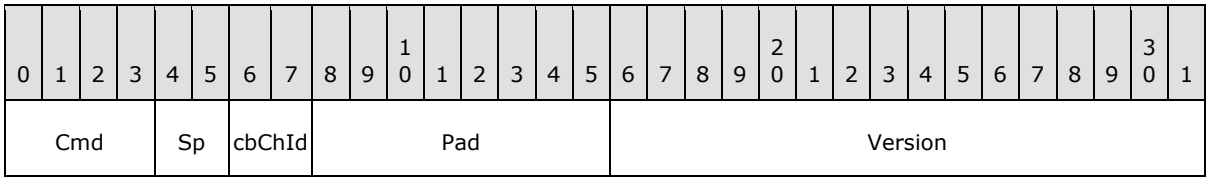
Calculating the priority from priority charges, as follows.

```
Base = 936 * 3276 * 9362 * 21845 /
    (3276 * 9362 * 21845 + 936 * 9362 *
    21845 + 936 * 3276 * 21845 + 936 *
    3276 * 9362) = 655

BandwidthPriority0 = 655/936 = 70%
BandwidthPriority1 = 655/3276 = 20%
BandwidthPriority2 = 655/9362 = 7%
BandwidthPriority3 = 655/21845 = 3%
```

2.2.1.2 Dynamic VC Capabilities Response PDU (DYNVC_CAPS_RSP)

The DYNVC_CAPS_RSP (section 2.2.1.2) PDU is sent by the DVC client manager to the DVC server manager acknowledging the version level capabilities supported.



Cmd (4 bits): MUST be set to 0x05 (Capabilities).

Sp (2 bits): Unused. MUST be set to 0x00.

cbChId (2 bits): Unused. MUST be set to 0x00.

Pad (1 byte): An 8-bit unsigned integer. Unused. MUST be set to 0x00.

Version (2 bytes): A 16-bit unsigned integer that indicates the protocol version level supported; MUST be set to the version level supported.

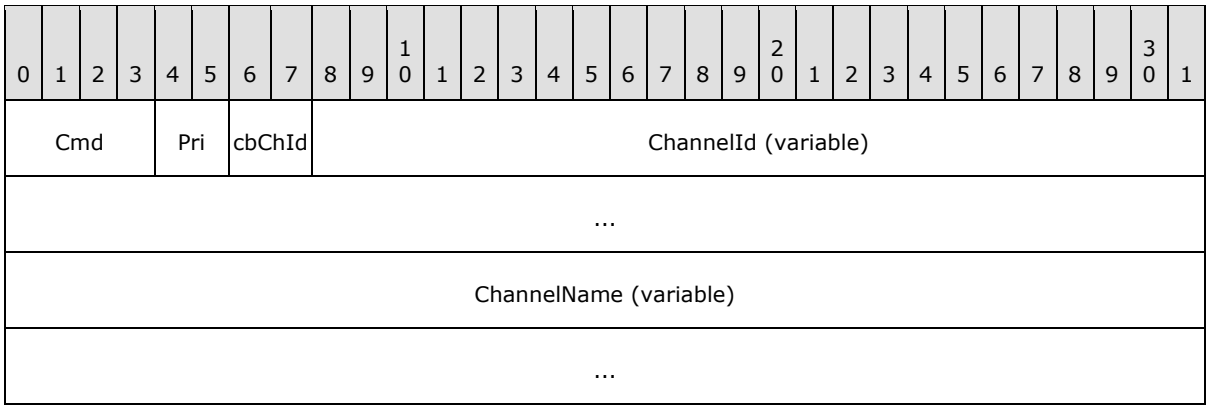
Value	Meaning
0x0001 0x1	Version level one is supported.
0x0002 0x2	Version level two is supported.

2.2.2 Opening a Dynamic Virtual Channel (DVC)

The DVC server manager initiates opening a dynamic virtual channel by exchanging Create PDUs with a DVC client manager. The server sends a DYNVC_CREATE_REQ (section 2.2.2.1) PDU to the client, and the client responds with a DYNVC_CREATE_RSP (section 2.2.2.2) PDU that indicates the status of the client endpoint creation.

2.2.2.1 Dynamic VC Create Request PDU (DYNVC_CREATE_REQ)

The DYNVC_CREATE_REQ (section 2.2.2.1) PDU is sent by the DVC server manager to the DVC client manager to request that a channel be opened.



Cmd (4 bits): MUST be set to 0x01 (Create).

Pri (2 bits): Unused. MUST be set to 0x00.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

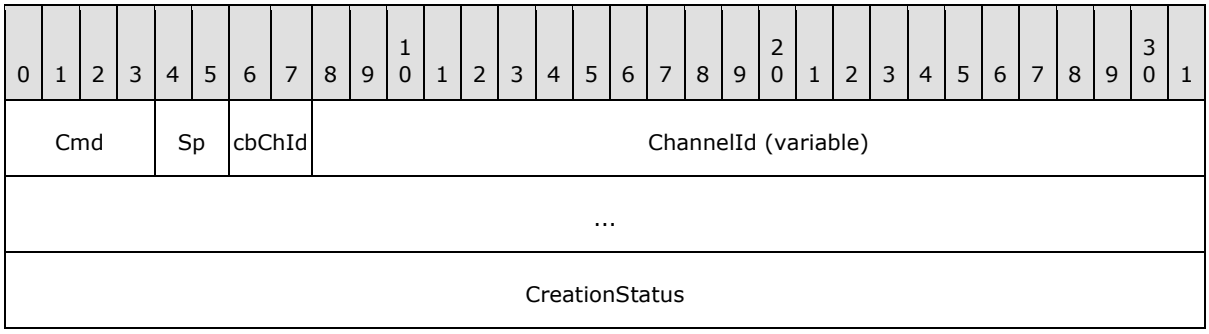
Value	Meaning
0x00	The **ChannelId** field length is 1 byte.
0x01	The **ChannelId** field length is 2 bytes.
0x02	The **ChannelId** field length is 4 bytes.
0x03	The **ChannelId** field length is 4 bytes.

ChannelId (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. A server-generated identifier for the channel being created. This number MUST be unique within a static virtual channel connection.

ChannelName (variable): A zero-terminated ANSI string. The name of the listener on the TS client with which the TS server application is requesting that a channel be opened.

2.2.2.2 Dynamic VC Create Response PDU (DYNVC_CREATE_RSP)

The DYNVC_CREATE_RSP (section 2.2.2.2) PDU is sent by the DVC client manager to indicate the status of the client endpoint create operation.



Cmd (4 bits): MUST be set to 0x01 (Create).

Sp (2 bits): Unused. SHOULD be initialized to 0x00.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

Value	Meaning
0x00	The **ChannelId** field length is 1 byte.
0x01	The **ChannelId** field length is 2 bytes.
0x02	The **ChannelId** field length is 4 bytes.
0x03	The **ChannelId** field length is 4 bytes.

ChannelId (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. Set to the value of the **ChannelId** in the DYNVC_CREATE_REQ (section 2.2.2.1) PDU.

CreationStatus (4 bytes): A 32-bit, signed integer. The status of the client endpoint creation. A zero or positive value indicates success; a negative value indicates failure.

2.2.3 Sending and Receiving Data

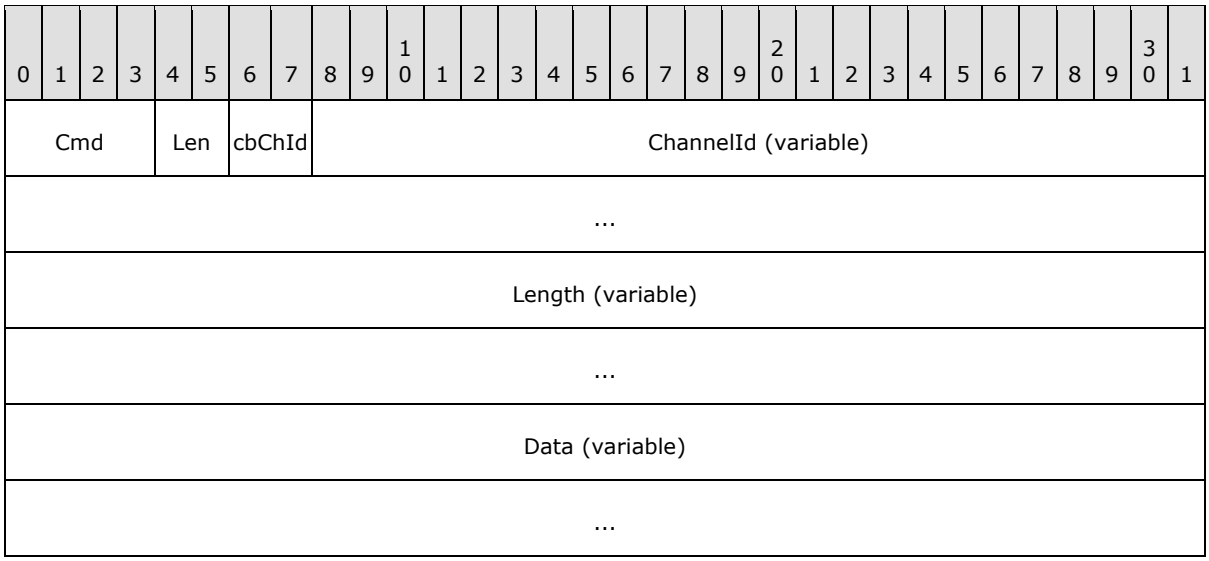
The maximum size of a message that a sender can pass to a DVC manager is 2^32 bytes. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension specifies a maximum PDU size of 1,600 bytes. A data PDU consists of a header and message data. When a message that exceeds 1,600 bytes needs to be sent by a DVC manager, the message MUST be fragmented. Two different PDUs are used to send data, as follows:

- DYNVC_DATA_FIRST (section 2.2.3.1)
- DYNVC_DATA (section 2.2.3.2)

For an overview of sending and receiving data, see section 1.3.3.2.

2.2.3.1 Dynamic VC Data First PDU (DYNVC_DATA_FIRST)

The DYNVC_DATA_FIRST (section 2.2.3.1) PDU is used to send the first block of data of a fragmented message. It MUST be the first PDU sent when a message has been fragmented. The total length of the message to be sent is indicated in the **Length** field, and the data field contains the first block of the fragmented data.



Cmd (4 bits): This field MUST be set to 0x02 (Data First).

Len (2 bits): Indicates the length of the **Length** field.

Value	Meaning
0x0	**Length** field length is 1 byte.
0x1	**Length** field length is 2 bytes.
0x2	**Length** field length is 4 bytes.
0x3	**Length** field length is 4 bytes.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

Value	Meaning
0x0	**ChannelId** field length is 1 byte.
0x1	**ChannelId** field length is 2 bytes.
0x2	**ChannelId** field length is 4 bytes.
0x3	**ChannelId** field length is 4 bytes.

ChannelId (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. Set to the value of the **ChannelId** associated with the endpoint of the channel on which the PDU is being sent.

Length (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. Set to total length of the message to be sent.

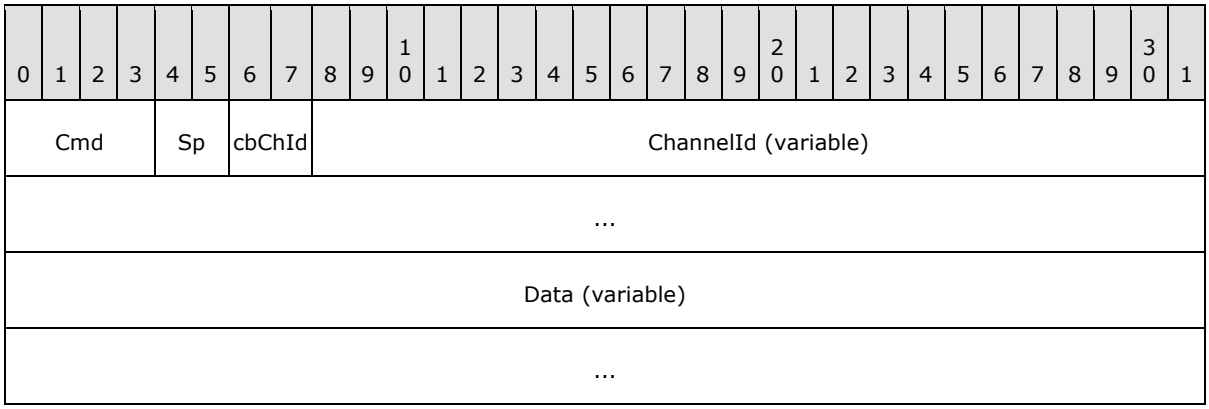
Data (variable): A variable length 8-bit unsigned integer. The first block of data of a fragmented message. Message data is sent as 8-bit unsigned integers. The length of the data in this field is equal to 1,600 bytes minus the sum of the sizes of the previous fields.

2.2.3.2 Dynamic VC Data PDU (DYNVC_DATA)

The DYNVC_DATA (section 2.2.3.2) PDU is used to send both single messages and blocks of fragmented messages.

A single DYNVC_DATA (section 2.2.3.2) PDU is used to send a message when the total length of the message plus the length of the PDU header is less than or equal to 1,600 bytes.

Multiple DYNVC_DATA (section 2.2.3.2) PDUs are used to send messages that have been fragmented and are sent subsequent to a DYNVC_DATA_FIRST (section 2.2.3.1) PDU. DYNVC_DATA (section 2.2.3.2) PDUs are sent until the entire fragmented message has been sent.



Cmd (4 bits): MUST be set to 0x03 (Data).

Sp (2 bits): Unused. SHOULD be initialized to 0x00.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

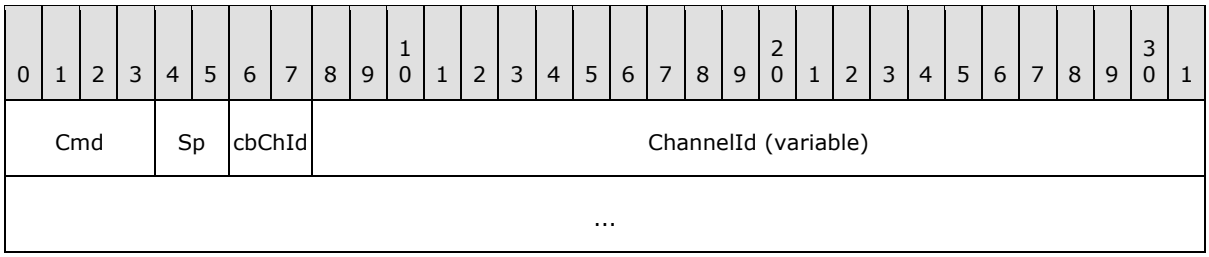
Value	Meaning
0x00	**ChannelId** field length is 1 byte.
0x01	**ChannelId** field length is 2 bytes.
0x02	**ChannelId** field length is 4 bytes.
0x03	**ChannelId** field length is 4 bytes.

ChannelId (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. Set to the value of the **ChannelId** associated with the endpoint of the channel upon which the PDU is being sent.

Data (variable): A variable length 8-bit unsigned integer that provides message data. Message data is sent as 8-bit unsigned integers.

2.2.4 Closing a Dynamic Virtual Channel (DYNVC_CLOSE)

A DYNVC_CLOSE (section 2.2.4) PDU is sent by either a DVC server manager or a DVC client manager to close a dynamic virtual channel. A DYNVC_CLOSE (section 2.2.4) PDU is used for both a close request and a close response.



Cmd (4 bits): MUST be set to 0x04. (Close)

Sp (2 bits): Unused. SHOULD be initialized to 0x00.

cbChId (2 bits): Indicates the length of the **ChannelId** field.

Value	Meaning
0x00	**ChannelId** field length is 1 byte.
0x01	**ChannelId** field length is 2 bytes.
0x02	**ChannelId** field length is 4 bytes.
0x03	**ChannelId** field length is 4 bytes.

ChannelId (variable): A variable length 8-bit, 16-bit, or 32-bit unsigned integer. Set to the value of the **ChannelId** associated with the endpoint of the channel that is being closed.

3 Protocol Details

The following sections specify details of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension, including common, client, and server abstract data models and message processing rules.

3.1 Common Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

No common timers are used.

3.1.3 Initialization

Before protocol operation can commence, the static virtual channel must be established using the parameters specified in section 2.1.

The TS server DVC manager begins the initialization sequence immediately following the establishment of a static virtual channel session, as specified in [MS-RDPBCGR]. A TS server DVC manager and a TS client DVC manager exchange Capabilities PDUs and initialize themselves to the version level negotiated. Individual dynamic virtual channels are created by the DVC server manager exchanging Channel Create messages with the DVC client manager. Details for initialization and channel creation are specified in their respective sections.

3.1.4 Higher-Layer Triggered Events

There are no common events specified for the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.

3.1.5 Message Processing Events and Sequencing Rules

Sending and receiving messages is symmetrical between the DVC server manager and the DVC client manager. After the server creates a dynamic virtual channel, applications running on either the server or the client can initiate sending a message. The PDUs and the sending sequence are the same regardless of who initiates sending the message.

Messages are sent and received using two different PDUs depending on the total length of the message being sent.

- DYNVC_DATA_FIRST (section 2.2.3.1)
- DYNVC_DATA (section 2.2.3.2)

The maximum message size that an application can pass to a DVC manager for sending to a receiver at one time is 2^32 bytes. The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension specifies a maximum PDU size of 1,600 bytes. A data PDU consists of a header and message data. Large messages are fragmented by the sending DVC manager and reassembled by the receiving DVC manager. This is provided as a service to the requesting application.

3.1.5.1 Sending Data

3.1.5.1.1 Dynamic VC Data First (DYNVC_DATA_FIRST)

When the total length of the message being sent causes the PDU to exceed 1,600 bytes, the DYNVC_DATA_FIRST (section 2.2.3.1) PDU is sent as the first data PDU. The **Length** field is set to the total length of the message, and the **Data** field contains the first block of fragmented data. Subsequent DYNVC_DATA (section 2.2.3.2) PDUs are sent until all the data has been sent.

3.1.5.1.2 Dynamic VC Data (DYNVC_DATA)

The DYNVC_DATA (section 2.2.3.2) PDU is used to send data when the total length of the PDU message plus the PDU header does not exceed 1,600 bytes.

3.1.5.2 Receiving Data

When the DVC manager receives a PDU, it checks the **Cmd** field to determine the type of PDU that has been sent. If the **Cmd** field is set to 0x02, the PDU type is DYNVC_DATA_FIRST (section 2.2.3.1). The **Length** field indicates the total length of the message that is being sent, and the data field contains the first block of the fragmented message. The DVC manager stores the data and reads the next data PDU.

3.1.5.2.1 Dynamic VC Data First (DYNVC_DATA_FIRST)

When the DVC manager receives a PDU, it checks the **Cmd** field to determine the type of PDU that has been sent. If the **Cmd** field is set to 0x02, the PDU type is DYNVC_DATA_FIRST (section 2.2.3.1). The **Length** field indicates the total length of the message that is being sent, and the data field contains the first block of the fragmented message. The DVC manager stores the data and reads the next data PDU.

3.1.5.2.2 Dynamic VC Data (DYNVC_DATA)

When the DVC manager receives a PDU, it checks the **Cmd** field to determine the type of PDU that has been sent. If the **Cmd** field is set to 0x03, the PDU type is DYNVC_DATA (section 2.2.3.2). This PDU is used to send blocks of fragmented messages or one complete nonfragmented message.

If a DYNVC_DATA (section 2.2.3.2) PDU is preceded by a DYNVC_DATA_FIRST (section 2.2.3.1) PDU, the receiver stores this data with the previously received data and continues to receive DYNVC_DATA (section 2.2.3.2) PDUs until the entire message has been received. The DVC manager reassembles the fragmented data and passes a complete message to the receiver.

If a DYNVC_DATA (section 2.2.3.2) PDU is not preceded by a DYNVC_DATA_FIRST (section 2.2.3.1) PDU, the receiver passes the message in the **Data** field directly to the receiver without further processing.

3.1.6 Timer Events

There are no common timer events.

3.1.7 Other Local Events

There are no local events specified for the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.

3.2 Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

There are no client timers.

3.2.3 Initialization

3.2.3.1 Dynamic Virtual Channel (DVC) Client Manager Initialization

The Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension encompasses two version levels. The server specifies priority charges in Version 2 (as described in section 2.2.1.1.2); no such priority charges are specified in Version 1. If the client supports channel priorities it MUST set the **Version** field of DYNVC_CAPS_RSP (section 2.2.1.2) to 2.

Capabilities PDUs are exchanged to negotiate the version level of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension that is supported. Three different Capabilities PDUs are used to negotiate version level support.

- DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU is sent by a server to indicate it supports version 1 of this protocol.
- DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU is sent by a server to indicate it supports version 2 of this protocol.
- DYNVC_CAPS_RSP (section 2.2.1.2) PDU is sent by a client to acknowledge the version level it supports.

A Capabilities PDU has the **Cmd** field set to 0x05.

When a DVC client manager receives a Capabilities Request, it checks the **Version** field to determine what protocol version level the DVC server manager is requesting support for. The client MUST respond with a DYNVC_CAPS_RSP (section 2.2.1.2) PDU indicating the highest version level supported by the client.

3.2.3.1.1 Version Level 1 (DYNVC_CAPS_VERSION1)

A DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU has the **Version** field set to 0x01.

If the **Version** field is set to 0x01, the client does not need to respond to the message.

3.2.3.1.2 Version Level 2 (DYNVC_CAPS_VERSION2)

A **DYNVC_CAPS_VERSION2** PDU has the **Version** field set to 0x02.

A DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU specifies a message priority class. This feature is currently unused.

3.2.3.1.3 Capabilities Response (DYNVC_CAPS_RSP)

If a client receives a DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU or a DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU from the server it MUST reply with a Capabilities Response PDU (section 2.2.1.2).

3.2.3.2 Dynamic Virtual Channel (DVC) Initialization

3.2.3.2.1 Dynamic VC Create Response (DYNVC_CREATE_RSP)

When a DVC client manager receives a DYNVC_CREATE_REQ (section 2.2.2.1) PDU from the DVC server manager, it uses the listener name and **ChannelId** to create a named endpoint. The DVC client manager uses the listener name to locate a listener on the TS client that has advertised itself as being available to accept connections. A client-side endpoint is created that associates the **ChannelId** with the specified listener. After the endpoint is created, the listener name is no longer used and all data is sent referencing the **ChannelId**.

The client maintains this endpoint for the life of the connection.

The client responds to the server with a DYNVC_CREATE_RSP (section 2.2.2.2) PDU indicating the endpoint creation status. Any positive or zero value indicates success. A negative value indicates failure.

3.2.4 Higher-Layer Triggered Events

- Channel Close Request: Sent by a listener to the DVC client manager requesting that a dynamic virtual channel be closed.
- Send-Data Request: A message passed to the DVC client manager for sending over a dynamic virtual channel.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Sending and Receiving Data

Sending and receiving messages is symmetrical between the DVC server manager and the DVC client manager. After the server creates a dynamic virtual channel, applications running on either the server or the client can initiate sending a message. The PDUs and the sending sequence are the same regardless of who initiates sending the message. The sending and receiving of data is specified in section 3.1.5.

3.2.5.2 Closing a Dynamic Virtual Channel (DYNVC_CLOSE)

A channel can be closed by either a DVC client manager or a DVC server manager. A DYNVC_CLOSE (section 2.2.4) PDU is used for both a close request and a close response.

When a DVC client manager receives a DYNVC_CLOSE (section 2.2.4) PDU, the client MAY respond with a DYNVC_CLOSE (section 2.2.4) PDU specifying the **ChannelId**.

When a client initiates a channel-close, it sends an unsolicited DYNVC_CLOSE (section 2.2.4) PDU specifying the **ChannelId** to the server. The server does not respond to the DYNVC_CLOSE (section 2.2.4) PDU.

Upon closing a channel, the client SHOULD remove the endpoint from the list of active endpoints.

3.2.6 Timer Events

There are no client timer events.

3.2.7 Other Local Events

There are no other local events used by the client.

3.3 Server Details

3.3.1 Abstract Data Model

None.

3.3.2 Timers

The DVC server manager sets a timer when it is initializing a dynamic virtual channel. The timer is set to expire 10 seconds after the server sends a Capabilities Request to the DVC client manager. The server sends either a DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) or a DYNVC_CAPS_VERSION2 (section 2.2.1.1.2), depending on the protocol version level it supports. If the client does not respond to the Capabilities Request within 10 seconds, the server fails the creation of the dynamic virtual channel.

3.3.3 Initialization

Before protocol operation can commence, the static virtual channel must be established using the parameters specified in section 2.1.

3.3.3.1 Dynamic Virtual Channel (DVC) Server Manager Initialization

The DVC managers on the TS server and the TS client initialize themselves by exchanging Capabilities PDUs to negotiate the version level of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension that is supported. Three different Capabilities PDUs are used to negotiate version level support.

- DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU is sent by a server to indicate it supports version 1 of this protocol.
- DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU is sent by a server to indicate it supports version 2 of this protocol.
- DYNVC_CAPS_RSP (section 2.2.1.2) PDU is sent by a client to acknowledge the version level it supports.

A Capabilities PDU has the **Cmd** field set to 0x05.

Immediately following the establishment of a static virtual channel session, as specified in [MS-RDPBCGR], the DVC server manager begins the initialization sequence. The DVC server manager sets a 10-second timer and sends a DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) or a DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU to the DVC client manager to indicate its highest version level supported. The DVC client manager MUST respond with a DYNVC_CAPS_RSP (section 2.2.1.2) PDU indicating the highest version level supported by the client. If the client does not respond to a Capabilities Request PDU before the 10-second timer expires, the server fails the creation of the dynamic virtual channel.

The DVC server manager MUST adjust to the version level supported by the DVC client manager.

3.3.3.1.1 Version Level 1 (DYNVC_CAPS_VERSION1)

A DYNVC_CAPS_VERSION1 (section 2.2.1.1.1) PDU is sent to indicate that the highest protocol version level that the DVC server manager supports is version level 1. For PDU initialization see section Version Level 1 (DYNVC_CAPS_VERSION1) (section 3.2.3.1.1).

3.3.3.1.2 Version Level 2 (DYNVC_CAPS_VERSION2)

A DYNVC_CAPS_VERSION2 (section 2.2.1.1.2) PDU is sent to indicate that the highest protocol version level the DVC server manager supports is version level 2. For PDU initialization see section Version Level 2 (DYNVC_CAPS_VERSION2) (section 3.2.3.1.2).

3.3.3.1.3 Capabilities Response (DYNVC_CAPS_RSP)

The DYNVC_CAPS_RSP (section 2.2.1.2) PDU is sent by the DVC client manager to indicate the protocol version level it supports.

The version level supported is indicated by the value of the **Version** field. If the client doesn't respond with a Capabilities Response packet within 10 seconds, the server will not make any further attempts to send a Dynamic VC Create Request packet.

3.3.3.2 Dynamic Virtual Channel (DVC) Initialization

A dynamic virtual channel consists of two endpoints logically connected over a static virtual channel. A DVC server manager initializes a channel by exchanging Create PDUs with a DVC client manager.

Two different Create PDUs are used to open a channel:

- DYNVC_CREATE_REQ (section 2.2.2.1)
- DYNVC_CREATE_RSP (section 2.2.2.2)

A DVC server manager creates a channel in response to a request by an application running on the TS server to create a channel to a listener running on the TS client. The server creates a channel endpoint for the requesting application using a server-generated channel identifier. The server sends the **ChannelId**, the listener name, and a channel priority to the DVC client manager. The server maintains this endpoint identifier for the life of the channel.

The DVC client manager responds to the DVC server manager with a DYNVC_CREATE_RSP (section 2.2.2.2) PDU indicating the endpoint creation status. Any positive or zero value indicates success. A negative value indicates failure.

The listener name is only used at channel initialization. After the channel is initialized, all data is sent referencing the **ChannelId**.

The Channel ID MUST be unique within a static virtual channel session.

There is no processing specified by the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension for a priority class.

3.3.4 Higher-Layer Triggered Events

- Static virtual channel created: Immediately upon creation of a static virtual channel, as specified in [MS-RDPBCGR], the DVC server manager begins the initialization sequence.

- Channel creation request: Sent by an application to the DVC server manager requesting the creation of a dynamic virtual channel with a listener running on a TS client.
- Channel close request: Sent by an application to the DVC server manager requesting the closure of a dynamic virtual channel.
- Send-Data request: A message passed to the DVC server manager for sending over a dynamic virtual channel.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Sending and Receiving Data

Sending and receiving messages is symmetrical between the DVC server manager and the DVC client manager. After the server creates a dynamic virtual channel, applications running on either the server or the client can initiate sending a message. The PDUs and the sending sequence are the same regardless of who initiates sending the message. The sending and receiving of data is specified in section 3.1.5.

3.3.5.2 Closing a Dynamic Virtual Channel (DYNVC_CLOSE)

The closure of a channel can be requested by either an application running on the TS server or a listener running on the TS client. A DYNVC_CLOSE (section 2.2.4) PDU is used for both a close request and a close response.

The DVC server manager sends a DYNVC_CLOSE (section 2.2.4) PDU (specifying the **ChannelId** to close) to the DVC client manager. The client optionally replies with a DYNVC_CLOSE (section 2.2.4) PDU.

When a DVC client manager initiates a channel-close, it sends an unsolicited DYNVC_CLOSE (section 2.2.4) PDU specifying the **ChannelId** to the server. The server does not respond to the DYNVC_CLOSE (section 2.2.4) PDU.

Upon closing the channel, the DVC server manager SHOULD remove the endpoint from the list of active endpoints.

3.3.6 Timer Events

There are no timer events used by the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.

3.3.7 Other Local Events

There are no other local events.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension Specification.

4.1 Annotated Initializing Dynamic Virtual Channels

4.1.1 Dynamic VC Capabilities Request (Version2) PDU

The following is an annotated dump of the DYNVC_CAPS_VERSION2 PDU (section 2.2.1.1.2).

```
00000000 58 00 02 00 33 33 11 11 3d 0a a7 04             X...33..=...

58 -> Header bitmask fields

0 - --\
1 -   | DYNVC_CAPS_VERSION2::Cmd = Capabilities (5)
0 -   |
1 - --/
1 - --\ DYNVC_CAPS_VERSION2::Sp = 2
0 - --/
0 - --\ DYNVC_CAPS_VERSION2::cbChId = 0
0 - --/

00 -> Pad

02 00 -> DYNVC_CAPS_VERSION2::Version = 0x0002 = 2

33 33 -> DYNVC_CAPS_VERSION2::PriorityCharge0 = 0x3333 = 13107 (~5%)
11 11 -> DYNVC_CAPS_VERSION2::PriorityCharge1 = 0x1111 = 4369 (~15%)
3d 0a -> DYNVC_CAPS_VERSION2::PriorityCharge2 = 0x0a3d = 2621 (~25%)
a7 04 -> DYNVC_CAPS_VERSION2::PriorityCharge3 = 0x04a7 = 1191 (~55%)
```

4.1.2 Dynamic VC Capabilities Response PDU

The following is an annotated dump of the DYNVC_CAPS_RSP PDU (section 2.2.1.2).

```
00000000 50 00 02 00

50 -> Header bitmask fields

0 - --\
1 -   | DYNVC_CAPS_RSP::Cmd = Capabilities (5)
0 -   |
1 - --/
0 - --\ DYNVC_CAPS_RSP::Sp = 0
0 - --/
0 - --\ DYNVC_CAPS_RSP::cbChId = 0
0 - --/

00 -> Pad

02 00 -> DYNVC_CAPS_RSP::Version = 0x0002 = 2
```

4.2 Annotated Opening a Dynamic Virtual Channel

4.2.1 Dynamic VC Create Request PDU

The following is an annotated dump of the DYNVC_CREATE_REQ PDU (section 2.2.2.1).

```
00000000 10 03 74 65 73 74 64 76 63 00                   ..testdvc.

10 -> Header bitmask fields

0 - --\
0 -   | DYNVC CREATE REQ::Cmd = Create (1)
0 -   |
1 - --/
0 - --\ DYNVC_CREATE_REQ::Pri = 0
0 - --/
0 - --\ DYNVC CREATE REQ::cbChId = 0
0 - --/

03 -> DYNVC_CREATE_REQ::ChannelId

74 65 73 74 64 76 63 00 -> DYNVC CREATE REQ::ChannelName = "testdvc" (null terminated
ANSI string)
```

4.2.2 Dynamic VC Create Response PDU

The following is an annotated dump of the DYNVC_CREATE_RSP PDU (section 2.2.2.2).

```
00000000 10 03 00 00 00 00                               ......

10 -> Header bitmask fields

0 - --\
0 -   | DYNVC_CREATE_RSP::Cmd = Create (1)
0 -   |
1 - --/
0 - --\ DYNVC_CREATE_RSP::Sp = 0
0 - --/
0 - --\ DYNVC_CREATE_RSP::cbChId = 0
0 - --/

03 -> DYNVC_CREATE_RSP::ChannelId

00 00 00 00 -> DYNVC_CREATE_RSP::CreationStatus = 0x00000000
```

4.3 Annotated Sending and Receiving Data

4.3.1 Dynamic VC Data First PDU

The following is an annotated dump of the DYNVC_DATA_FIRST PDU (section 2.2.3.1).

```
00000000 24 03 7b 0c 71 71 71 71 71 71 71 71 71 71 71 71 $.{.qqqqqqqqqqqq
00000010 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000020 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000030 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000040 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
```

```
00000050 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000060 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000070 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000080 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000090 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000100 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000110 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000120 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000130 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000140 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000150 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000160 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000170 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000180 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000190 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000200 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000210 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000220 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000230 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000240 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000250 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000260 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000270 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000280 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000290 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000300 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000310 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000320 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000330 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000340 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000350 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000360 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000370 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000380 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000390 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000400 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000410 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000420 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000430 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000440 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000450 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
```

```
00000460 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000470 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000480 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000490 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000500 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000510 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000520 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000530 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000540 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000550 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000560 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000570 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000580 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000590 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000600 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000610 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000620 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000630 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq

24 -> Header bitmask fields

0 - --\
0 -   | DYNVC DATA FIRST::Cmd = DataFirst (2)
1 -   |
0 - --/
0 - --\ DYNVC_DATA_FIRST::Sp = 1
1 - --/
0 - --\ DYNVC DATA FIRST::cbChId = 0
0 - --/

03 -> DYNVC_DATA_FIRST::ChannelId = 0x3

7b 0c -> DYNVC DATA FIRST::Length = 0n0c7b = 3195 bytes

71 .... 71 -> DYNVC_DATA_FIRST::Data = (1600 - 4) bytes of actual payload
```

4.3.2 Dynamic VC Data PDU

The following is an annotated dump of the DYNVC_DATA PDU (section 2.2.3.2).

```
00000000 34 03 71 71 71 71 71 71 71 71 71 71 71 71 71 71 4.qqqqqqqqqqqqqq
00000010 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000020 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000030 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000040 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000050 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000060 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000070 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000080 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000090 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
```

```
000000b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000000f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000100 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000110 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000120 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000130 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000140 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000150 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000160 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000170 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000180 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000190 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000001f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000200 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000210 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000220 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000230 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000240 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000250 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000260 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000270 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000280 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000290 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000002f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000300 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000310 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000320 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000330 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000340 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000350 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000360 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000370 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000380 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000390 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000003f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000400 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000410 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000420 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000430 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000440 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000450 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000460 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000470 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000480 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000490 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
```

```
000004c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000004f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000500 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000510 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000520 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000530 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000540 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000550 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000560 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000570 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000580 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000590 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005a0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005b0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005c0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005d0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005e0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
000005f0 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000600 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000610 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000620 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq
00000630 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 qqqqqqqqqqqqqqqq

34 -> Header bitmask fields

0 - --\
0 -   | DYNVC DATA::Cmd = Data (3)
1 -   |
1 - --/
1 - --\ DYNVC_DATA::Sp = 1
0 - --/
0 - --\ DYNVC DATA::cbChId = 0
0 - --/

03 -> DYNVC DATA::ChannelId = 0x3

71 .... 71 -> DYNVC_DATA::Data = (1600 - 2) bytes of actual payload
```

4.4 Annotated Closing a Dynamic Virtual Channel

4.4.1 Dynamic VC Close PDU

The following is an annotated dump of the DYNVC_CLOSE PDU (section 2.2.4).

```
00000000 40 03                                                  @.

40 -> Header bitmask fields

0 - --\
1 -   | DYNVC_CLOSE::Cmd = Close (4)
0 -   |
0 - --/
0 - --\ DYNVC_CLOSE::Sp = 0
0 - --/
0 - --\ DYNVC_CLOSE::cbChId = 0
0 - --/
```

```
03 -> DYNVC_CLOSE::ChannelId = 0x3
```

5 Security

The following sections specify security considerations for implementers of the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.

5.1 Security Considerations for Implementers

There are no security considerations for Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension messages because all static virtual channel traffic is secured by the underlying RDP core protocol. For an overview of the implemented security-related mechanisms, see [MS-RDPBCGR] section 5.

5.2 Index of Security Parameters

There are no security parameters in the Remote Desktop Protocol: Dynamic Channel Virtual Channel Extension.

6 Appendix A: Windows Behavior

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

- Windows Vista

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

7 Index