[MS-RDPECLIP]:
Remote Desktop Protocol:
Clipboard Virtual Channel Extension

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.01		MCPP Milestone Longhorn Initial Availability
07/03/2007	1.0	Major	MLonghorn+90
07/20/2007	1.0.1	Editorial	Revised and edited the technical content.
08/10/2007	1.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
09/28/2007	1.0.3	Editorial	Revised and edited the technical content.
10/23/2007	1.0.4	Editorial	Revised and edited the technical content.
11/30/2007	2.0	Major	Updated and revised the technical content.
01/25/2008	3.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

The goal of the Remote Desktop Protocol: Clipboard Virtual Channel Extension is to allow users to seamlessly transfer data, via the system clipboard, between applications that are running on different computers. To accomplish this objective, the Remote Desktop Protocol: Clipboard Virtual Channel Extension specifies how to keep two distinct system clipboards in sync so that at any given time, the data available to an application on one computer (via its local clipboard) is identical to the data available to another application on a remote computer (via its local clipboard).

1.1 Glossary

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

ANSI
Client
Endpoint
Little-Endian
Peer
Protocol Data Unit (PDU)
Server
Unicode

The following terms are specific to this document:

Virtual Channel: A static transport used for lossless communication between a **client** and a **server** component over a main data connection, in 1600-byte chunks. More details are specified in [MS-RDPBCGR] section 1.3.3.

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

1.2 References

1.2.1 Normative References

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

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

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

[MS-WMF] Microsoft Corporation, "Windows Metafile Format 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-META] Microsoft Corporation, "Metafiles", http://msdn2.microsoft.com/en-us/library/ms534574.aspx

[MSDN-SHELLCLIP] Microsoft Corporation, "Shell Clipboard Formats", http://msdn2.microsoft.com/en-us/library/aa969394.aspx

1.3 Protocol Overview (Synopsis)

This section describes the fundamentals of the system clipboard and gives a high-level overview of the operation of the Remote Desktop Protocol: Clipboard Virtual Channel Extension.

1.3.1 Clipboard Basics

The system clipboard provided by modern operating systems allows users to transfer data of various formats between applications that are running on the same computer.

To copy data from one application to another, a user first places a selection of data onto the clipboard. This is called a "copy operation." The user then switches to another application and, after having navigated to an appropriate location within this application, the data is extracted from the clipboard and inserted into the target location. This is called a "paste" operation.

The copy and paste operations are two distinct actions and form the basis for clipboard manipulation. The copy operation always takes place before the paste operation to guarantee that the correct data is transferred. The passage of time between the copy and paste operations does not affect the outcome of the transfer of data.

System clipboards are not limited to holding one particular format of data at a given time. Instead, they provide the ability to store data of various formats simultaneously. For example, assume that a user has copied some text to the clipboard. The application that places the text onto the clipboard can place multiple formats of that text onto the clipboard, such as **Unicode**, **ANSI**, or a graphical image representation. This flexibility enables a wide spectrum of applications to extract data from the clipboard. For example, some applications may be able to manipulate only ANSI text. By allowing both Unicode and ANSI formats to coexist on the clipboard, ANSI-only applications can obtain text from applications that natively manipulate Unicode characters, but are flexible enough to place data onto the clipboard in a wider array of formats.

Basic programmatic access to the clipboard provided by an operating system usually ensures that any application has the ability to do the following:

- Place data onto the clipboard.
- Extract data from the clipboard.
- Enumerate the data formats available on the clipboard.
- Register to receive notifications when the system clipboard is updated.

Applications that leverage operating system-supplied clipboard functionality can share data seamlessly, provided that the data is of the appropriate format. Of course, if an application does not make use of system-supplied clipboard functionality, its ability to exchange data with other applications is constrained and limited to ad-hoc or proprietary mechanisms.

1.3.1.1 Data Types

Data placed onto the clipboard can conform to any format and applications can use this data as long as they are able to correctly interpret the format. The type of data that can be transferred by using the Remote Desktop Protocol: Clipboard Virtual Channel Extension is divided into four categories:

- Generic data (section 1.3.1.1.1)
- Palette data (section 1.3.1.1.2)
- Metafile data (section 1.3.1.1.3)
- File stream data (section 1.3.1.1.4)

These four classes of data are the only data formats manipulated by the Remote Desktop Protocol: Clipboard Virtual Channel Extension.

1.3.1.1.1 Generic

Generic data is not manipulated or re-encoded by the Remote Desktop Protocol: Clipboard Virtual Channel Extension. It is treated as opaque and passed from one **virtual channel endpoint** to another without any modification.

1.3.1.1.2 Palette

Palette data contains a predefined set of mappings from a given index to a red, green, and blue (RGB) triplet. Each triplet represents a color in the additive RGB color space. Palette data to be transferred between virtual channel endpoints is specially encoded for transport on the wire by the Remote Desktop Protocol: Clipboard Virtual Channel Extension.

1.3.1.1.3 Metafile

A Windows metafile (as specified in [MS-WMF]) is a collection of structures that can store an image in an application-independent format. The stored image can be recreated by processing the metafile structures. Also called a vector image, a metafile contains a sequence of drawing commands and settings. The commands and settings recorded in a metafile object can be rendered on a display, output by a printer or plotter, stored in memory, or saved to a file or stream. Metafile data to be transferred between virtual channel endpoints is specially encoded and decoded for transport on the wire by the Remote Desktop Protocol: Clipboard Virtual Channel Extension.

1.3.1.1.4 File Stream

A file stream encapsulates the contents of a file that resides on some form of long-term storage. The Remote Desktop Protocol: Clipboard Virtual Channel Extension provides the ability to transfer selected chunks of a file between virtual channel endpoints, as opposed to having to transfer the entire file image. A file stream can also be part of a larger collection of streams, where each stream can be referenced independently (for example, when transferring a group of files).

1.3.1.2 Clipboard Format

All data copied to a system clipboard must conform to a format specification, known as a Clipboard Format. Each Clipboard Format is identified by a unique numeric format ID. This format ID is used to tag the data on the clipboard so that any application enumerating the contents of the clipboard is able to determine the format of the data without having to extract and analyze it.

A textual name can also be associated with each Clipboard Format. This format name is required when the Clipboard Format ID that identifies the format is not constant across computer systems. For example, the Clipboard Format name for "XYZ Drawing Image" may be identified by the Clipboard Format ID "0x1234" on one system, but "0x4321" on another.

Some formats may be implicit to an operating system and hence be represented by a hard-coded numeric Clipboard Format ID that is constant across all computers running that operating system. In this case, a Clipboard Format name is not required because the well-known Clipboard Format ID can always be used to uniquely identify the type.

It is the responsibility of the operating system to assign unique Clipboard Format IDs and to manage the associated Clipboard Format names of data that may be placed on the system clipboard.

Within the context of the Remote Desktop Protocol: Clipboard Virtual Channel Extension, the Palette (section 1.3.1.1.2) and Metafile (section 1.3.1.1.3) format types use the following hard-coded Clipboard Format IDs.

Format name	Format ID
Palette (section 1.3.1.1.2)	9
Metafile (section 1.3.1.1.3)	3

When referring to palette or metafile data within the context of the Remote Desktop Protocol: Clipboard Virtual Channel Extension **PDUs**, these Clipboard Format IDs are always used.

1.3.1.3 Monitoring Clipboard Updates

To be able to keep two independent system clipboards in sync, it is necessary to monitor both in order to detect when one of them has been updated with new Clipboard Formats (section 1.3.1.2). This monitoring can be carried out by polling the contents of both clipboards on a regular basis to determine if the contents have been updated. A more efficient mechanism than polling is to register for update notifications when the clipboard is changed; however, the ability to register for update notifications may not be available on all operating systems.

1.3.1.4 Delayed Rendering of Clipboard Data

Delayed rendering is a data transfer principle that makes it possible to keep two clipboards in sync while minimizing the network bandwidth required for communication. The underlying premise of delayed rendering is that data needs to be provided only when requested. When a copy operation takes place, the actual data associated with the Clipboard Format (section 1.3.1.2) is not copied onto the remote clipboard. Only the format ID that represents the Clipboard Format is placed on the clipboard. The data associated with the Clipboard Format is sent only if a paste operation is executed. The Remote Desktop Protocol: Clipboard Virtual Channel Extension requires that the system clipboard support delayed rendering of data to ensure the efficiency of clipboard synchronization.

1.3.2 Clipboard Redirection Virtual Channel Protocol

The Remote Desktop Protocol: Clipboard Virtual Channel Extension is divided into two distinct sequences:

- Initialization Sequence (section 1.3.2.1)
- Data Transfer Sequence (section 1.3.2.2)

During the Initialization Sequence, the connection is set up and capabilities and settings exchanged. The transfer of Clipboard Format IDs, names, and data takes place during the Data Transfer Sequence.

1.3.2.1 Initialization Sequence

The goal of the Initialization Sequence is to establish the **client** and the **server** capabilities, exchange settings, and synchronize the initial state of the client and server clipboards.

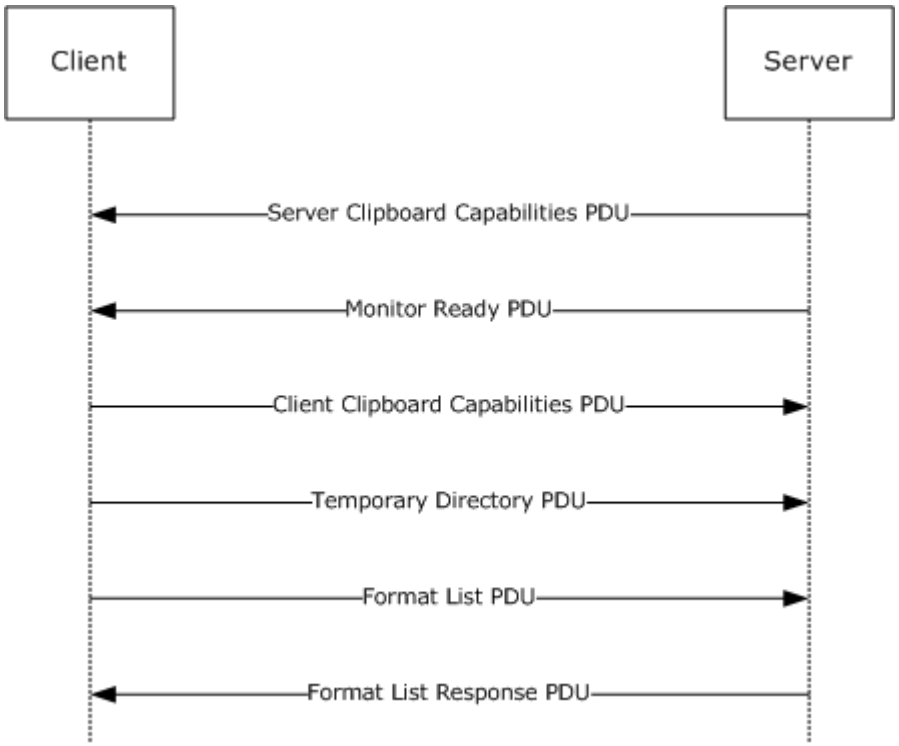


Figure 1: Clipboard Redirection Initialization Sequence

1. The server sends a Clipboard Capabilities PDU to the client to advertise the capabilities that it supports.
2. The server sends a Monitor Ready PDU to the client.
3. Upon receiving the Monitor Ready PDU, the client transmits its capabilities to the server by using a Clipboard Capabilities PDU.
4. The client sends the Temporary Directory PDU to inform the server of a temporary location on the client file system that can be used to deposit any temporary clipboard-related information. To make use of this location, the server must be able to access it directly. At this point, the client and the server capability negotiation is complete.
5. The final stage of the Initialization Sequence involves synchronizing the Clipboard Formats on the server clipboard with the client. This is accomplished by effectively mimicking a copy operation on the client by forcing it to send a Format List PDU.
6. The server responds with a Format List Response PDU.

1.3.2.2 Data Transfer Sequences

The goal of the Data Transfer Sequences is to perform a copy or paste operation.

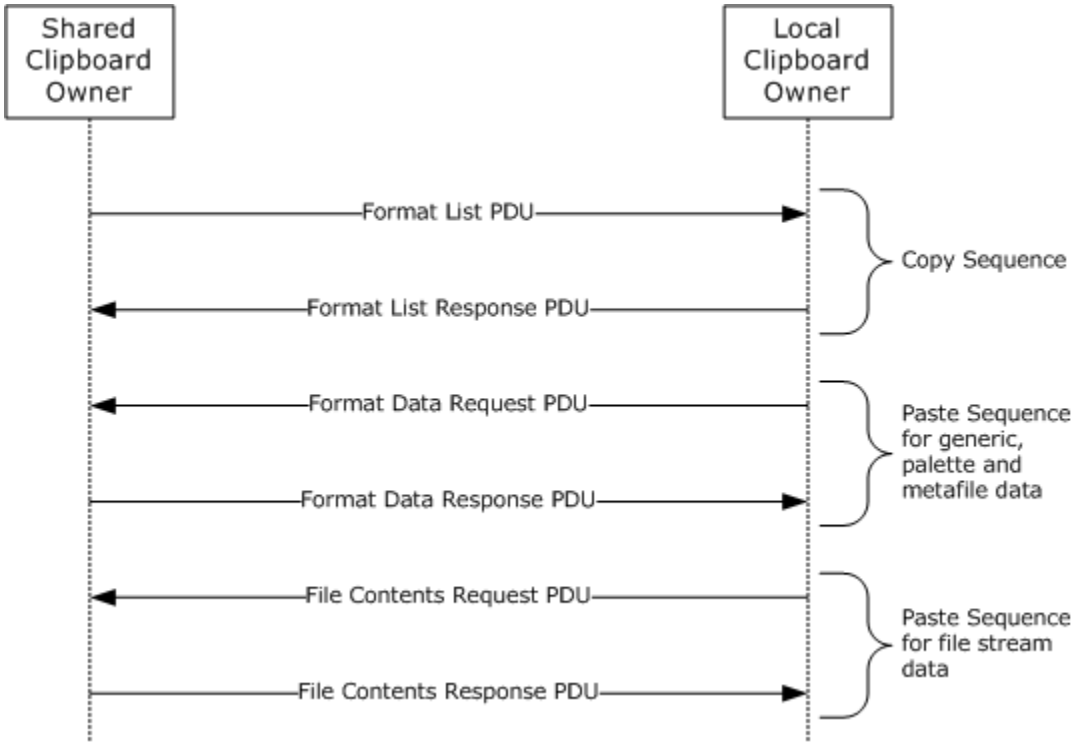


Figure 2: Data transfer using the shared clipboard

1. The sequence of messages for a copy operation is the same for all format types, as specified in section 1.3.2.2.1.

2. However, the messages exchanged to transfer file stream data during a paste operation differs from those used to transfer other format data, as specified in section 1.3.2.2.2.

1.3.2.2.1 Copy Sequence

The goal of the Copy Sequence is to synchronize the list of available formats across the client and the server clipboards.

The Format List PDU is sent by the Remote Desktop Protocol: Clipboard Virtual Channel Extension virtual channel endpoint on which the local clipboard has been updated. This endpoint is called the "Shared Clipboard Owner." The Format List PDU contains a list of the updated formats that are available on the clipboard of the Shared Owner. The recipient of the Format List PDU must update its local system clipboard with IDs of the Clipboard Formats that are available on the Shared Owner, and then send the Format List Response PDU in response. The sender of the Format List Response PDU is called the "Local Clipboard Owner." The format data is delay rendered, as specified in section 1.3.1.4.

1.3.2.2.2 Paste Sequence

The goal of the Paste Sequence is to transfer the data for a single format from the Shared Clipboard Owner to the Local Clipboard Owner.

The Format Data Request PDU is sent by the Local Clipboard Owner in response to a paste operation. This PDU contains the Clipboard Format ID, relative to the Shared Clipboard Owner's system, of the data that is required to complete the paste operation on the Local Clipboard Owner. The Shared Clipboard Owner retrieves the requested data from its local system clipboard and sends it to the Local Clipboard Owner in a Format Data Response PDU.

The File Contents Request PDU and File Contents Response PDU are used to implement the transfer of files. The Local Clipboard Owner must first request the list of files available from the clipboard. Upon receipt of this list, Local Clipboard Owner can request the contents of a file listed therein by sending a File Contents Request PDU to the Shared Clipboard Owner. The resultant file contents data is transmitted by the Shared Clipboard Owner to the Local Clipboard Owner by using a File Contents Response PDU.

1.3.2.3 Interacting with Local Clipboard and Applications

The following diagram and accompanying explanation illustrate how an application, the system clipboards, and the Remote Desktop Protocol: Clipboard Virtual Channel Extension endpoints interact during a copy and paste operation.

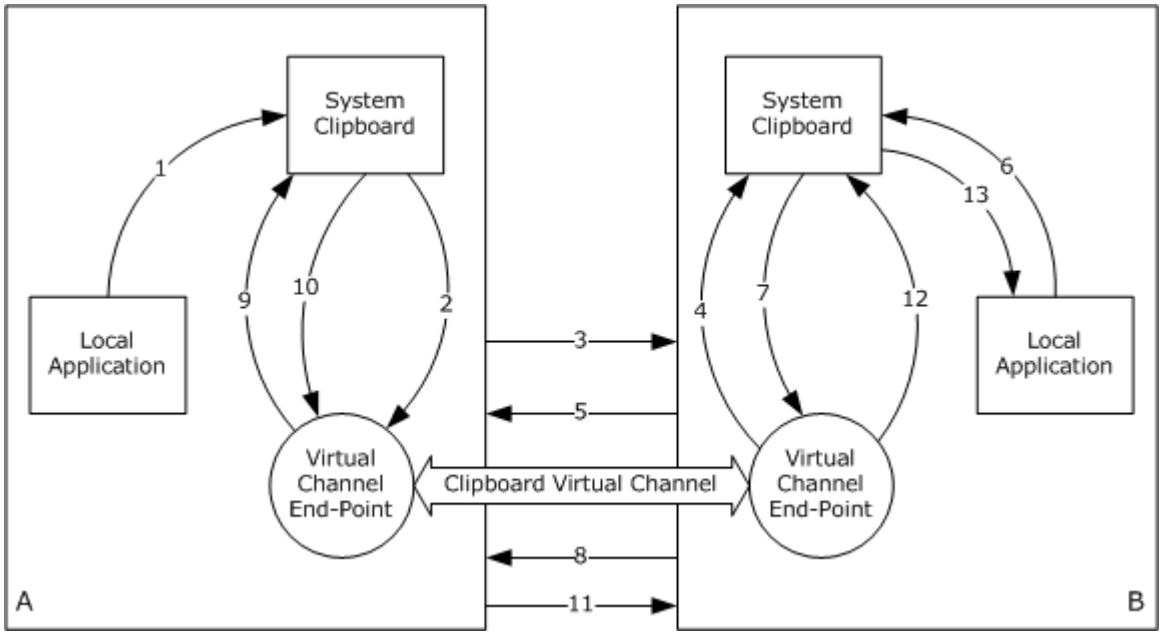


Figure 3: Interaction of applications, the system clipboards, and virtual channel endpoints

The copy phase is performed in steps 1 through 5 and the paste phase is performed in steps 6 through 13.

1. Local Application A copies data to System Clipboard A.

2. Virtual ChannelEndpoint A is notified of the clipboard update.

3. A list of formats on System Clipboard A are sent to Virtual ChannelEndpoint B in a Format List PDU.

4. Virtual ChannelEndpoint B updates System Clipboard B.

5. Virtual ChannelEndpoint B confirms success of System Clipboard B update in a Format List Response PDU.

6. Local Application B requests data from System Clipboard B.

7. System Clipboard B requests delay-rendered data from Virtual ChannelEndpoint B.

8. Virtual ChannelEndpoint B sends a request for data of requested type using a Format Data Request PDU or File Contents Response PDU.

9. Virtual ChannelEndpoint A requests data from System Clipboard A.

10. System Clipboard A returns data to Virtual ChannelEndpoint A.

11. Data is sent to Virtual ChannelEndpoint B using a Format List Response PDU or File Contents Response PDU.

12. Virtual ChannelEndpoint B supplies System Clipboard B with data.

13. System Clipboard B supplies Application B with data.

1.4 Relationship to Other Protocols

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

1.5 Prerequisites/Preconditions

The Remote Desktop Protocol: Clipboard 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: Clipboard Virtual Channel Extension occurs.

1.6 Applicability Statement

The Remote Desktop Protocol: Clipboard Virtual Channel Extension is designed to be run within the context of a Remote Desktop Protocol virtual channel established between a client and server. This protocol is applicable when bidirectional data transfer between the local client clipboard and the clipboard in the remote session (hosted on the server) is required.

1.7 Versioning and Capability Negotiation

The Remote Desktop Protocol: Clipboard Virtual Channel Extension is capability-based. The client and the server exchange capabilities during the protocol Initialization Sequence (section 1.3.2.1) by using the Clipboard Capabilities PDU. Capability sets are packaged in a combined capability set structure. This structure contains a count of the number of capability sets, followed by the contents of the individual capability sets.

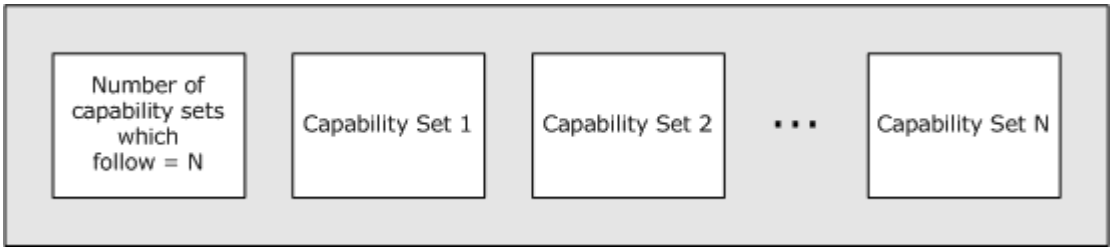


Figure 4: Combined capability set structure

After the capabilities have been received and stored, the client and the server do not send PDUs or data formats that cannot be processed by the **peer**.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

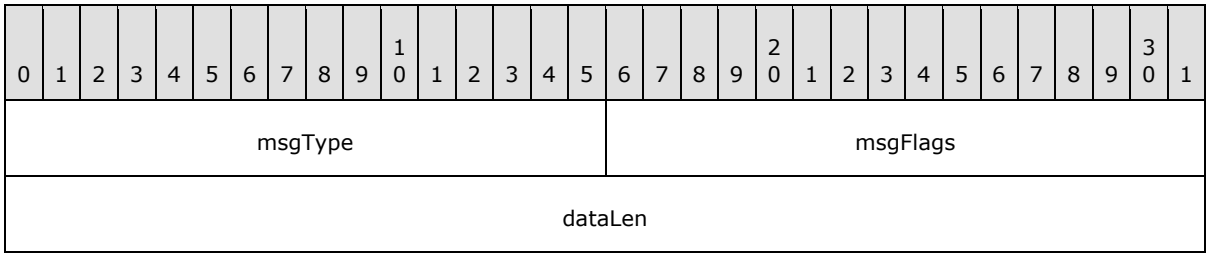
The Remote Desktop Protocol: Clipboard Virtual Channel Extension is designed to operate over a static virtual channel, as specified in [MS-RDPBCGR]. The virtual channel name is "CLIPRDR". The Remote Desktop Protocol layer manages the creation, setup, and transmission of data over the virtual channel.

2.2 Message Syntax

The following sections contain Remote Desktop Protocol: Desktop Composition Virtual Channel Extension message syntax.

2.2.1 Clipboard PDU Header (CLIPRDR_HEADER)

The CLIPRDR_HEADER structure is present in all clipboard PDUs. It is used to identify the PDU type, specify the length of the PDU, and convey message flags.



msgType (2 bytes): An unsigned 16-bit integer that specifies the type of the clipboard PDU that follows the **dataLen** field.

Value	Meaning
CB_MONITOR_READY 0x0001	Monitor Ready PDU
CB_FORMAT_LIST 0x0002	Format List PDU
CB_FORMAT_LIST_RESPONSE 0x0003	Format List Response PDU
CB_FORMAT_DATA_REQUEST 0x0004	Format Data Request PDU
CB_FORMAT_DATA_RESPONSE 0x0005	Format Data Response PDU
CB_TEMP_DIRECTORY 0x0006	Temporary Directory PDU
CB_CLIP_CAPS 0x0007	Clipboard Capabilities PDU
CB_FILECONTENTS_REQUEST 0x0008	File Contents Request PDU

Value	Meaning
CB_FILECONTENTS_RESPONSE 0x0009	File Contents Response PDU

msgFlags (2 bytes): An unsigned 16-bit integer that indicates message flags.

Value	Meaning
CB_RESPONSE_OK 0x0001	Used by the Format List Response PDU, Format Data Response PDU, and File Contents Response PDU to indicate that the associated request Format List PDU, Format Data Request PDU, and File Contents Request PDU were processed successfully.
CB_RESPONSE_FAIL 0x0002	Used by the Format List Response PDU, Format Data Response PDU, and File Contents Response PDU to indicate that the associated Format List PDU, Format Data Request PDU, and File Contents Request PDU were not processed successfully.
CB_ASCII_NAMES 0x0004	Used by the Short Format Name variant of the Format List Response PDU to indicate the format names are in ANSI.

dataLen (4 bytes): An unsigned 32-bit integer that specifies the size, in bytes, of the data which follows the Clipboard PDU Header.

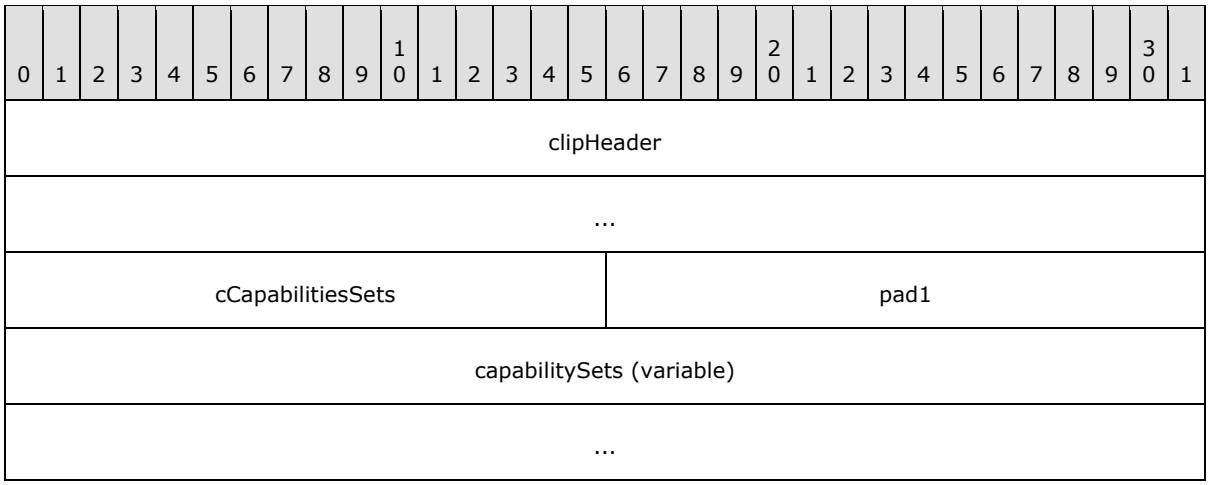
2.2.2 Initialization Sequence

The following sections contain Remote Desktop Protocol: Desktop Composition Virtual Channel Extension message syntax for the Initialization Sequence (section 1.3.2.1).

2.2.2.1 Clipboard Capabilities PDU (CLIPRDR_CAPS)

The Clipboard Capabilities PDU is an optional PDU used to exchange capability information. It is first sent from the server to the client. Upon receipt of the Monitor Ready PDU, the client sends the Clipboard Capabilities PDU to the server.

If this PDU is not sent by a Remote Desktop Protocol: Clipboard Virtual Channel Extension endpoint, it is assumed that the endpoint is using the default values for each capability field.



clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_CLIP_CAPS (0x0007).

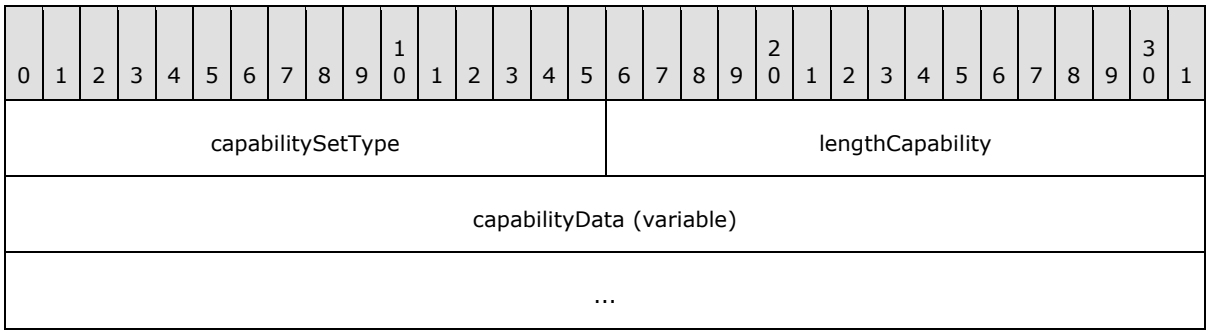
cCapabilitiesSets (2 bytes): An unsigned 16-bit integer that specifies the number of CLIPRDR_CAPS_SETs, present in the **capabilitySets** field.

pad1 (2 bytes): An unsigned 16-bit integer used for padding. Values in this field are ignored.

capabilitySets (variable): A variable-sized array of capability sets, each conforming in structure to the CLIPRDR_CAPS_SET.

2.2.2.1.1 Capability Set (CLIPRDR_CAPS_SET)

The CLIPRDR_CAPS_SET structure is used to wrap capability set data and to specify the type and size of this data exchanged between the client and the server. All capability sets conform to this basic structure.



capabilitySetType (2 bytes): An unsigned 16-bit integer used as a type identifier of the capability set.

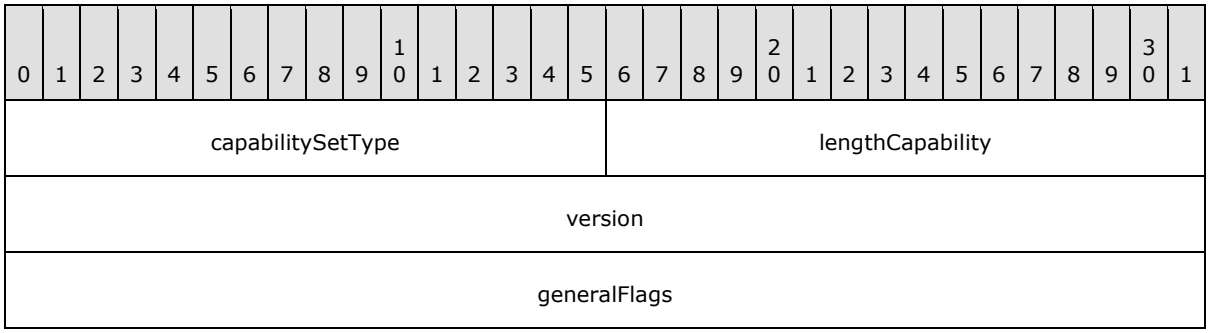
Value	Meaning
CB_CAPSTYPE_GENERAL 0x0001	General Capability Set

lengthCapability (2 bytes): An unsigned 16-bit integer that specifies the combined length, in bytes, of the **capabilitySetType**, **capabilityData** and **lengthCapability** fields.

capabilityData (variable): Capability set data specified by the type given in the **capabilitySetType** field. This field is a variable number of bytes.

2.2.2.1.1.1 General Capability Set (CLIPRDR_GENERAL_CAPABILITY)

The CLIPRDR_GENERAL_CAPABILITY structure is used to advertise general clipboard settings.



capabilitySetType (2 bytes): An unsigned 16-bit integer that specifies the type of the capability set. This field MUST be set to CB_CAPSTYPE_GENERAL (0x0001).

lengthCapability (2 bytes): An unsigned 16-bit integer that specifies the length, in bytes, of the **capabilitySetType**, **capability data** and **lengthCapability** fields.

version (4 bytes): An unsigned 32-bit integer that specifies the Remote Desktop Protocol: Clipboard Virtual Channel Extension version number. This field is for informational purposes and MUST not be used to make protocol capability decisions. The actual features supported are specified in the **generalFlags** field.

Value	Meaning
CB_CAPS_VERSION_1 0x00000001	Version 1
CB_CAPS_VERSION_2 0x00000002	Version 2

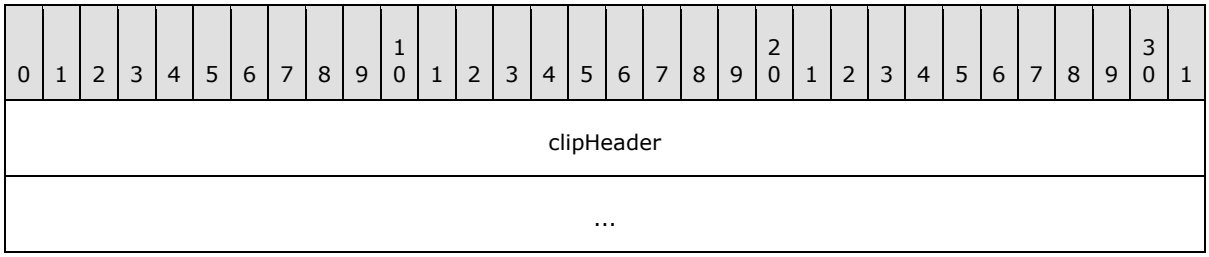
generalFlags (4 bytes): An unsigned 32-bit integer that specifies the general capability flags.

Value	Meaning
CB_USE_LONG_FORMAT_NAMES 0x00000002	The Long Format Name variant of the Format List PDU is used to exchange updated format names. If this flag is not set, the Short Format Name variant MUST be used.
CB_STREAM_FILECLIP_ENABLED 0x00000004	File copy and paste using stream-based operations with the File Contents Request PDU and File Contents Response PDU are supported.
CB_FILECLIP_NO_FILE_PATHS 0x00000008	Indicates that any description of files to copy and paste MUST NOT include the source path of the files.

If the General Capability Set is not present in the Clipboard Capabilities PDU, this field MUST be assumed to contain no flags.

2.2.2.2 Server Monitor Ready PDU (CLIPRDR_MONITOR_READY)

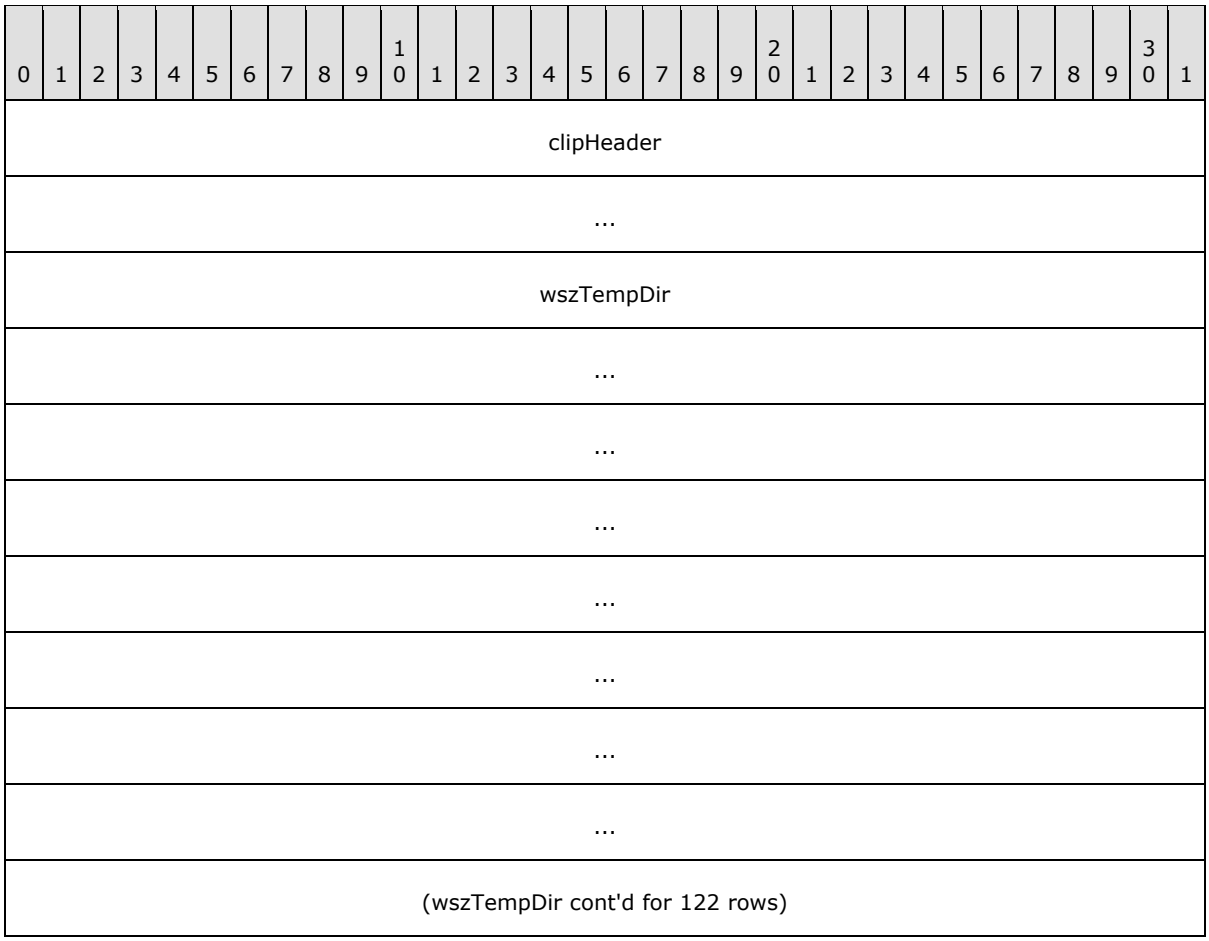
The Monitor Ready PDU is sent from the server to the client to indicate that the server is initialized and ready. This PDU is transmitted by the server after it has sent the Clipboard Capabilities PDU to the client.



clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_MONITOR_READY (0x0001).

2.2.2.3 Client Temporary Directory PDU (CLIPRDR_TEMP_DIRECTORY)

The Temporary Directory PDU is an optional PDU sent from the client to the server. This PDU informs the server of a temporary location on the client file system that can be used to store any temporary clipboard related information. The location MUST be accessible by the server to be useful. Section 3.1.1.3 specifies how direct file access impacts file copy and paste. This PDU is sent by the client after receiving the Monitor Ready PDU.



clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_TEMP_DIRECTORY (0x0006).

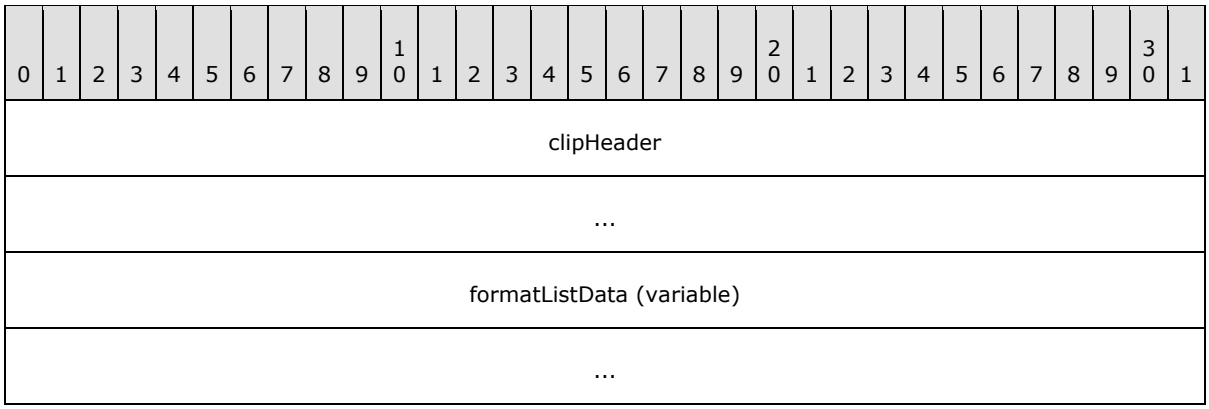
wszTempDir (520 bytes): A 520-byte block that contains a null-terminated string that represents the directory on the client that can be used to store any temporary clipboard related information. The supplied path should be absolute and relative to the local client system, for example, "c:\temp\clipdata". Any space not used in this field SHOULD be filled with null characters.

2.2.3 Copy Sequence

The following sections contain Remote Desktop Protocol: Desktop Composition Virtual Channel Extension message syntax for the Copy Sequence (section 1.3.2.2.1).

2.2.3.1 Format List PDU (CLIPRDR_FORMAT_LIST)

The Format List PDU is sent by either the client or the server when its local system clipboard is updated with new clipboard data. This PDU contains the Clipboard Format ID and name pairs of the new Clipboard Formats on the clipboard.

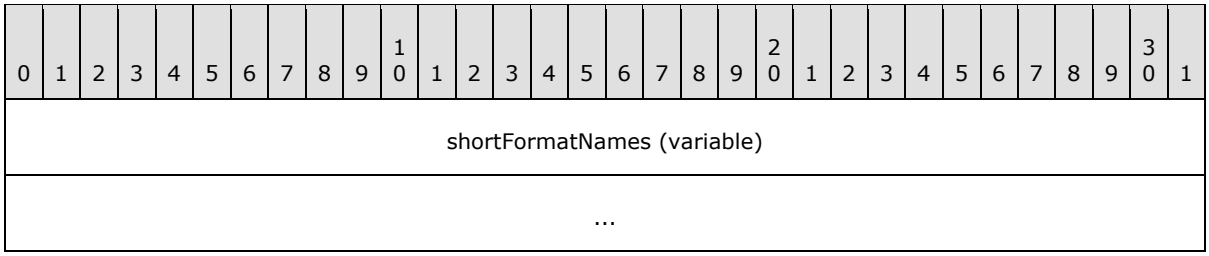


clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FORMAT_LIST (0x0002).

formatListData (variable): An array consisting solely of either Short Format Names or Long Format Names. Each array holds a list of the Clipboard Format ID and name pairs available on the local system clipboard of the sender. The Clipboard Format names are in either ANSI or Unicode format, depending on the presence of the CB_USE_LONG_FORMAT_NAMES (0x00000002) flag in the General Capability Set.

2.2.3.1.1 Short Format Names (CLIPRDR_SHORT_FORMAT_NAMES)

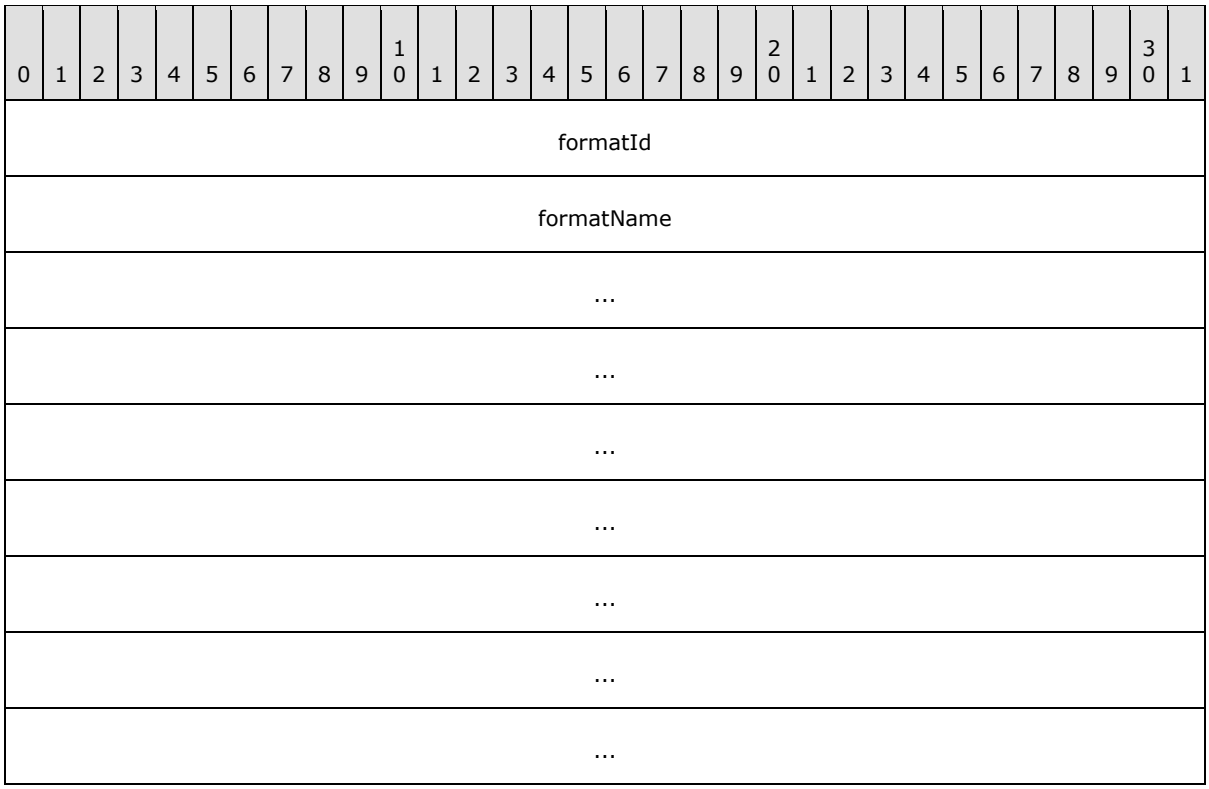
The CLIPRDR_SHORT_FORMAT_NAMES structure holds a collection of CLIPRDR_SHORT_FORMAT_NAME structures.



shortFormatNames (variable): An array of CLIPRDR_SHORT_FORMAT_NAME structures.

2.2.3.1.1.1 Short Format Name (CLIPRDR_SHORT_FORMAT_NAME)

The CLIPRDR_SHORT_FORMAT_NAME structure holds a Clipboard Format ID and Clipboard Format name pair.

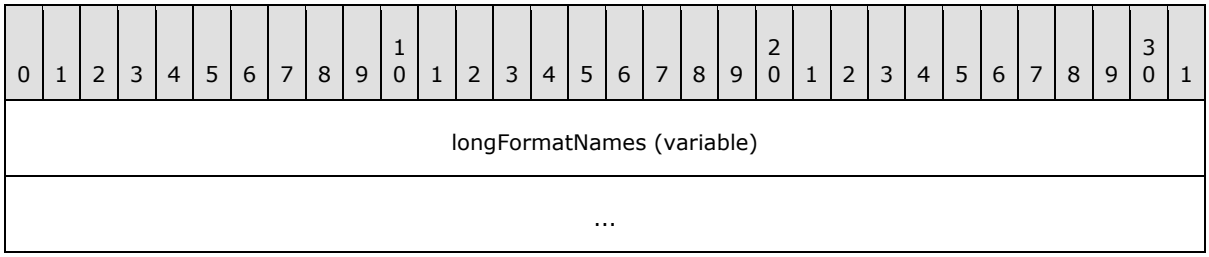


formatId (4 bytes): An unsigned 32-bit integer specifying the Clipboard Format ID.

formatName (32 bytes): A 32-byte block containing the null-terminated name assigned to the Clipboard Format (32 ANSI characters or 16 Unicode characters). If the name does not fit, it MUST be truncated. Not all Clipboard Formats have a name, and in that case the **formatName** field MUST contain only zeros.

2.2.3.1.2 Long Format Names (CLIPRDR_LONG_FORMAT_NAMES)

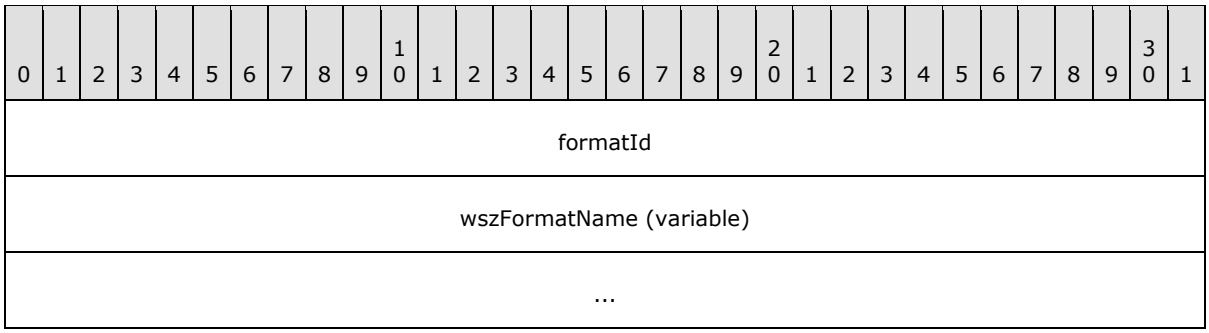
The CLIPRDR_LONG_FORMAT_NAMES structure holds a collection of CLIPRDR_LONG_FORMAT_NAME structures.



longFormatNames (variable): An array of CLIPRDR_LONG_FORMAT_NAME structures.

2.2.3.1.2.1 Long Format Name (CLIPRDR_LONG_FORMAT_NAME)

The CLIPRDR_LONG_FORMAT_NAME structure holds a Clipboard Format ID and a Clipboard Format name pair.

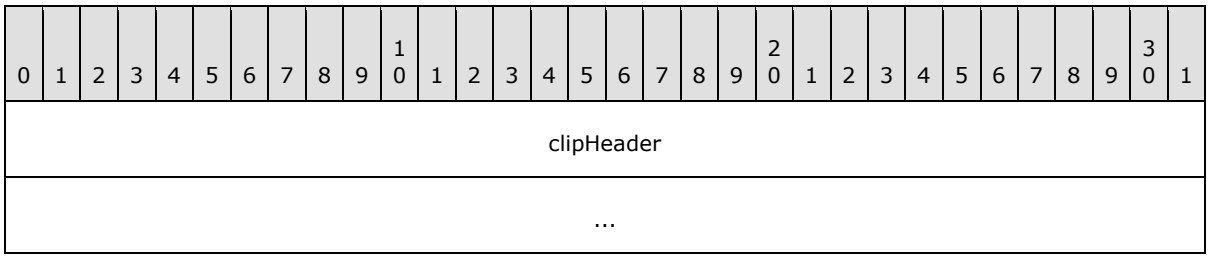


formatId (4 bytes): An unsigned 32-bit integer that specifies the Clipboard Format ID.

wszFormatName (variable): A variable length null-terminated Unicode string name that contains the Clipboard Format name. Not all Clipboard Formats have a name; in such cases, the **formatName** field MUST consist of a single Unicode null character.

2.2.3.2 Format List Response PDU (FORMAT_LIST_RESPONSE)

The Format List Response PDU is sent as a reply to the Format List PDU. It is used to indicate whether processing of the Format List PDU was successful.



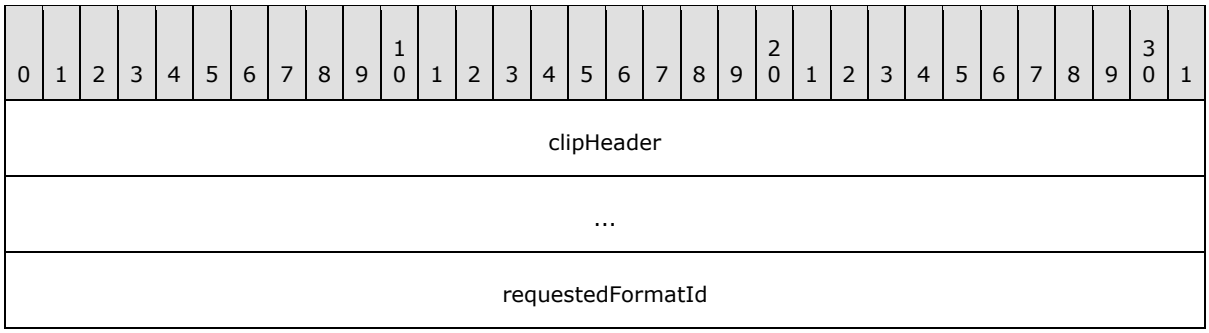
clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FORMAT_LIST_RESPONSE (0x0003). The CB_RESPONSE_OK (0x0001) or CB_RESPONSE_FAIL (0x0002) flag MUST be set in the **msgFlags** field of the Clipboard PDU Header.

2.2.4 Paste Sequence

The following sections contain Remote Desktop Protocol: Desktop Composition Virtual Channel Extension message syntax for the Paste Sequence (section 1.3.2.2.2).

2.2.4.1 Format Data Request PDU (CLIPRDR_FORMAT_DATA_REQUEST)

The Format Data Request PDU is sent by the recipient of the Format List PDU. It is used to request the data for one of the formats that was listed in the Format List PDU.

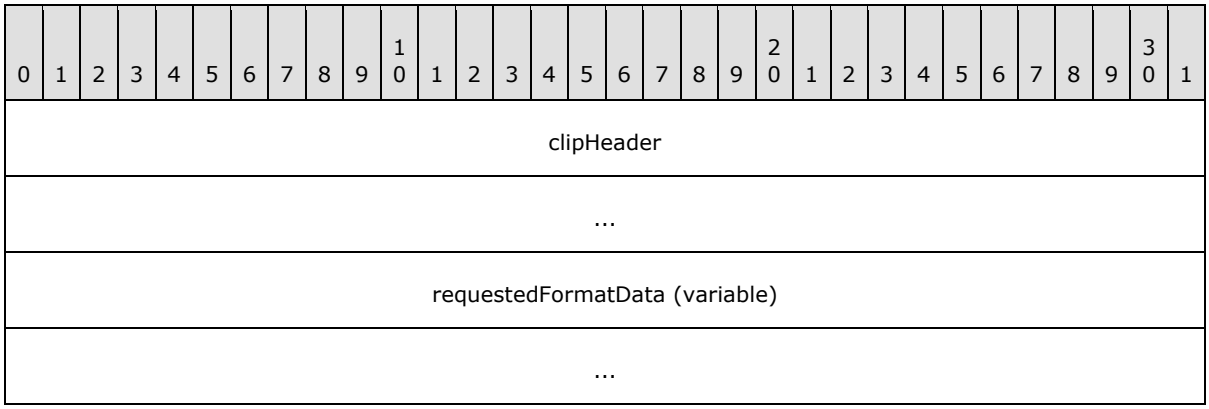


clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FORMAT_DATA_REQUEST (0x0004).

requestedFormatId (4 bytes): An unsigned 32-bit integer that specifies the Clipboard Format ID of the clipboard data. The Clipboard Format ID MUST be one listed previously in the Format List PDU.

2.2.4.2 Format Data Response PDU (CLIPRDR_FORMAT_DATA_RESPONSE)

The Format Data Response PDU is sent as a reply to the Format Data Request PDU. It is used to indicate whether processing of the Format Data Request PDU was successful. If the processing was successful, the Format Data Response PDU includes the contents of the requested clipboard data.

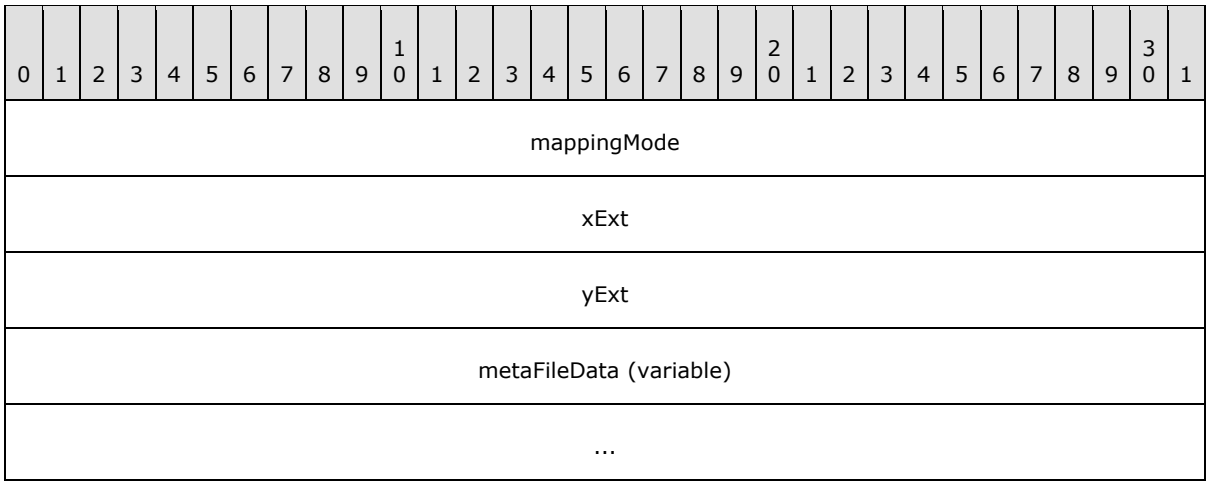


clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FORMAT_DATA_RESPONSE (0x0005). The CB_RESPONSE_OK (0x0001) or CB_RESPONSE_FAIL (0x0002) flag MUST be set in the **msgFlags** field of the Clipboard PDU Header structure.

requestedFormatData (variable): Variable length clipboard format data. The contents of this field will be one of the following types: generic, type-specific data, Packed Metafile Payload, or Packed Palette Payload.

2.2.4.2.1 Packed Metafile Payload (CLIPRDR_MFPICT)

The CLIPRDR_MFPICT structure is used to transfer a graphics metafile. The metafile format is specified in [MS-WMF] and described in [MSDN-META].



mappingMode (4 bytes): An unsigned 32-bit integer specifying the mapping mode in which the picture is drawn.

Value	Meaning
MM_TEXT 0x00000001	Each logical unit is mapped to one device pixel. Positive x is to the right; positive y is down.
MM_LOMETRIC 0x00000002	Each logical unit is mapped to 0.1 millimeter. Positive x is to the right; positive y is up.
MM_HIMETRIC 0x00000003	Each logical unit is mapped to 0.01 millimeter. Positive x is to the right; positive y is up.
MM_LOENGLISH 0x00000004	Each logical unit is mapped to 0.01 inch. Positive x is to the right; positive y is up.
MM_HIENGLISH 0x00000005	Each logical unit is mapped to 0.001 inch. Positive x is to the right; positive y is up.
MM_TWIPS 0x00000006	Each logical unit is mapped to 1/20 of a printer's point (1/1440 of an inch), also called a twip. Positive x is to the right; positive y is up.
MM_ISOTROPIC 0x00000007	Logical units are mapped to arbitrary units with equally scaled axes; one unit along the x-axis is equal to one unit along the y-axis.
MM_ANISOTROPIC 0x00000008	Logical units are mapped to arbitrary units with arbitrarily scaled axes.

For MM_ISOTROPIC and MM_ANISOTROPIC modes, which can be scaled, the **xExt** and **yExt** fields contain an optional suggested size in MM_HIMETRIC units. For MM_ANISOTROPIC pictures, **xExt** and **yExt** can be zero when no suggested size is given. For MM_ISOTROPIC pictures, an aspect ratio MUST be supplied even when no suggested size is given. If a suggested size is given, the aspect ratio is implied by the size. To give an aspect ratio without implying a suggested size, the **xExt** and **yExt** fields are set to negative values whose ratio is the appropriate aspect ratio. The magnitude of the negative **xExt** and **yExt** values is ignored; only the ratio is used.

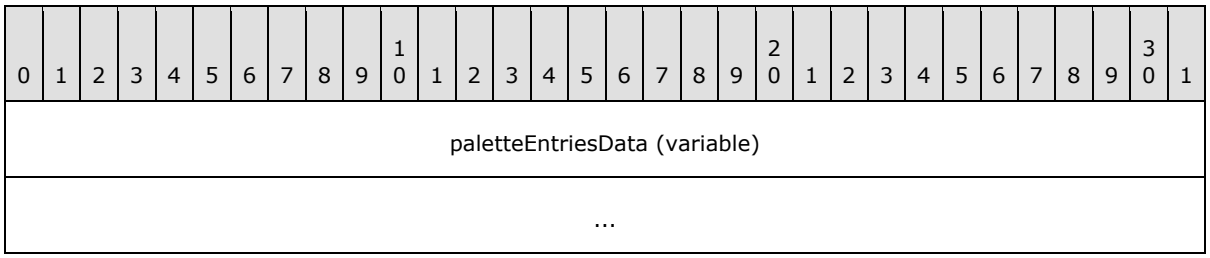
xExt (4 bytes): An unsigned 32-bit integer that specifies the width of the rectangle within which the picture is drawn, except in the MM_ISOTROPIC and MM_ANISOTROPIC modes. The coordinates are in units that correspond to the mapping mode.

yExt (4 bytes): An unsigned 32-bit integer that specifies the height of the rectangle within which the picture is drawn, except in the MM_ISOTROPIC and MM_ANISOTROPIC modes. The coordinates are in units that correspond to the mapping mode.

metaFileData (variable): The variable sized contents of the metafile.

2.2.4.2.2 Packed Palette Payload (CLIPRDR_PALETTE)

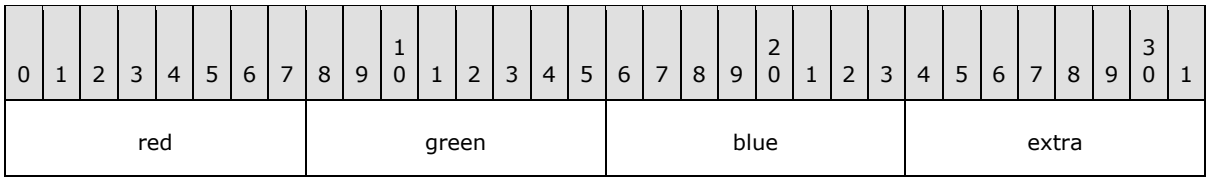
The CLIPRDR_PALETTE structure is used to transfer palette format data.



paletteEntriesData (variable): A variable sized array of PALETTEENTRY structures.

2.2.4.2.2.1 Palette Entry (PALETTEENTRY)

The PALETTEENTRY structure contains a single palette entry that specifies the red, green, and blue components for a given color index, in addition to any application-specific information related to the entry.



red (1 byte): An unsigned 8-bit red color component.

green (1 byte): An unsigned 8-bit green color component.

blue (1 byte): An unsigned 8-bit blue color component.

extra (1 byte): This field MAY be used to convey application-specific palette information. Some applications use this field to specify how the palette entry should be used.

2.2.4.3 File Contents Request PDU (CLIPRDR_FILECONTENTS_REQUEST)

The File Contents Request PDU is sent by the recipient of the Format List PDU. It is used to request either the size of a remote file copied to the clipboard or a portion of the data in the file.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
clipHeader																															
...																															
streamId																															
lindex																															
dwFlags																															
nPositionLow																															
nPositionHigh																															
cbRequested																															

clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FILECONTENTS_REQUEST (0x0008).

streamId (4 bytes): An unsigned 32-bit format ID used to associate the File Contents Request PDU with the corresponding File Contents Response PDU. The File Contents Response PDU is sent as a reply and contains an identical value in the **streamId** field.

lindex (4 bytes): A signed 32-bit integer that specifies the numeric ID of the remote file that is the target of the File Contents Request PDU. This field is used as an index that identifies a particular file in a File List. This File List SHOULD have been obtained as clipboard data in a prior Format Data Request PDU and Format Data Response PDU exchange.

dwFlags (4 bytes): An unsigned 32-bit integer that specifies the type of operation to be performed by the recipient.

Value	Meaning
FILECONTENTS_SIZE 0x00000001	A request for the size of the file identified by the **lindex** field. The size MUST be returned as a 64-bit unsigned integer. The **cbRequested** field MUST be set to 8 bytes and both the **nPositionLow** and **nPositionHigh** fields MUST be set to 0.
FILECONTENTS_RANGE 0x00000002	A request for the data present in the file identified by the **lindex** field. The data to be retrieved is extracted starting from the offset given by the **nPositionLow** and **nPositionHigh** fields. The maximum number of bytes to extract is specified by the **cbRequested** field.

The FILECONTENTS_SIZE and FILECONTENTS_RANGE flags cannot be set at the same time.

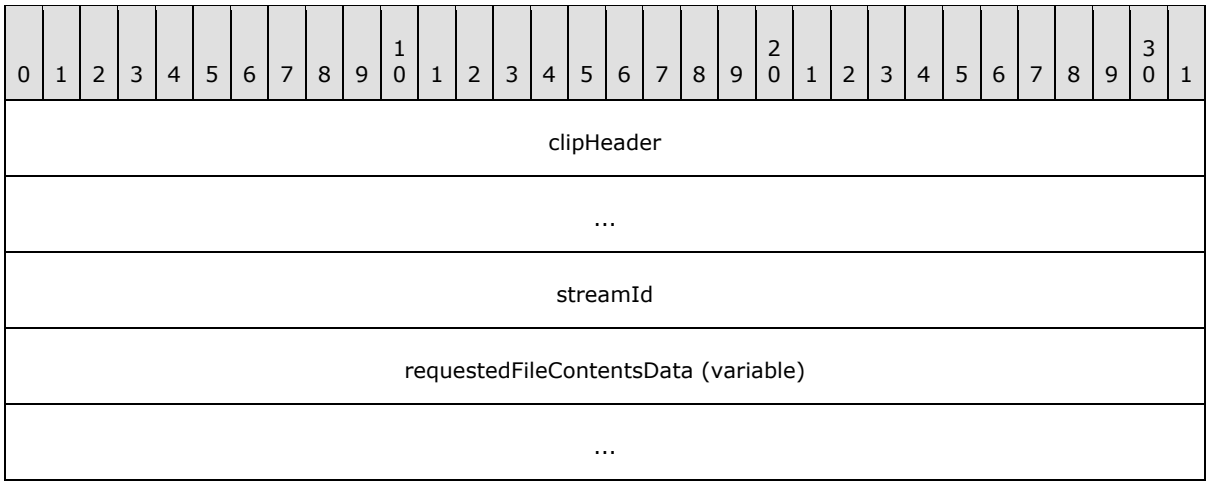
nPositionLow (4 bytes): An unsigned 32-bit integer that specifies the low bytes of the offset into the remote file, identified by the **lindex** field, from where the data needs to be extracted to satisfy a FILECONTENTS_RANGE operation.

nPositionHigh (4 bytes): An unsigned 32-bit integer that specifies the high bytes of the offset into the remote file, identified by the **lindex** field, from where the data needs to be extracted to satisfy a FILECONTENTS_RANGE operation. This field is currently not used because offsets greater than 4,294,967,295 bytes are not supported, and MUST be set to zero.

cbRequested (4 bytes): An unsigned 32-bit integer that specifies the size, in bytes, of the data to retrieve. For a FILECONTENTS_SIZE operation, this field MUST be set to 8. In the case of a FILECONTENTS_RANGE operation, this field contains the maximum number of bytes to read from the remote file.

2.2.4.4 File Contents Response PDU (CLIPRDR_FILECONTENTS_RESPONSE)

The File Contents Response PDU is sent as a reply to the File Contents Request PDU. It is used to indicate whether processing of the File Contents Request PDU was successful. If the processing was successful, the File Contents Response PDU includes either a file size or extracted file data, based on the operation requested in the corresponding File Contents Request PDU.



clipHeader (8 bytes): A Clipboard PDU Header. The **msgType** field of the Clipboard PDU Header MUST be set to CB_FILECONTENTS_RESPONSE (0x0009).

streamId (4 bytes): An unsigned 32-bit numeric ID used to associate the File Contents Response PDU with the corresponding File Contents Request PDU . The File Contents Request PDU that triggered the response MUST contain an identical value in the **streamId** field.

requestedFileContentsData (variable): This field contains a variable number of bytes. If the response is to a FILECONTENTS_SIZE (0x00000001) operation, the **requestedFileContentsData** field holds a 64-bit unsigned integer containing the size of the file. In the case of a FILECONTENTS_RANGE (0x00000002) operation, the **requestedFileContentsData** field contains a byte-stream of data extracted from the file.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

3.1.1.1 Clipboard Format ID Map

The Clipboard Format ID Map is used to translate local Clipboard Format IDs to remote Clipboard Format IDs.

For example, assume that on System A, the Clipboard Format with a format name of "Format X" maps to the format ID 0x1111; and on System B, the format ID corresponding to the format name "Format X" might be 0x2222.

- On System A, the format map entry for "Format X" would appear as follows:

  Format X: Local ID 0x1111 maps to Remote ID 0x2222

- On System B, the format map entry for "Format X" would appear as follows:

  Format X: Local ID 0x2222 maps to Remote ID 0x1111

The Clipboard Format ID Map is cleared and initialized whenever a Format List PDU (section 3.1.5.2.2) is processed.

3.1.1.2 File List

When a collection of files is copied to the system clipboard, accompanying metadata containing the list of files, called the "file list," is also placed onto the clipboard using a generic, operating system-defined format. This list contains information about each file on the clipboard, such as the file name, size, and access permissions. Applications can examine the file list to enumerate the list of files available on the system clipboard.

When a paste operation is initiated to obtain the contents of a file on the clipboard, the index of the file in the file list, along with a description of the file chunks required, is sent to the system clipboard. The system clipboard responds by returning the file contents data that was requested.

The usage of the file list is best illustrated with a practical example.

1. Assume that a user copies two files to the clipboard so that the associated file list on the clipboard appears as follows (notice that the exact location of the files is not specified in the file list):

   1. temp\file1.txt (20 bytes)

   2. temp\file2.txt (10 bytes)

2. Next, assume that the user decides to paste the first 15 bytes of file1.txt into a target application that can accept file data. In this case, the target application examines the file list on the clipboard and issues a request for the first 15 bytes of the file in the file list at Index 1 (the system clipboard MUST be contacted because the exact location of the file on the local file system is not necessarily advertised in the file list). The system clipboard responds with the appropriate data.<1>

3.1.1.3 Direct File Access

If the client or server has direct access to the local file system of the peer, a File List that uses absolute paths (as opposed to relative paths) can be modified in transmit to point directly to the source files, thus bypassing having to contact the system clipboard for the actual file contents.

For example:

1. Assume that the server is able to view the client files via a network share such that the client file c:\temp\file1.txt is accessible as: \\client-files\c\temp\file1.txt.

2. Next, assume that a File List (which contains the files c:\temp\file1.txt and c:\temp\file2.txt) is copied to the client's local clipboard. Then, when transmitting the data in the File List to the server, the client could modify the File List contents as follows:

    1. \\client-files\c\temp\file1.txt (20 bytes)

    2. \\client-files\c\temp\file2.txt (10 bytes)

Thus the server merely needs to retrieve the File List and directly access any of the files therein via the mapped network share to perform a file paste operation.

3.1.2 Timers

None.

3.1.3 Initialization

The static virtual channel MUST be established, using the parameters specified in section 2.1, before protocol operation can commence.

3.1.4 Higher-Layer Triggered Events

This section contains details about the higher-layer triggered events.

3.1.4.1 Local Clipboard Update

When the local system clipboard is updated, the client or the server associated with the clipboard MUST send the Format List PDU (section 3.1.5.2.1) to ensure that the formats available on the remote clipboards are kept in sync.

3.1.4.2 Local Paste Operation

When a local application requests data from the clipboard, and that data resides on the clipboard of a remote computer, the local computer MUST send the Format Data Request PDU or the File Contents Request PDU, depending on the type of data requested.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing a Clipboard PDU

All clipboard PDUs are prefaced by the Clipboard PDU Header structure.

When processing a clipboard PDU, the **msgType** field in the header MUST first be examined to determine if the PDU is within the subset of expected messages. If the PDU is not expected, it SHOULD be ignored.

After determining that the PDU is in the correct sequence, the **dataLen** field MUST be examined to make sure that it is consistent with the amount of data read from the "CLIPRDR" static virtual channel. If this is not the case, the connection SHOULD be dropped.

3.1.5.2 Copy Sequence

3.1.5.2.1 Sending a Format List PDU

The fields of the Format List PDU are specified in section 2.2.3.1.

To construct the Format List PDU, the sender MUST enumerate all of the formats that are currently available from the local system clipboard, and for each format.

- Obtain the format ID associated with the Clipboard Format.
- Determine if the Clipboard Format has a corresponding format name.

The Format List PDU MUST be populated with this data. The usage of the Short Format Names structure or Long Format Names structure MUST be based on the capabilities specified by the General Capability Set. If short format names in ANSI format are being used, the CB_ASCII_NAMES flag MUST be set in the **msgFlags** field of the **clipHeader** field. Unicode names MUST always be used with the long format names.

3.1.5.2.2 Processing a Format List PDU

The fields of the Format List PDU are specified in section 2.2.3.1.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the format types present in the PDU can be extracted, processed, and placed on the local system clipboard.

For each Clipboard Format listed in the Format List PDU, the recipient of the PDU MUST do the following:

- Store the mapping of the remote Clipboard Format ID to the local Clipboard Format ID Map. The remote format ID is specified in the **formatId** field of the Short Format Name structure and the Long Format Name structure.
- Update the local system clipboard by registering the local Clipboard Format ID as an available format for transfer. The system clipboard MUST support delayed rendering (as specified in section 1.3.1.4) for this step to be possible.

If the PDU was processed successfully and the local system clipboard was updated with all the received Clipboard Formats, the recipient MUST send a Format List Response PDU indicating success, as specified in section 3.1.5.2.3. If the PDU could not be processed, or the local clipboard could not be updated, a Format List Response PDU indicating failure MUST be sent, as specified in section 3.1.5.2.3.

3.1.5.2.3 Sending a Format List Response PDU

The fields of the Format List Response PDU are specified in section 2.2.3.2.

This PDU is being sent to indicate the success or failure of processing the Format List PDU, as specified in section 3.1.5.2.2. On success, the **msgFlags** field of the **clipHeader** field MUST contain the CB_RESPONSE_OK flag. On failure, it MUST contain the CB_RESPONSE_FAIL flag.

3.1.5.2.4 Processing a Format List Response PDU

The fields of the Format List Response PDU are specified in section 2.2.3.2.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the response code can be extracted from the **msgFlags** field of the **clipHeader** field. If the response code indicates that the associated Format List PDU was successful, the recipient can expect to receive a Format Data Request PDU with a request for format data.

3.1.5.3 Paste Sequence

3.1.5.3.1 Sending a Format Data Request PDU

The fields of the Format Data Request PDU are specified in section 2.2.4.1.

The Clipboard Format ID of the clipboard data MUST be specified in the **requestedFormatId** field. The Clipboard Format ID Map MUST be used to map the local Clipboard Format ID of the requested clipboard data to the equivalent value on the remote system.

3.1.5.3.2 Processing a Format Data Request PDU

The fields of the Format Data Request PDU are specified in section 2.2.4.1.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the requested Clipboard Format ID MUST be extracted from the PDU and the clipboard data retrieved from the local clipboard. The retrieved clipboard data MUST then be encoded appropriately, depending on the type.

- Metafile data MUST be encoded using the Packed Metafile Payload structure.
- Palette data MUST be encoded using the Packed Palette Payload structure.
- If the clipboard data is not a metafile or palette, it is left unchanged.

The clipboard data MUST then be sent to the remote computer by using a Format Data Response PDU. If the request cannot be completed, a Format Data Response PDU containing the CB_RESPONSE_FAIL (0x0002) flag MUST be sent to the remote computer. On success, the CB_RESPONSE_OK (0x0001) flag MUST be specified.

3.1.5.3.3 Sending a Format Data Response PDU

The fields of the Format Data Response PDU are specified in section 2.2.4.2.

If there is clipboard data to send, it MUST be copied into the **requestedFormatData** field and the **clipHeader** field MUST contain the CB_RESPONSE_OK (0x0001) flag. If the requested clipboard data could not be retrieved, the **clipHeader** field MUST contain the CB_RESPONSE_FAIL (0x0002) flag and the **requestedFormatData** field MUST contain no data(zero-length).

3.1.5.3.4 Processing a Format Data Response PDU

The fields of the Format Data Response PDU are specified in section 2.2.4.2.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the attached data can be extracted if the **msgFlags** indicate success.

- Metafile data MUST be decoded using the Packed Metafile Payload structure.

- Palette data MUST be decoded using the Packed Palette Payload structure.
- If the data is not a metafile or palette, it does not need to be decoded.

The processed clipboard data MUST be returned to the clipboard to satisfy the paste operation.

3.1.5.3.5 Sending a File Contents Request PDU

The fields of the File Contents Request PDU are specified in section 2.2.4.3.

Prior to requesting any file contents data, the sender of the File Contents Request PDU MUST determine the appropriate index (specified in the **lindex** field) to identify the file on the remote clipboard. This index can be obtained through a File List, which is transferred via the Format Data Request PDU and the Format Data Response PDU, or through some other out-of-band mechanism.

Knowledge of the size of a file on the remote clipboard, identified by a particular index value, is a prerequisite to requesting the actual contents of the file by using the File Contents Request PDU. The size, in bytes, of a particular file can be obtained from the File List associated with the file, or the File Contents Request PDU can be used to request the size by setting the FILECONTENTS_SIZE (0x00000001) flag on the **dwFlags** field and populating the PDU fields.

If the size of a file on the remote clipboard is known, the File Contents Request PDU can be used to obtain the file contents at a particular offset by setting the FILECONTENTS_RANGE (0x00000002) flag on the **dwFlags** field and populating the PDU fields. The specified range MUST be within the bounds of the file size.

3.1.5.3.6 Processing a File Contents Request PDU

The fields of the Format Data Response PDU are specified in section 2.2.4.2.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the data requested for the file (specified by the **lindex** field) MUST be returned to the sender.

The recipient of this PDU MUST perform a lookup using the **lindex** field to find the file that is the target of the request. The lookup most likely involves accessing the File List with which the current transaction is associated and using it to obtain the file information and contents.

After the file information has been acquired, the size or contents MUST be sent to the Remote Desktop Protocol: Clipboard Virtual Channel Extension endpoint by using the File Contents Response PDU, and sent as specified in section 3.1.5.3.7. If the request cannot be satisfied, a File Contents Response PDU that contains the CB_RESPONSE_FAIL (0x0002) flag MUST be sent; otherwise, the CB_RESPONSE_OK (0x0001) flag MUST be specified.

3.1.5.3.7 Sending a File Contents Response PDU

The fields of the File Contents Response PDU are specified in section 2.2.4.4.

If there is response data to send, the data MUST be copied into the **requestedFileContentsData** field and the **clipHeader** field MUST contain the CB_RESPONSE_OK (0x0001) flag. If the requested clipboard data could not be retrieved, the **clipHeader** field MUST contain the CB_RESPONSE_FAIL (0x0002) flag and the **requestedFileContentsData** field MUST be left empty.

3.1.5.3.8 Processing a File Contents Response PDU

The fields of the File Contents Response PDU are specified in section 2.2.4.4.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid and contains file contents data, the bytes MUST be extracted from the PDU and returned to the clipboard to satisfy the paste operation. If the PDU contains the size of the file, it MUST be read from the **requestedFileContentsData** field as a 64-bit **little-endian** unsigned integer.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

3.2.1.1 Server Capabilities

The Server Capabilities store contains capability data received from the server in the Clipboard Capabilities PDU. The client MUST ensure that it does not violate any of the server capabilities when sending data.

If a Clipboard Capabilities PDU is not received from the server, it MUST be assumed that the server is using the default capability values as specified in section 2.2.2.1.

3.2.2 Timers

None.

3.2.3 Initialization

The static virtual channel MUST be established, using the parameters as specified in section 2.1, before protocol operation can commence.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Initialization Sequence

3.2.5.1.1 Processing a Server Clipboard Capabilities PDU

The fields of the Clipboard Capabilities PDU are as specified in section 2.2.2.1.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the capability data MUST be extracted and stored in the Server Capabilities store.

3.2.5.1.2 Processing a Monitor Ready PDU

The fields of the Monitor Ready PDU are as specified in section 2.2.2.2.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the client SHOULD do the following:

1. Send a Clipboard Capabilities PDU (as specified in section 3.2.5.1.3) to the server if it received the capabilities from the server.

2. Send a Temporary Directory PDU (as specified in section 3.2.5.1.4) to the server if it is necessary to inform the server of a location on the local client file system that MAY be used to store temporary clipboard related data.

After possibly sending the Clipboard Capabilities PDU and Temporary Directory PDU, the client MUST send a Format List PDU to the server, as specified in section 3.1.5.2.1. This ensures that the peer system clipboards are in sync.

3.2.5.1.3 Sending a Client Clipboard Capabilities PDU

The fields of the Clipboard Capabilities PDU are as specified in section 2.2.2.1.

The client MUST perform a merge with the server capabilities in the Server Capabilities store before populating the Clipboard Capabilities PDU so that it sends equivalent or lesser capabilities.

3.2.5.1.4 Sending a Temporary Directory PDU

The fields of the Temporary Directory PDU are specified in section 2.2.2.3.

Prior to sending this PDU, the client MUST ensure that the location specified is accessible to the server.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Server Details

3.3.1 Abstract Data Model

3.3.1.1 Client Capabilities

The Client Capabilities store contains capability data received from the client in the Clipboard Capabilities PDU. The server MUST ensure that it does not violate any of the client capabilities when sending data.

If a Clipboard Capabilities PDU is not received from the client, it MUST be assumed that the server is using the default capability values as specified in section 2.2.2.1.

3.3.1.2 Client Temporary Directory

The Client Temporary Directory store holds the path to a temporary location on the client file system that may be used to deposit any temporary clipboard related information. This information is received when processing the Temporary Directory PDU, as specified in section 3.3.5.1.4. If the

Temporary Directory PDU is not received from the client, the server MUST NOT store temporary data on any local client file systems.

3.3.2 Timers

None.

3.3.3 Initialization

The static virtual channel MUST be established, using the parameters as specified in section 2.1, before protocol operation can commence.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Initialization Sequence

3.3.5.1.1 Sending a Server Clipboard Capabilities PDU

The fields of the Clipboard Capabilities PDU are as specified in section 2.2.2.1.

After sending the Clipboard Capabilities PDU, the server MUST send the Monitor Ready PDU to the client, as specified in section 3.3.5.1.2.

3.3.5.1.2 Sending a Monitor Ready PDU

The fields of the Monitor Ready PDU are specified in section 2.2.2.2.

After sending the Monitor Ready PDU, the server can start processing clipboard updates contained in Format List PDUs, which it receives from the client, as specified in section 3.1.5.2.2. The server MUST still be prepared to receive and process the client Clipboard Capabilities PDU (as specified in section 3.3.5.1.3) and Temporary Directory PDU, as specified in section 3.3.5.1.4.

3.3.5.1.3 Processing a Client Clipboard Capabilities PDU

The fields of the Clipboard Capabilities PDU are specified in section 2.2.2.1.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the capability data MUST be extracted and stored in the Client Capabilities store as specified in 3.3.1.1.

3.3.5.1.4 Processing a Temporary Directory PDU

The fields of the Temporary Directory PDU are specified in section 2.2.2.3.

The **clipHeader** field MUST be processed as specified in section 3.1.5.1. If the PDU is valid, the temporary directory path MUST be extracted and stored in the Client Temporary Directory store (section 3.3.1.2).

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

4.1 Annotated Initialization Sequence

The following is an annotated dump of an Initialization Sequence (section 1.3.2.1).

4.1.1 Server Clipboard Capabilities PDU

The following is an annotated dump of a server-to-client Clipboard Capabilities PDU.

```
00000000 07 00 00 00 10 00 00 00 01 00 00 00 01 00 0c 00 ................
00000010 02 00 00 00 0e 00 00 00                         ........


07 00 -> CLIPRDR_HEADER::msgType = CB_CLIP_CAPS (7)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
10 00 00 00 -> CLIPRDR_HEADER::dataLen = 0x10 = 16 bytes
01 00 -> CLIPRDR_CAPS::cCapabilitiesSets = 1
00 00 -> CLIPRDR_CAPS::pad1
01 00 -> CLIPRDR_CAPS_SET::capabilitySetType = CB_CAPSTYPE_GENERAL (1)
0c 00 -> CLIPRDR_CAPS_SET::lengthCapability = 0x0c = 12 bytes

02 00 00 00 -> CLIPRDR_GENERAL_CAPABILITY::version = CB_CAPS_VERSION_2 (2)
0e 00 00 00 -> CLIPRDR_GENERAL_CAPABILITY::capabilityFlags = 0x0000000e
0x0e
= 0x02 |
  0x04 |
  0x08
= CB_USE_LONG_FORMAT_NAMES |
  CB_STREAM_FILECLIP_ENABLED |
  CB_FILECLIP_NO_FILE_PATHS
```

4.1.2 Server Monitor Ready PDU

The following is an annotated dump of a Monitor Ready PDU.

```
00000000 01 00 00 00 00 00 00 00 ........

01 00 -> CLIPRDR_HEADER::msgType = CB_MONITOR_READY (1)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
10 00 00 00 -> CLIPRDR_HEADER::dataLen = 0 bytes
```

4.1.3 Client Clipboard Capabilities PDU

The following is an annotated dump of a client-to-server Clipboard Capabilities PDU.

```
00000000 07 00 00 00 10 00 00 00 01 00 00 00 01 00 0c 00 ................
00000010 02 00 00 00 0e 00 00 00                         ........

07 00 -> CLIPRDR_HEADER::msgType = CB_CLIP_CAPS (7)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
10 00 00 00 -> CLIPRDR_HEADER::dataLen = 0x10 = 16 bytes
01 00 -> CLIPRDR_CAPS::cCapabilitiesSets = 1
```

```
00 00 -> CLIPRDR_CAPS::pad1
01 00 -> CLIPRDR_CAPS_SET::capabilitySetType = CB_CAPSTYPE_GENERAL (1)
0c 00 -> CLIPRDR_CAPS_SET::lengthCapability = 0x0c = 12 bytes


02 00 00 00 -> CLIPRDR GENERAL CAPABILITY::version = CB CAPS VERSION 2 (2)
0e 00 00 00 -> CLIPRDR GENERAL CAPABILITY::capabilityFlags = 0x0000000e
0x0e
= 0x02 |
  0x04 |
  0x08
= CB USE LONG FORMAT NAMES |
  CB_STREAM_FILECLIP_ENABLED |
  CB_FILECLIP_NO_FILE_PATHS
```

4.1.4 Client Temporary Directory PDU

The following is an annotated dump of a Temporary Directory PDU.

```
00000000 06 00 00 00 08 02 00 00 43 00 3a 00 5c 00 44 00  ........C.:.\.D.
00000010 4f 00 43 00 55 00 4d 00 45 00 7e 00 31 00 5c 00  O.C.U.M.E.~.1.\.
00000020 45 00 4c 00 54 00 4f 00 4e 00 53 00 7e 00 31 00  E.L.T.O.N.S.~.1.
00000030 2e 00 4e 00 54 00 44 00 5c 00 4c 00 4f 00 43 00  ..N.T.D.\.L.O.C.
00000040 41 00 4c 00 53 00 7e 00 31 00 5c 00 54 00 65 00  A.L.S.~.1.\.T.e.
00000050 6d 00 70 00 5c 00 63 00 64 00 65 00 70 00 6f 00  m.p.\.c.d.e.p.o.
00000060 74 00 73 00 6c 00 68 00 72 00 64 00 70 00 5f 00  t.s.l.h.r.d.p. .
00000070 31 00 5c 00 5f 00 54 00 53 00 41 00 42 00 44 00  1.\. .T.S.A.B.D.
00000080 2e 00 74 00 6d 00 70 00 00 00 00 00 00 00 00 00  ..t.m.p.........
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

06 00 -> CLIPRDR_HEADER::msgType = CB_TEMP_DIRECTORY (6)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
08 02 00 00 -> CLIPRDR_HEADER::dataLen = 0x208 = 520 bytes
43 00 3a 00 5c 00 44 00 4f 00 43 00 55 00 4d 00
45 00 7e 00 31 00 5c 00 45 00 4c 00 54 00 4f 00
4e 00 53 00 7e 00 31 00 2e 00 4e 00 54 00 44 00
5c 00 4c 00 4f 00 43 00 41 00 4c 00 53 00 7e 00
31 00 5c 00 54 00 65 00 6d 00 70 00 5c 00 63 00
```

```
64 00 65 00 70 00 6f 00 74 00 73 00 6c 00 68 00
72 00 64 00 70 00 5f 00 31 00 5c 00 5f 00 54 00
53 00 41 00 42 00 44 00 2e 00 74 00 6d 00 70 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 -> CLIPRDR_TEMP_DIRECTORY::wszTempDir =
"c:\docume~1\eltons~1.ntd\locals~1\Temp\cdepotslhrdp_1\_TSABD.tmp"
```

4.2 Annotated Copy Sequence

The following is an annotated dump of a Copy Sequence (section 1.3.2.2.1).

4.2.1 Format List PDU

The following is an annotated dump of a Format List PDU.

```
00000000 02 00 00 00 e0 00 00 00 8a c0 00 00 52 00 69 00  ............R.i.
00000010 63 00 68 00 20 00 54 00 65 00 78 00 74 00 20 00  c.h. .T.e.x.t. .
00000020 46 00 6f 00 72 00 6d 00 61 00 74 00 00 00 45 c1  F.o.r.m.a.t...E.
00000030 00 00 52 00 69 00 63 00 68 00 20 00 54 00 65 00  ..R.i.c.h. .T.e.
00000040 78 00 74 00 20 00 46 00 6f 00 72 00 6d 00 61 00  x.t. .F.o.r.m.a.
00000050 74 00 20 00 57 00 69 00 74 00 68 00 6f 00 75 00  t. .W.i.t.h.o.u.
00000060 74 00 20 00 4f 00 62 00 6a 00 65 00 63 00 74 00  t. .O.b.j.e.c.t.
00000070 73 00 00 00 43 c1 00 00 52 00 54 00 46 00 20 00  s...C...R.T.F. .
00000080 41 00 73 00 20 00 54 00 65 00 78 00 74 00 00 00  A.s. .T.e.x.t...
00000090 01 00 00 00 00 00 0d 00 00 00 00 00 04 c0 00 00  ................
000000a0 4e 00 61 00 74 00 69 00 76 00 65 00 00 00 0e c0  N.a.t.i.v.e.....
000000b0 00 00 4f 00 62 00 6a 00 65 00 63 00 74 00 20 00  ..O.b.j.e.c.t. .
000000c0 44 00 65 00 73 00 63 00 72 00 69 00 70 00 74 00  D.e.s.c.r.i.p.t.
000000d0 6f 00 72 00 00 00 03 00 00 00 00 00 10 00 00 00  o.r.............
000000e0 00 00 07 00 00 00 00 00                          ........

02 00 -> CLIPRDR_HEADER::msgType = CB_FORMAT_LIST (2)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
e0 00 00 00 -> CLIPRDR_HEADER::dataLen = 0xe0 = 224 bytes
```

```
8a c0 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0xc08a = 13
52 00 69 00 63 00 68 00 20 00 54 00 65 00 78 00
74 00 20 00 46 00 6f 00 72 00 6d 00 61 00 74 00
00 00 -> -> CLIPRDR_LONG_FORMAT_NAME::formatName = "Rich Text Format"
45 c1 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0xc145
52 00 69 00 63 00 68 00 20 00 54 00 65 00 78 00
74 00 20 00 46 00 6f 00 72 00 6d 00 61 00 74 00
20 00 57 00 69 00 74 00 68 00 6f 00 75 00 74 00
20 00 4f 00 62 00 6a 00 65 00 63 00 74 00 73 00
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName =
"Rich Text Format Without Objects"
43 c1 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0xc143
52 00 54 00 46 00 20 00 41 00 73 00 20 00 54 00
65 00 78 00 74 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName =
"RTF As Text"
01 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 1
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName = ""
0d 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0x0d = 13
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName = ""
04 c0 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0xc004
4e 00 61 00 74 00 69 00 76 00 65 00 00 00 -> "Native"
0e c0 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 0xc00e
4f 00 62 00 6a 00 65 00 63 00 74 00 20 00 44 00
65 00 73 00 63 00 72 00 69 00 70 00 74 00 6f 00
72 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName =
"Object Descriptor"
03 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 3
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName = ""
10 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 16
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName = ""
07 00 00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatId = 7
00 00 -> CLIPRDR_LONG_FORMAT_NAME::formatName = ""
```

4.2.2 Format List Response PDU

The following is an annotated dump of a Format List Response PDU.

```
00000000 03 00 01 00 00 00 00 00                          ........
03 00 -> CLIPRDR_HEADER::msgType = CB_FORMAT_LIST_RESPONSE (3)
01 00 -> CLIPRDR_HEADER::msgFlags = 0x0001 = CB_RESPONSE_OK
00 00 00 00 -> CLIPRDR_HEADER::dataLen = 0 bytes
```

4.3 Annotated Paste Sequence

The following is an annotated dump of a Paste Sequence (section 1.3.2.2.2).

4.3.1 Format Data Request PDU

The following is an annotated dump of a Format Data Request PDU.

```
00000000 04 00 00 00 04 00 00 00 0d 00 00 00              ............
05 00 -> CLIPRDR_HEADER::msgType = CB_FORMAT_DATA_REQUEST (4)
00 00 -> CLIPRDR_HEADER::msgFlags = 0
```

```
04 00 00 00 -> CLIPRDR_HEADER::dataLen = 4 bytes
0d 00 00 00 -> CLIPRDR_FORMAT_DATA_REQUEST::RequestedFormatId = 0x0d
```

4.3.2 Format Data Response PDU

The following is an annotated dump of a Format Data Response PDU.

```
00000010 05 00 01 00 18 00 00 00 68 00 65 00 6c 00 6c 00 ........h.e.l.l.
00000020 6f 00 20 00 77 00 6f 00 72 00 6c 00 64 00 00 00 o. .w.o.r.l.d...

05 00 -> CLIPRDR_HEADER::msgType = CB_FORMAT_DATA_RESPONSE (5)
01 00 -> CLIPRDR_HEADER::msgFlags = 0x0001 = CB_RESPONSE_OK
18 00 00 00 -> CLIPRDR_HEADER::dataLen = 0x18 = 24 bytes


68 00 65 00 6c 00 6c 00 6f 00 20 00 77 00 6f 00
72 00 6c 00 64 00 00 00 -> CLIPRDR_FORMAT_DATA_RESPONSE::requestedFormatData
```

5 Security

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

5.1 Security Considerations for Implementers

There are no security considerations for protocol messages because all static virtual channel traffic is secured by the underlying core Remote Desktop Protocol. An overview of the implemented security-related mechanisms is as specified in [MS-RDPBCGR] section 5.

5.2 Index of Security Parameters

None.

6 Appendix A: Windows Behavior

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

- Windows Server 2008
- Windows Server 2003
- Windows 2000 Server
- Windows Vista
- Windows XP

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.

<1> Section 3.1.1.2: On Windows-based systems, the file list is encapsulated in a "File Group Descriptor" (for more information, see [MSDN-SHELLCLIP], "CFSTR_FILEDESCRIPTOR") generic data format. This format consists of an array of File Descriptors, each of which describes a single file in a collection. The Format Name of the File Group Descriptor format is "FileGroupDescriptorW" if the descriptor is in Unicode format; otherwise, it is "FileGroupDescriptor". The Clipboard Format ID is assigned dynamically and is not constant across computer systems.

7 Index

V

W