[MS-RRSP2-Diff]:

Remote Rendering Server Protocol Version 2.0

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards ~~as well as overviews of the interaction among each of these technologies~~support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you ~~may~~can make copies of it in order to develop implementations of the technologies that are described in ~~the Open Specifications~~ this documentation and ~~may~~can distribute portions of it in your implementations ~~using~~that use these technologies or in your documentation as necessary to properly document the implementation. You ~~may~~can also distribute in your implementation, with or without modification, any ~~schema, IDL's~~schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications~~.~~ documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that ~~may~~might cover your implementations of the technologies described in the Open Specifications~~.~~ documentation. Neither this notice nor Microsoft's delivery of ~~the~~this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open ~~Specification may~~Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in ~~the Open Specifications~~this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation ~~may~~might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, ~~e-mail~~email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
8/27/2010	0.1	New	Released new document.
10/8/2010	0.1	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	0.1	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	0.1	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	0.2	Minor	Clarified the meaning of the technical content.
9/23/2011	0.2	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	1.0	Major	Updated and revised the technical content.
3/30/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	2.0	Major	Updated and revised the technical content.
8/8/2013	3.0	Major	Updated and revised the technical content.
11/14/2013	4.0	Major	Updated and revised the technical content.
2/13/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	5.0	Major	Significantly changed the technical content.
10/16/2015	5.0	~~No Change~~None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Remote Rendering Protocol Version 2, is a user interface system for applications in Windows Media Center, which is comprised of an application-side component model connected to a remote renderer by an asynchronous messaging system that enables the quick and easy construction of captivating interfaces.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative ~~and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms~~. All other sections and examples in this specification are informative.

1.1 Glossary

~~The~~This document uses the following terms ~~are specific to this document~~:

ARGB: A color space wherein each color is represented as a quad (A, R, G, B), where A represents the alpha (transparency) component, R represents the red component, G represents the green component, and B represents the blue component. The ARGB value is typically stored as a 32-bit integer, wherein the alpha channel is stored in the highest 8 bits and the blue value is stored in the lowest 8 bits.

context: Logical container spaces where objects exist "together" in memory and can efficiently communicate with each other.

handle: A recipient of a message.

network byte order: The order in which the bytes of a multiple-byte number are transmitted on a network, most significant byte first (in big-endian storage). This may or may not match the order in which numbers are normally stored in memory for a particular processor.

renderer: A component that is responsible for receiving draw and animation commands, and rendering the scene to an output device.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[MS-DSPA] Microsoft Corporation, "Device Session Property Access Protocol".

[MS-DTAG] Microsoft Corporation, "Device Trust Agreement Protocol".

[MS-RXAD] Microsoft Corporation, "Remote Experience Advertisement Protocol".

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

1.2.2 Informative References

None.

1.3 Protocol Overview (Synopsis)

The Remote Rendering Protocol Version 2 enables the creation of interfaces in a remote **renderer** through an asynchronous messaging system. The application-side component model (server) connected to the remote renderer (client) can be deployed within a single process, across multiple processes, or across multiple computers on a network over a reliable point-to-point connection.

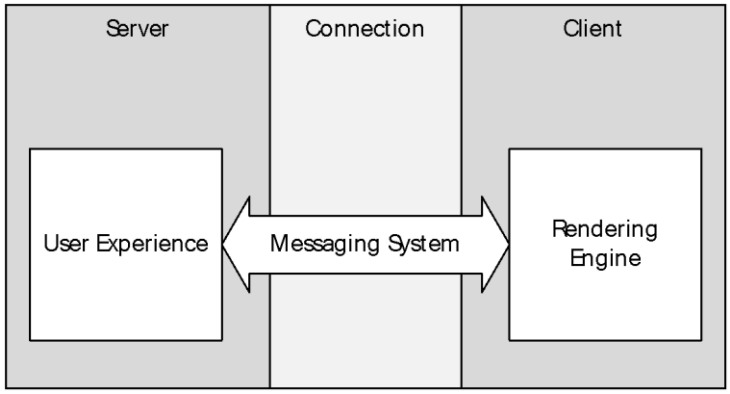


Figure 1: Point-to-point connection

1.3.1 User Experience

The Remote Rendering Protocol Version 2 component model defines a basic programming model and several reusable services for various user interface (UI) tasks, hereafter referred to as the "user experience". The primary logic runs in process with the client application, which isolates the developer from the more complex implementation details of rendering and asynchronous communication.

1.3.1.1 Internal Componentization

The messages shared through the messaging system are contained within the following components/classes. The server provides the appropriate information to the client before the messages can be executed.

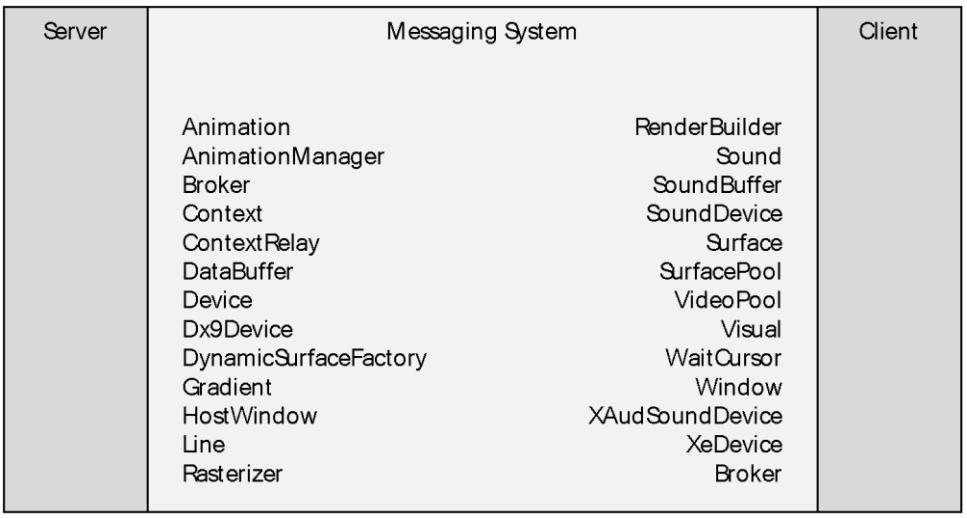


Figure 2: Components

The following components make up the remaining implementation of the Remote Rendering Protocol Version 2 component model.

1.3.2 Rendering Engine

Remote Rendering Protocol Version 2 is designed to work with a mid-level application compositing rendering engine that can operate independently of the application and is driven by a stream of asynchronous rendering commands that describe the scenes to be displayed. The renderer has to be capable of running autonomously for significant periods of time in the absence of new commands from the application.

1.3.2.1 Addressing Mechanism

An internal addressing mechanism for delivering messages to objects is layered over the transport.

This mechanism is encompassed by:

1. **Contexts**, which are logical container spaces where objects exist "together" in memory and can efficiently communicate with each other.

2. A **handle**, which specifies the recipients of the message because all messages are addressed to some endpoint with a specific handle.

Both class and instance handles are valid. A message to a class handle is called a "static message". A message to an instance handle is called an "instance message".

1.3.3 Message Sequence

The following messages are sent during the lifetime of a given remote service. The legend below describes the sequence these messages use during the service.

The messages that could take place during the session are those that depend on the implementation (what the user interface communicates to the renderer); therefore, these messages are not necessarily present during a specific session. The messages that are present during the session are those that encompass Remote Rendering Protocol Version 2.

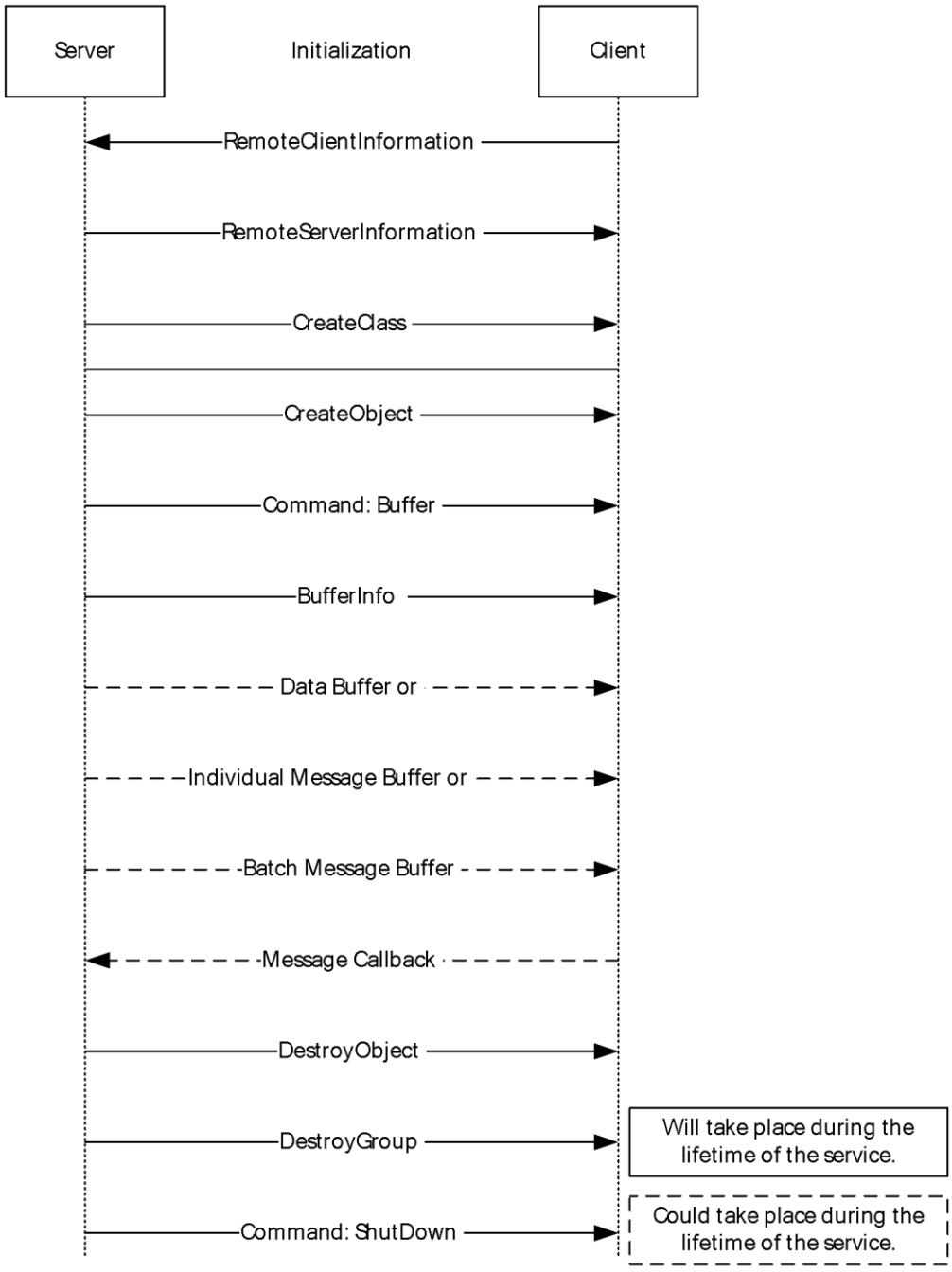


Figure 3: Message sequence

1.4 Relationship to Other Protocols

The Remote Rendering Protocol Version 2 is not a standalone protocol. It depends on an established connection between the server and the client, which is covered in [MS-DTAG] Device Trust Agreement and [MS-RXAD] Remoted Experience Advertisement.

1.5 Prerequisites/Preconditions

Other than the relationship called out in the previous section:

1. Communication to the device has to have been established.

2. The server capabilities (graphics, memory, and so on) have to be identified and the information has to be provided to Remote Rendering Protocol Version 2 by the Device Session Property Access Protocol.

1.6 Applicability Statement

Remote Rendering Protocol Version 2 is applicable to environments that require the ability to send rendering instructions over a reliable and pre-established connection to a remote renderer.

1.7 Versioning and Capability Negotiation

Remote Rendering Protocol Version 2 does not handle versioning and capabilities directly; instead, they are taken care of by the Device Session Property Access Protocol because Remote Rendering Protocol Version 2 is carried over this protocol. For further information please refer to section 1.6 of [MS-DSPA].

1.8 Vendor-Extensible Fields

None

1.9 Standards Assignments

None

2 Messages

2.1 Transport

Remote Rendering Protocol Version 2 is transport agnostic and can be carried over any reliable transport.

2.2 Message Syntax

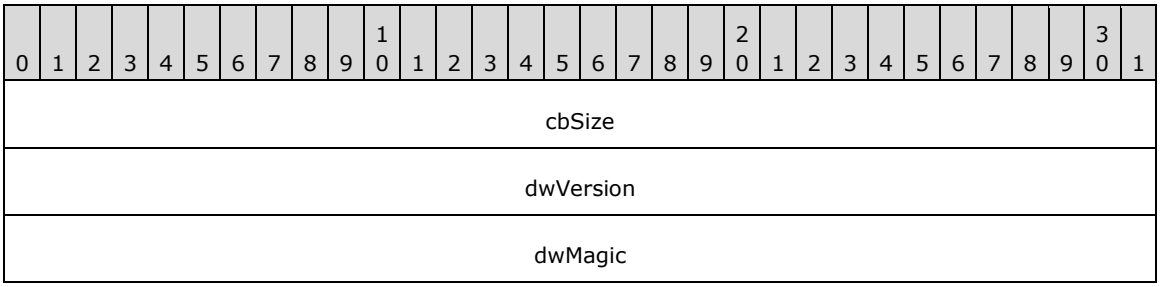
2.2.1 Initialization Messages (Handshake)

Upon establishing a transport connection, the client sends a RemoteClientInformation message, as specified in section 2.2.1.1. Next, the server sends a RemoteServerInformation message, as specified in section 2.2.1.2.

Initialization messages are sent in **network byte order**.

2.2.1.1 RemoteClientInformation message

The RemoteClientInformation message is used to send the client's information that is required for initialization to the server.



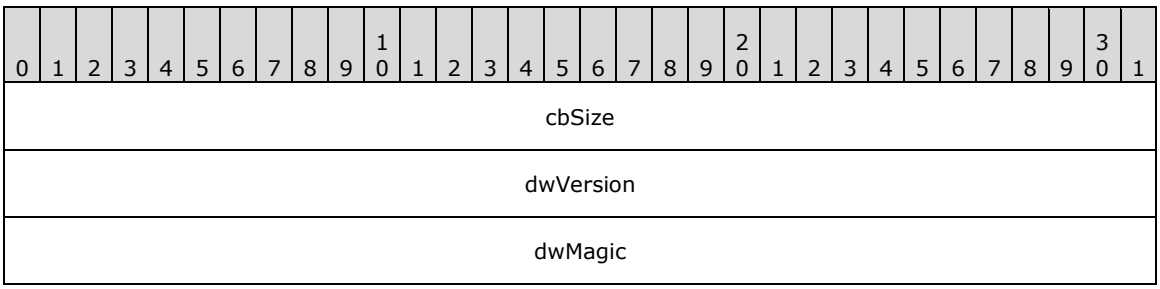
cbSize (4 bytes): An unsigned 32-bit integer. The size of the message.

dwVersion (4 bytes): An unsigned 32-bit integer. The client MUST be set this field to 0x00010006. The version of the client's network pipe.

dwMagic (4 bytes): An unsigned 32-bit integer. The client MUST set this field to 0x19740721. A number used to identify the protocol family of the client.

2.2.1.2 RemoteServerInformation message

The RemoteServerInformation message sends information about the server to the client.



idContextApplication
idContextRender
dwReserved1
cItemsPerGroupBits
cGroupBits
idObjectBrokerClass

cbSize (4 bytes): An unsigned 32-bit integer. The size of the message.

dwVersion (4 bytes): An unsigned 32-bit integer. The client MUST be set this field to 0x00010006. The version of the server's network pipe.

dwMagic (4 bytes): An unsigned 32-bit integer. The client MUST be set this field to 0x19740721. A number used to identify the protocol family of the server.

idContextApplication (4 bytes): An unsigned 32-bit integer. Contains the context ID of the server.

idContextRender (4 bytes): An unsigned 32-bit integer. Contains the context ID to which the receiving client is being assigned.

dwReserved1 (4 bytes): An unsigned 32-bit integer. Unused and MUST be set to zero.

cItemsPerGroupBits (4 bytes): A signed 32-bit integer. Specifies how many bits in the handle are used for object indices within a group.

cGroupBits (4 bytes): A signed 32-bit integer. Specifies how many bits in the handle are used for "groups" of objects.

idObjectBrokerClass (4 bytes): An unsigned 32-bit integer. A predefined handle to the "broker" class.

2.2.2 Command Messages

Once the handshake is completed, the connection is open for either client or server to send commands. There are two types of commands that can be sent: buffer commands and shutdown commands.

Command messages are sent in network byte order.

2.2.2.1 Command Message

A command message is used to signal that either a buffer and payload are following, or that the endpoint has to shut down.

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
nCommandType																															

nCommandType (4 bytes): An unsigned 32-bit integer. The type of command to follow.

Defined types are described in the following table.

Value	Description
0x00000001	Buffer: Followed by a buffer information header and associated buffer payload.
0x00000002	Shutdown: Last message sent. Endpoint will no longer communicate.

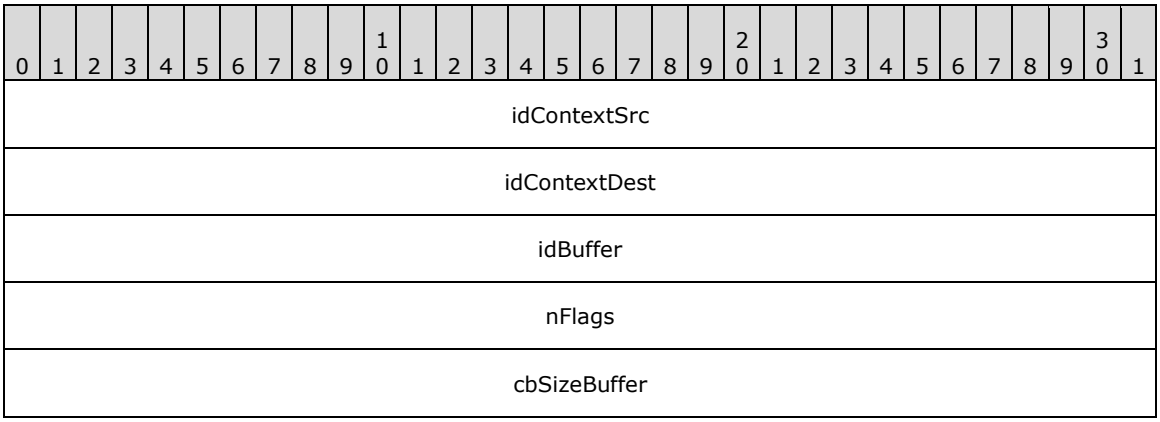
2.2.3 Framing Messages

If a command message is sent as a buffer command type, it is followed by a sequence of messages, starting with a BufferInfo message.

Framing messages are sent in network byte order.

2.2.3.1 BufferInfo Message

Sends information about the message to follow.



idContextSrc (4 bytes): An unsigned 32-bit integer. Contains the unique context ID of the sender.

idContextDest (4 bytes): An unsigned 32-bit integer. Contains the unique context ID of the recipient.

idBuffer (4 bytes): An unsigned 32-bit integer. Contains the unique ID for the buffer, which can be any of the following:

1. Data Buffer: The buffer has a non-null ObjectID (idBuffer) and the memory can be associated with a new DataBuffer instance bearing this handle. No broker creation sequence will precede this transaction; the DataBuffer instance is created implicitly when the data is received.

2. Individual Message Buffer: The buffer has a NULL ObjectID (idBuffer) and the IsBatch flag will be cleared. The payload can be interpreted as a single message and processed.

3. Batch Message Buffer: The buffer has a NULL ObjectID (idBuffer) and the IsBatch flag will be SET. The payload can be interpreted as a batch with multiple messages and processed in order.

nFlags (4 bytes): An unsigned 32-bit integer. Describes the BufferFlags.

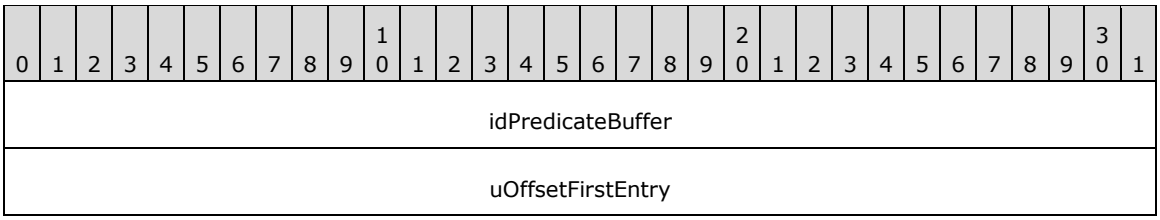
Value	Description
0x00000001	IsBatch. The Buffer is a batch of messages.

All other flags are reserved, MUST be ignored, and MUST never be specified.

cbSizeBuffer (4 bytes): An unsigned 32-bit integer. The size of the buffer's data.

2.2.3.2 MessageBatch Message

If the BufferInfo message's ObjectID is NULL and IsBatch flag is set, the next message is a MessageBatch.



idPredicateBuffer (4 bytes): An unsigned 32-bit integer. This is the ID of the previously sent buffer that MUST be processed before this one.

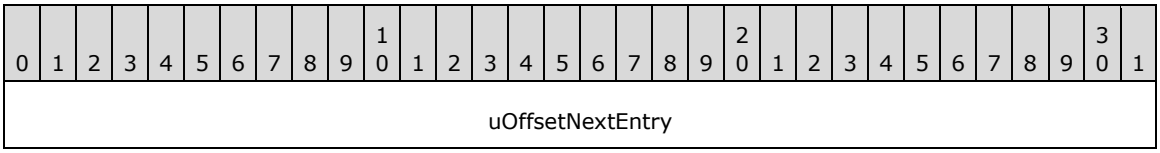
If idPredicateBuffer is not 0x00000000, this refers to a previously transmitted data buffer that can be processed as a batch buffer prior to processing this message. That buffer can also refer to another predicate buffer (and so on).

If idPredicateBuffer is 0x00000000, the following message entries will be processed.

uOffsetFirstEntry (4 bytes): An unsigned 32-bit integer. The size of the buffer offset of the first message entry.

2.2.3.3 MessageBatchEntry Message

A message batch can contain one or more message entries, which are identified by the following header:

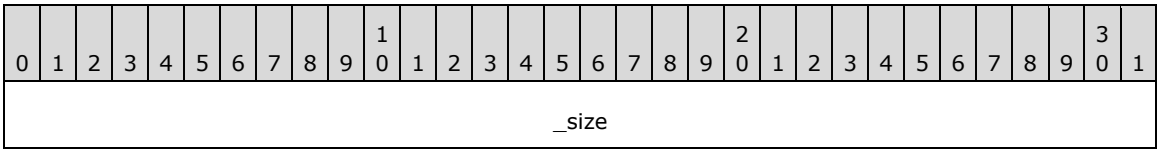


uOffsetNextEntry (4 bytes): An unsigned 32-bit integer. This is the size of the buffer offset of the next message entry. The final entry in a batch has a uOffsetNextEntry of 0x00000000.

2.2.4 Payload Messages

Payload messages are used to issue rendering and sound commands. Payload messages are sent in client-byte order, as determined by the extender capabilities exchange.

Every payload message has the standard header listed in the following table.



_msgid
_idObjectSubject

_size (4 bytes): An unsigned 32-bit integer. Describes the total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The value specified in _msgid is used to indicate which action to take on the target object.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object. The _idObjectSubject refers to an object that was previously created by sending a Broker_CreateObject payload.

2.2.4.1 DataBuffer

2.2.4.1.1 DataBuffer_RegisterOwner

The DataBuffer_RegisterOwner message registers the owner of the buffer. The owner is notified when the buffer usage is complete, which allows the owner to reclaim resources.

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
_size																															
_msgid																															
_idObjectSubject																															
_objcb																															
_ctxcb																															

_size (4 bytes): An unsigned 32-bit integer. Describes the total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.2 ContextRelay

2.2.4.2.1 ContextRelay_Create

The ContextRelay_Create message creates a transport bridge to relay messages from a remote application to an existing context.

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
_size																															
_msgid																															
_idObjectSubject																															
protocol																															
stServer																															
stSession																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

protocol (4 bytes): A signed 32-bit integer. The transport protocol to use for communication.

The possible values are:

Value	Description
0x00000001	RDP Virtual Channel
0x00000002	TCP
0x00000003	UDP
0x00000004	Named Pipes

stServer (4 bytes): A BLOBREF (section 2.2.6.1) that specifies the name of the remote server or address.

stSession (4 bytes): A BLOBREF (section 2.2.6.1) that specifies the name for the local session. This value is only used for Named Pipes; otherwise, this value is ignored.

2.2.4.2.2 ContextRelay_UnlinkContext

The ContextRelay_UnlinkContext message disassociates the specified context alias from an existing context.

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
_size																															
_msgid																															
_idObjectSubject																															
idContextExisting																															

idContextAlias

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idContextExisting (4 bytes): An unsigned 32-bit integer. The ID of the existing context.

idContextAlias (4 bytes): An unsigned 32-bit integer. The ID of the alias context to be unlinked from the existing context.

2.2.4.2.3 ContextRelay_LinkContext

The ContextRelay_LinkContext message links the specified context alias to an existing context.

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
_size																															
_msgid																															
_idObjectSubject																															
idContextExisting																															
idContextAlias																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idContextExisting (4 bytes): An unsigned 32-bit integer. The ID of the existing context.

idContextAlias (4 bytes): An unsigned 32-bit integer. The ID of the alias context to be linked to the existing context.

2.2.4.3 Broker

2.2.4.3.1 Broker_DestroyObject

The Broker_DestroyObject message destroys a previously created object. The object is destroyed immediately.

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
_size																															

_msgid
_idObjectSubject
idObject

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idObject (4 bytes): An unsigned 32-bit integer. The ID of the object to be destroyed.

2.2.4.3.2 Broker_CreateObject

The Broker_CreateObject message creates a new instance of the specified class.

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
_size																															
_msgid																															
_idObjectSubject																															
idObjectClass																															
idObjectNew																															
msgConstruction																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idObjectClass (4 bytes): An unsigned 32-bit integer. The ID of the object class to be created.

idObjectNew (4 bytes): An unsigned 32-bit integer. The ID that is assigned to the created object instance. The Object ID MUST be unique for the given context.

msgConstruction (4 bytes): A BLOBREF (section 2.2.6.1) that specifies a reference to a construction parameters message.

2.2.4.3.3 Broker_CreateClass

The Broker_CreateClass message creates a new object that can be used to identify a Class.

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
_size																															
_msgid																															
_idObjectSubject																															
stClassName																															
idObjectClass																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

stClassName (4 bytes): A BLOBREF (section 2.2.6.1) that specifies the name of the remote server or address.

idObjectClass (4 bytes): An unsigned 32-bit integer. The ID that is assigned to the object class. The Object ID MUST be unique for the given context.

2.2.4.4 Context

2.2.4.4.1 Context_ForwardMessage

The Context_ForwardMessage message forwards the given message to the given object. This message can be used by a component that is required to be called back after a set of prior messages have been processed.

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
_size																															
_msgid																															
_idObjectSubject																															
idContextDest																															
msgReturn																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

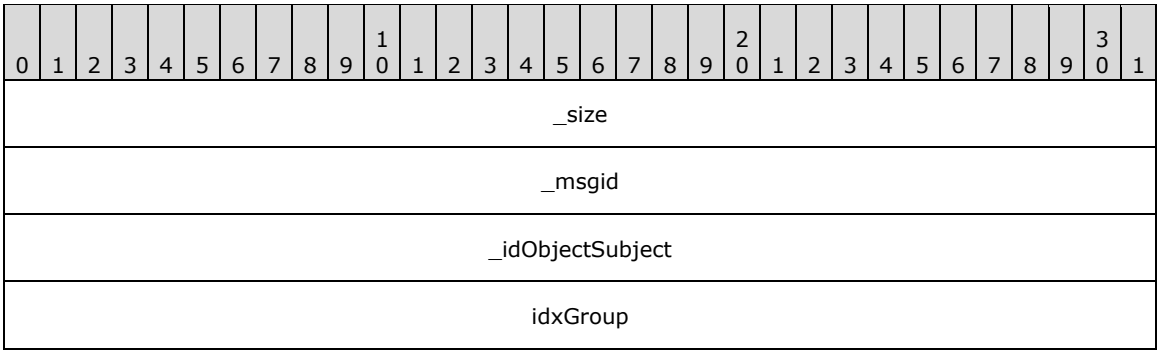
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idContextDest (4 bytes): An unsigned 32-bit integer. The destination context for the message.

msgReturn (4 bytes): A BLOBREF (section 2.2.6.1) that specifies the message to send.

2.2.4.4.2 Context_DestroyGroup

The Context_Destroyroup message destroys a collection of objects, including the objects themselves, in the given context.



_size (4 bytes): An unsigned32-bit integer. It describes the total message size, in bytes.

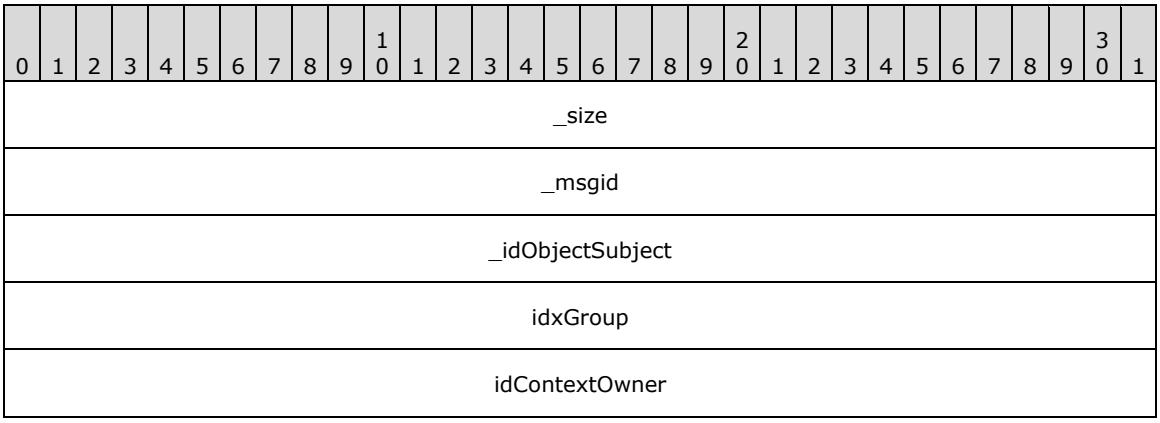
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxGroup (4 bytes): A signed 32-bit integer. The unique ID of the group.

2.2.4.4.3 Context_CreateGroup

The Context_CreateGroup message creates a collection of objects within the given context.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

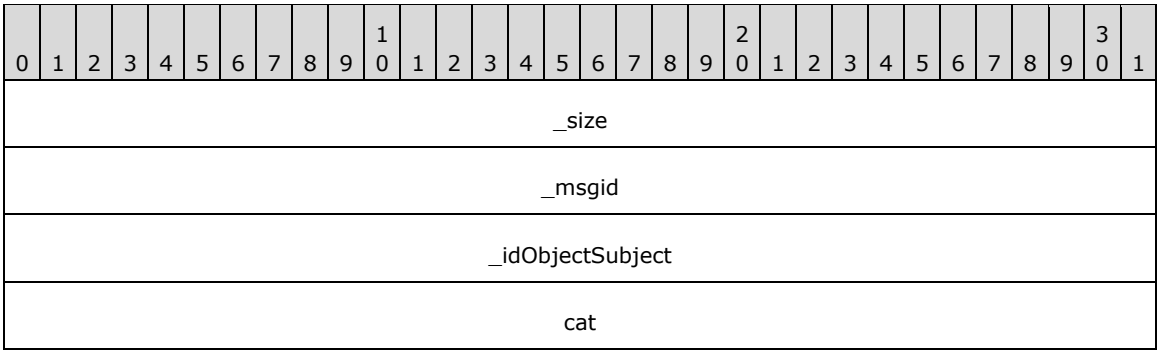
idxGroup (4 bytes): A signed 32-bit integer. The unique ID of the group.

idContextOwner (4 bytes): An unsigned 32-bit integer. The context that owns the group.

2.2.4.5 RenderBuilder

2.2.4.5.1 RenderBuilder_Create

The RenderBuilder_Create message completes construction of a new RenderBuilder.



_size (4 bytes): An unsigned32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

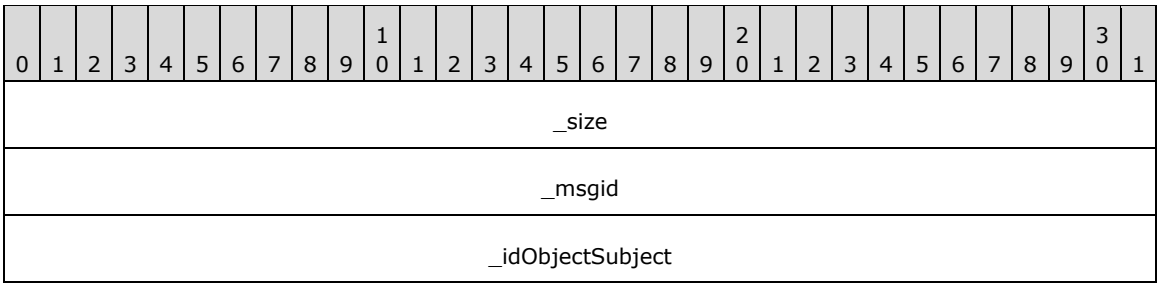
cat (4 bytes): A signed-32 bit integer. Indicates whether the render operations can occur pre-scene or in-scene.

Possible values are listed in the following table.

Value	Description
0x00000000	Pre-scene
0x00000001	In-scene

2.2.4.5.2 RenderBuilder_Clear

The RenderBuilder_Clear message empties the contents of this RenderBuilder, allowing it to be used for painting another object.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

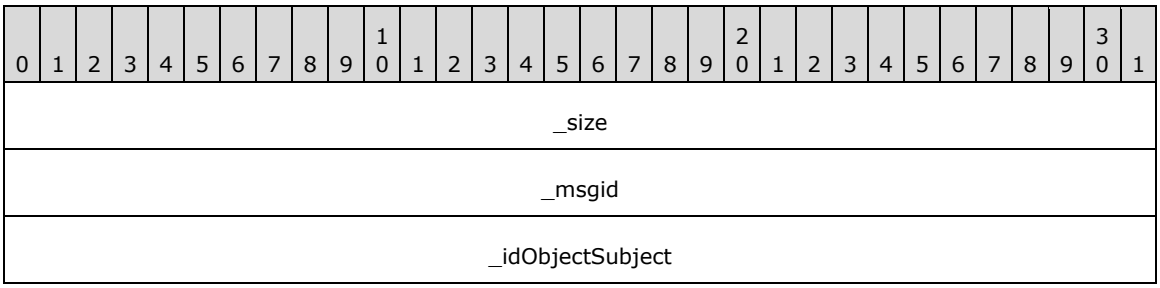
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.6 Visual

2.2.4.6.1 Visual_Create

The Visual_Create message completes construction of a new visual.



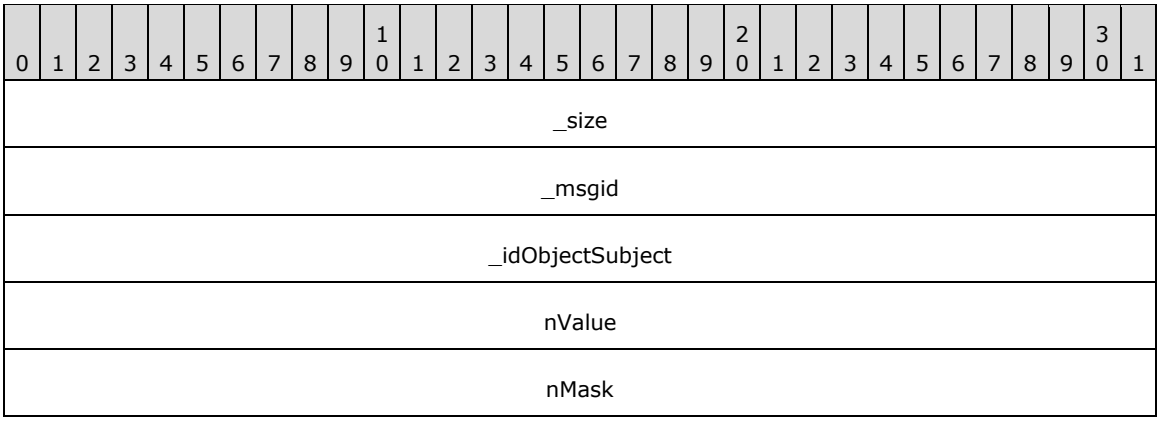
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000001A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.6.2 Visual_ChangeDataBits

The Visual_ChangeDataBits message changes the user-defined bits set on the target visual.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

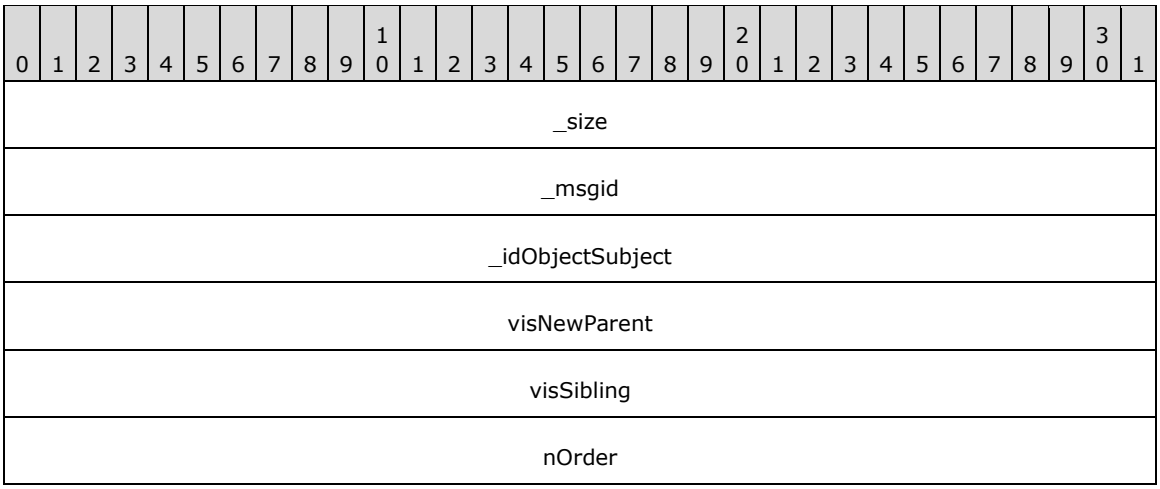
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nValue (4 bytes): An unsigned 32-bit integer. The new value.

nMask (4 bytes): An unsigned 32-bit integer. A mask to use when changing the bits.

2.2.4.6.3 Visual_ChangeParent

Changes the parent and z-order inside the sub-tree.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

visNewParent (4 bytes): An unsigned 32-bit integer. The ID of the new parent visual.

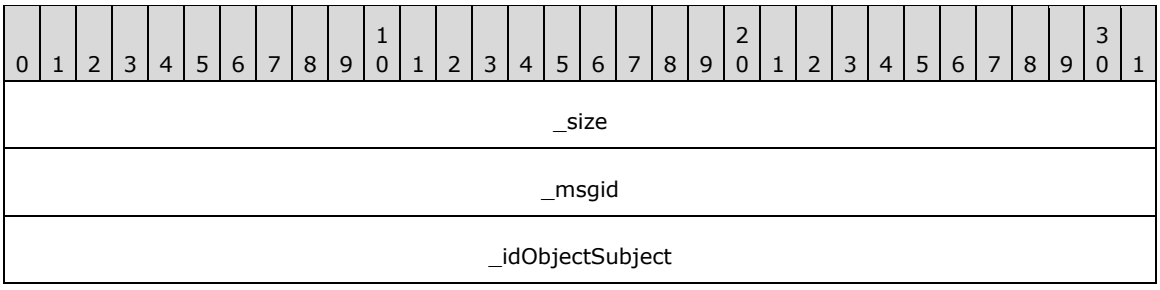
visSibling (4 bytes): An unsigned 32-bit integer. The ID of the sibling visual.

nOrder (4 bytes): A signed 32-bit integer. The place to add the visual, relative to the sibling. Possible values are listed in the following table.

Value	Description
0x00000000	Any - Any position amongst its siblings.
0x00000001	Before - Before the specified sibling.
0x00000002	Behind - Behind the specified sibling.
0x00000003	Top - The top of the parent's children list.
0x00000004	Bottom - The bottom of the parent's children list.

2.2.4.6.4 Visual_SetColor

The Visual_SetColor message sets the color value of the visual.



clr

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

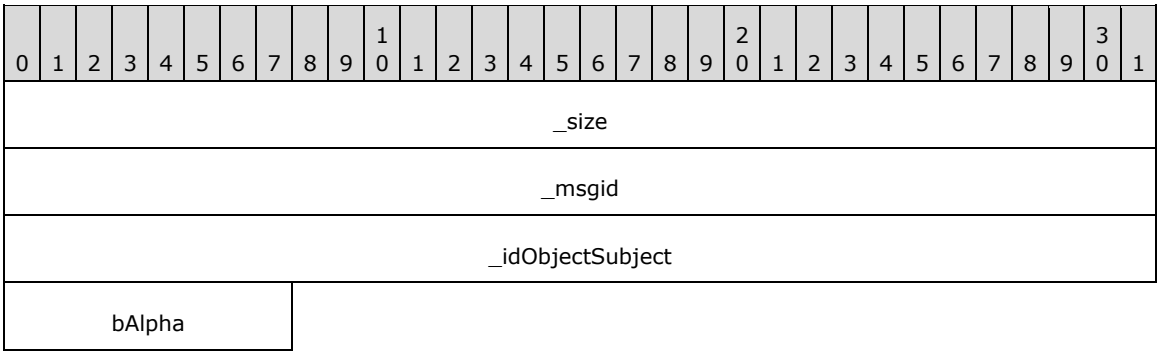
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clr (4 bytes): An unsigned 32-bit integer. The **ARGB** value of the color.

2.2.4.6.5 Visual_SetAlpha

The Visual_SetAlpha message sets the alpha value of the visual.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

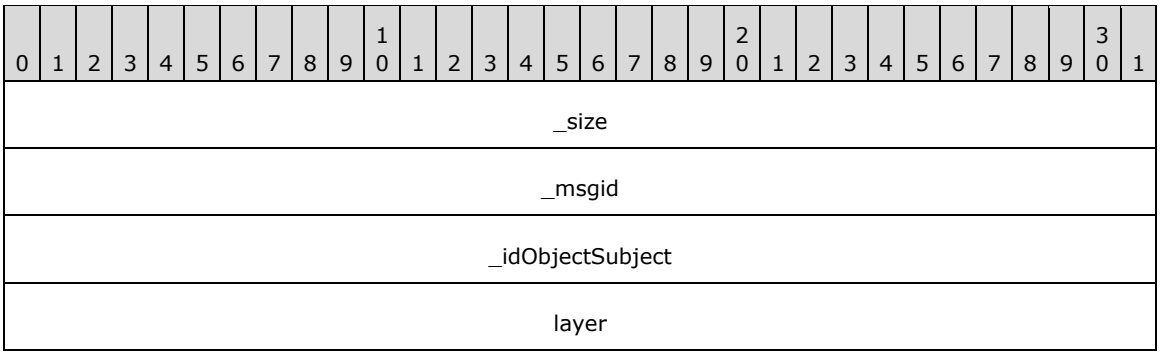
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

bAlpha (1 byte): A byte that specifies the alpha value.

2.2.4.6.6 Visual_SetLayer

The Visual_SetLayer message sets the layer number of the visual.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

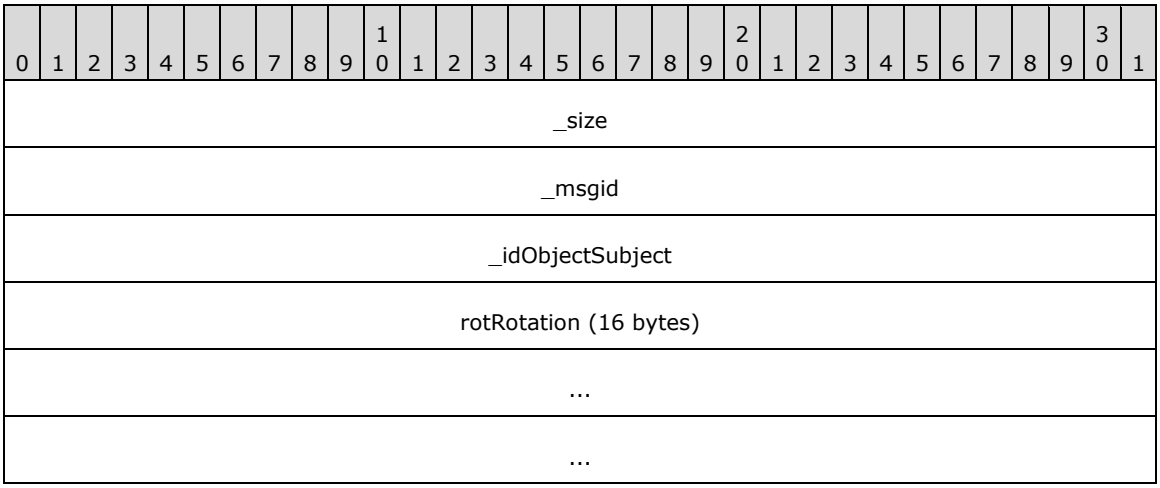
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

layer (4 bytes): An unsigned 32-bit integer. The layer number. The value MUST be between 0x00000000 (the back-most layer) and 4294967295 (the front-most layer).

2.2.4.6.7 Visual_SetRotation

The Visual_SetRotation message changes the current rotation that is assigned to the specific visual. Rotations of parents, siblings, and children are not changed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

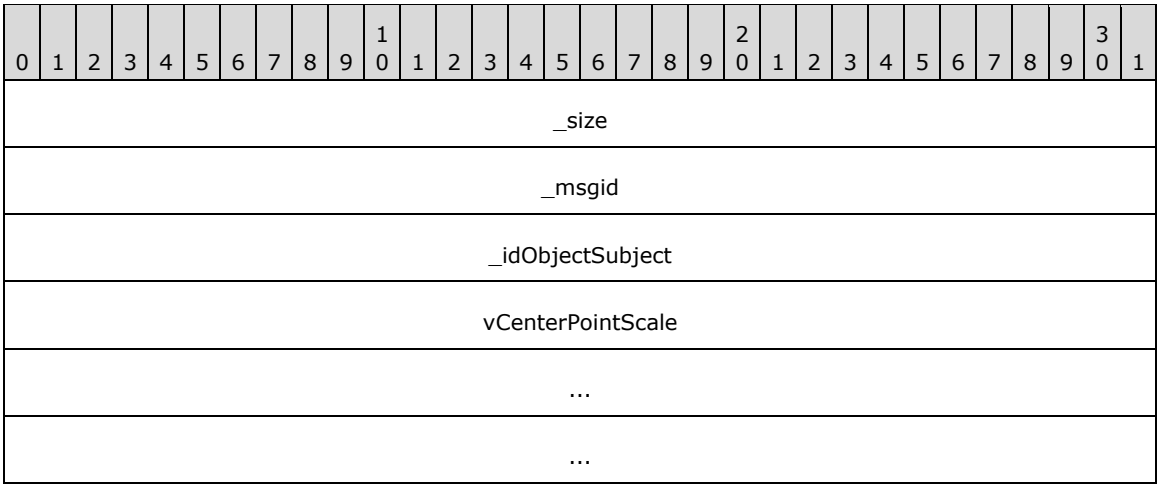
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rotRotation (16 bytes): A Rotation (section 2.2.6.2) that specifies the new rotation of the visual.

2.2.4.6.8 Visual_SetCenterPointScale

The Visual_SetCenterPointScale message changes the current center point scale that is assigned to the specific visual. Center point scales of parents, siblings, and children are not changed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

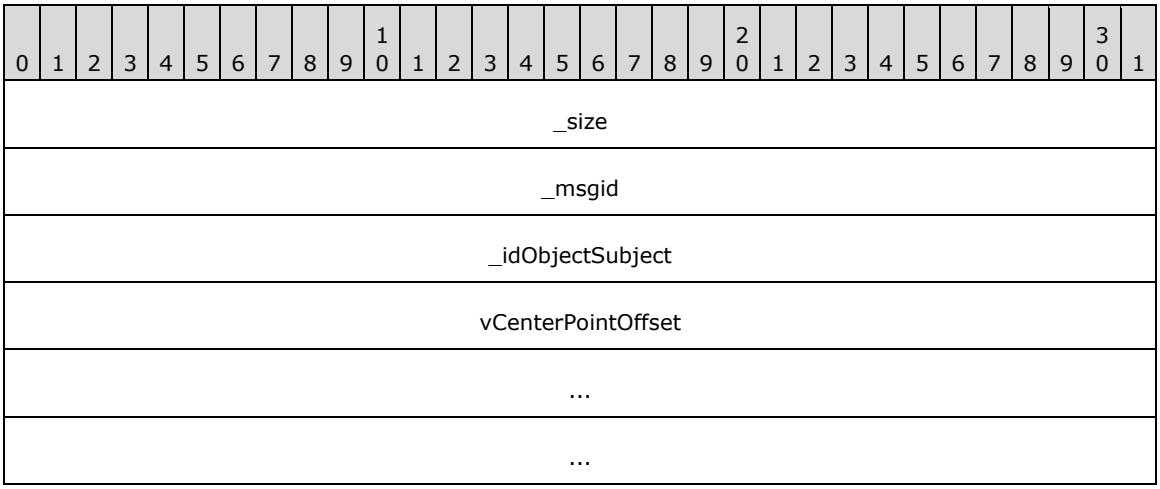
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000C for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

vCenterPointScale (12 bytes): A Vector3 (section 2.2.6.3) that specifies the center point scale of the visual.

2.2.4.6.9 Visual_SetCenterPointOffset

The Visual_SetCenterPointOffset changes the current center point that is assigned to the specific visual. Center points of parents, siblings, and children are not changed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

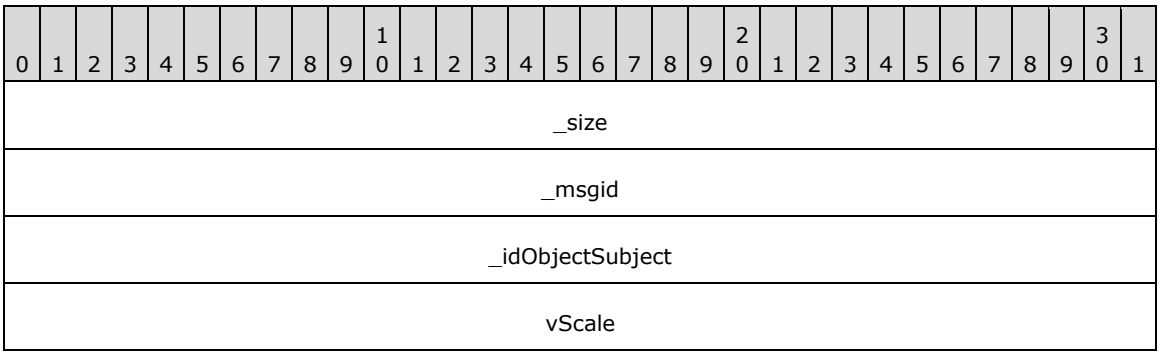
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000E for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

vCenterPointOffset (12 bytes): A Vector3 (section 2.2.6.3) that specifies the center point of the visual.

2.2.4.6.10 Visual_SetScale

The Visual_SetScale message changes the current scaling factor assigned to the specific visual. Scaling factors of parents, siblings, and children are not changed.



...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000010 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

vScale (12 bytes): A Vector3 (section 2.2.6.3) that specifies the scale of the visual.

2.2.4.6.11 Visual_SetSize

The Visual_SetSize message changes the width, height, and depth of the visual, relative to itself.

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
_size																															
_msgid																															
_idObjectSubject																															
vSizePxl																															
...																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000012 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

vSizePxl (12 bytes): A Vector3 (section 2.2.6.3) that specifies the size of the visual.

2.2.4.6.12 Visual_SetPosition

The Visual_SetPosition message changes the X, Y, and Z of the visual, relative to its parent.

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
_size																															
_msgid																															
_idObjectSubject																															

vPositionPxl
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000014 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

vPositionPxl (12 bytes): A Vector3 (section 2.2.6.3) that specifies the position of the visual.

2.2.4.6.13 Visual_SetContent

The Visual_SetContent message transfers the RenderOperation contents from the given RenderBuilder into the visual.

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
_size																															
_msgid																															
_idObjectSubject																															
rbContent																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000017 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rbContent (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder.

2.2.4.6.14 Visual_SetVisible

The Visual_SetVisible message determines whether the given visual participates in rendering and hit-testing.

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
_size																															
_msgid																															
_idObjectSubject																															

fVisible

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000018 for this message.

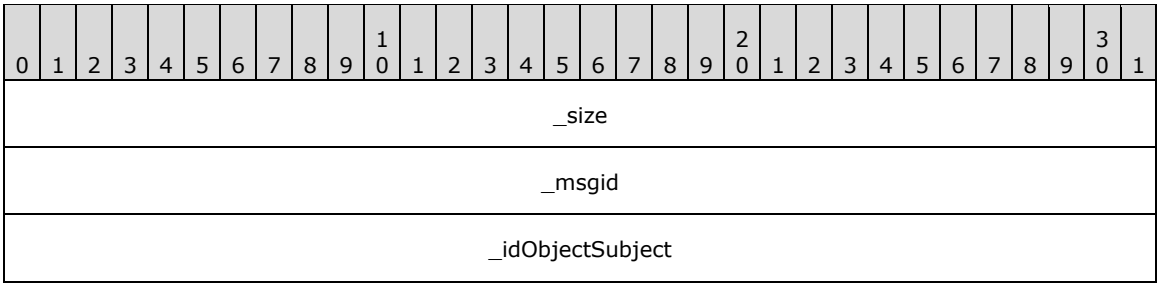
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

fVisible (4 bytes): An unsigned 32-bit integer. Visibility value.

2.2.4.7 AnimationManager

2.2.4.7.1 AnimationManager_Create

The AnimationManager_Create message builds a new AnimationManager for the given context.



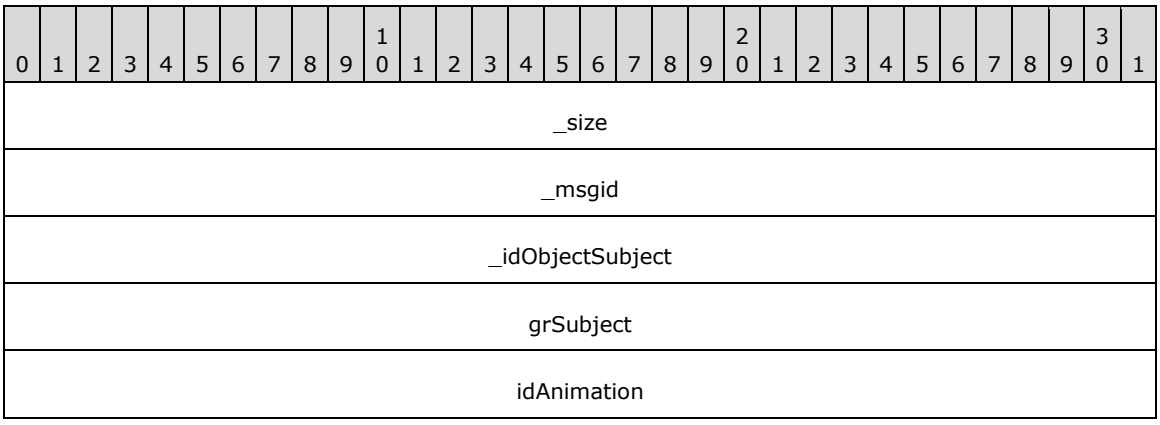
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.7.2 AnimationManager_BuildGradientColorMaskAnimation

The AnimationManager_BuildGradientColorMaskAnimation message builds an animation to modify a gradient's ColorMask.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

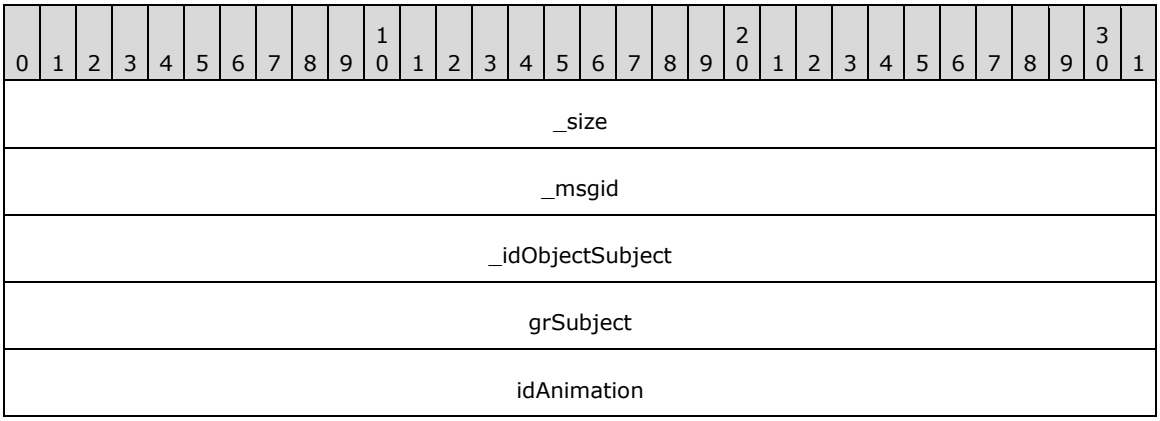
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

grSubject (4 bytes): An unsigned 32-bit integer. The ID of the target gradient object.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.3 AnimationManager_BuildGradientOffsetAnimation

The AnimationManager_BuildGradientOffsetAnimation message builds an animation to modify a gradient.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

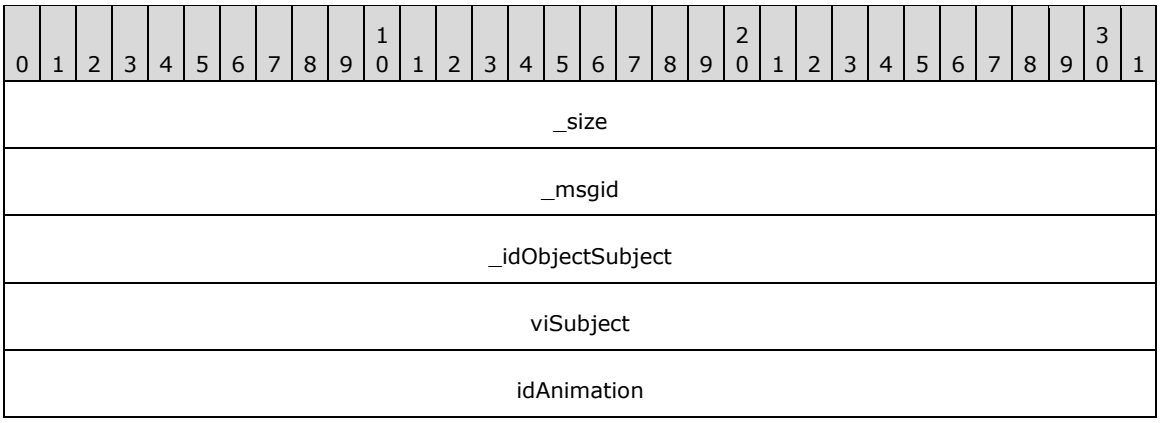
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

grSubject (4 bytes): An unsigned 32-bit integer. The ID of the target gradient object.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.4 AnimationManager_BuildRotationAnimation

The AnimationManager_BuildRotationAnimation message builds an animation to modify the visual's rotation property.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

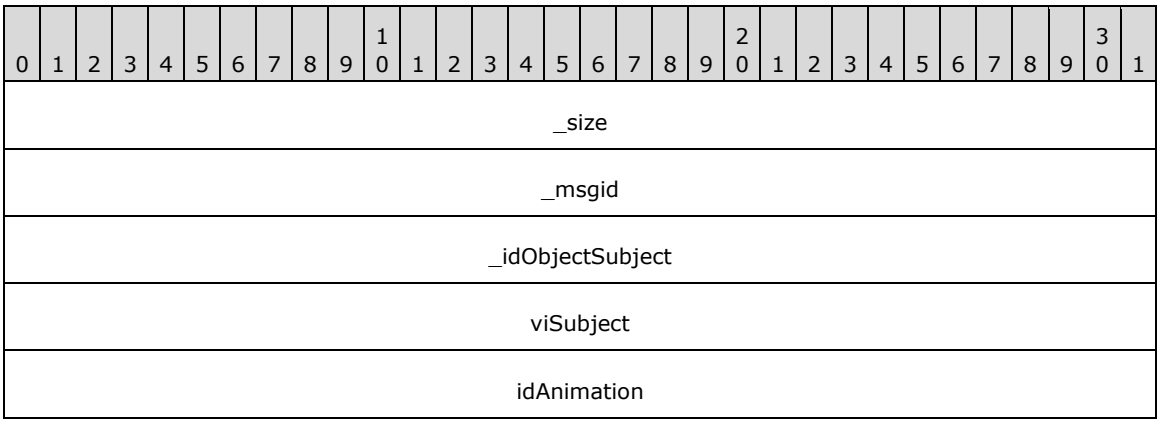
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.5 AnimationManager_BuildSizeAnimation

The AnimationManager_BuildSizeAnimation message builds an animation to modify the visual's size property.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

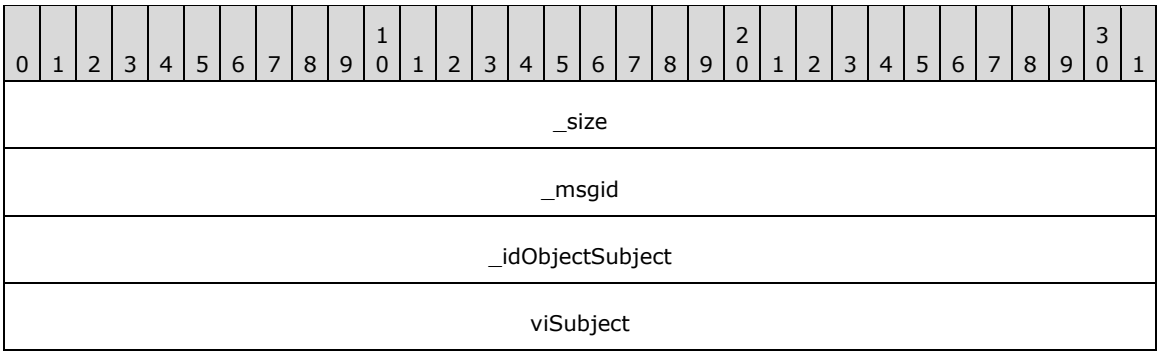
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.6 AnimationManager_BuildScaleAnimation

The AnimationManager_BuildScaleAnimation message builds an animation to modify the visual's scale property.



idAnimation

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.7 AnimationManager_BuildPositionAnimation

The AnimationManager_BuildPositionAnimation message builds an animation to modify the visual's position property.

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
_size																															
_msgid																															
_idObjectSubject																															
viSubject																															
idAnimation																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.8 AnimationManager_BuildColorAnimation

The AnimationManager_BuildColorAnimation message builds an animation to modify the visual's color property.

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
_size																															
_msgid																															
_idObjectSubject																															

viSubject
idAnimation

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.7.9 AnimationManager_BuildAlphaAnimation

The AnimationManager_BuildAlphaAnimation message builds an animation to modify the visual's alpha property.

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
_size																															
_msgid																															
_idObjectSubject																															
viSubject																															
idAnimation																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

viSubject (4 bytes): An unsigned 32-bit integer. The ID of the target visual.

idAnimation (4 bytes): An unsigned 32-bit integer. The ID to assign to the created animation.

2.2.4.8 WaitCursor

2.2.4.8.1 WaitCursor_Create

The WaitCursor_Create message builds a new instance of the WaitCursor for the given context.

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
_size																															

_msgid
_idObjectSubject

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.8.2 WaitCursor_Show

The WaitCursor_Show message starts the animations to show the wait cursor.

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
_size																															
_msgid																															
_idObjectSubject																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.8.3 WaitCursor_Hide

The WaitCursor_Hide message starts the animations to hide the wait cursor. Once the animations have completed, the owned visuals are hidden.

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
_size																															
_msgid																															
_idObjectSubject																															

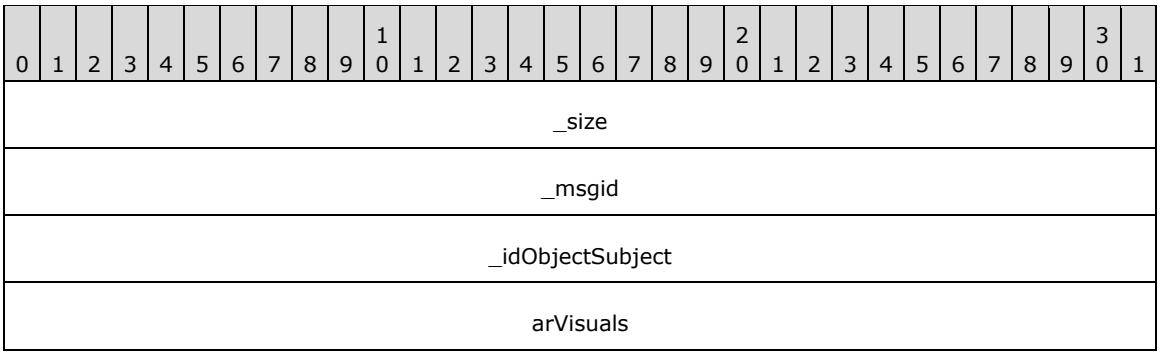
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.8.4 WaitCursor_SetVisuals

The WaitCursor_SetVisuals message sets the visuals being used to construct the wait cursor.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

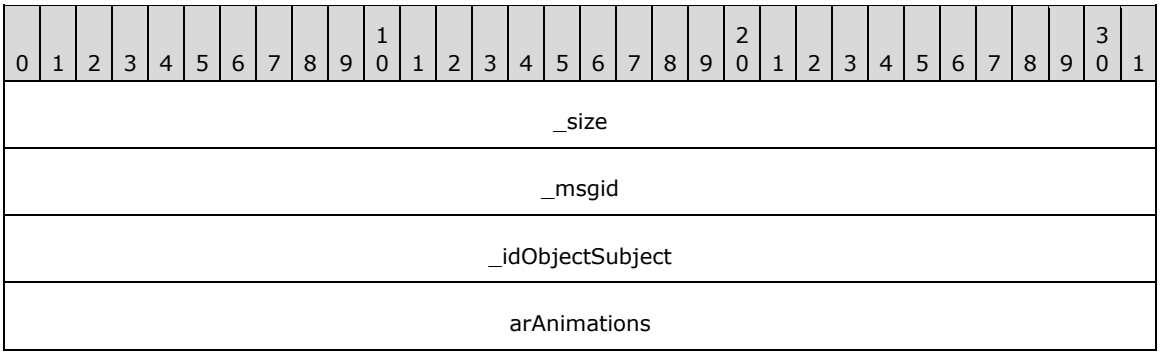
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

arVisuals (4 bytes): A BLOBREF (section 2.2.6.1) that specifies an array of visuals to use for the wait cursor.

2.2.4.8.5 WaitCursor_SetShowAnimations

The WaitCursor_SetShowAnimations message sets the animations to use to show the wait cursor.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

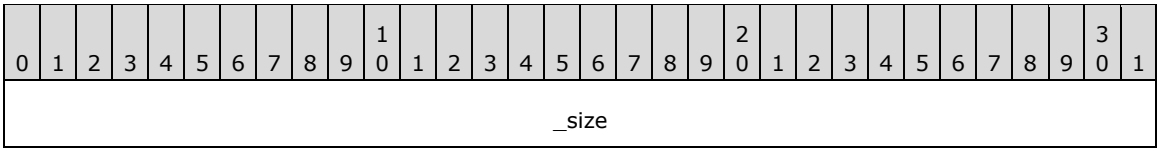
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

arAnimations (4 bytes): A BLOBREF (section 2.2.6.1) that specifies an array of animations to use for showing the wait cursor.

2.2.4.8.6 WaitCursor_SetHideAnimations

The WaitCursor_SetHideAnimations message sets the animations to use to hide the wait cursor.



_msgid
_idObjectSubject
arAnimations

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

arAnimations (4 bytes): A BLOBREF (section 2.2.6.1) that specifies an array of animations to use for hiding the wait cursor.

2.2.4.9 Device

2.2.4.9.1 Device_Stop

The Device_Stop message stops rendering the current generation on this device. Any time rendering has to stop, this count is increased. For rendering to continue, the application MUST restart the new generation, when ready, to allow the application to setup any state before it displays to the user.

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
_size																															
_msgid																															
_idObjectSubject																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.9.2 Device_Restart

The Device_Restart message restarts a previously stopped rendering generation.

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
_size																															
_msgid																															
_idObjectSubject																															

nRenderGeneration

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nRenderGeneration (4 bytes): An unsigned 32-bit integer. The render generation to restart.

2.2.4.9.3 Device_DrawLine

The Device_DrawLine message draws a line of the given color.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															
clrLine																															
flThickness																															
vStart																															
...																															
...																															
vEnd																															
...																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrLine (4 bytes): A Color (section 2.2.6.9) that specifies the color of the line.

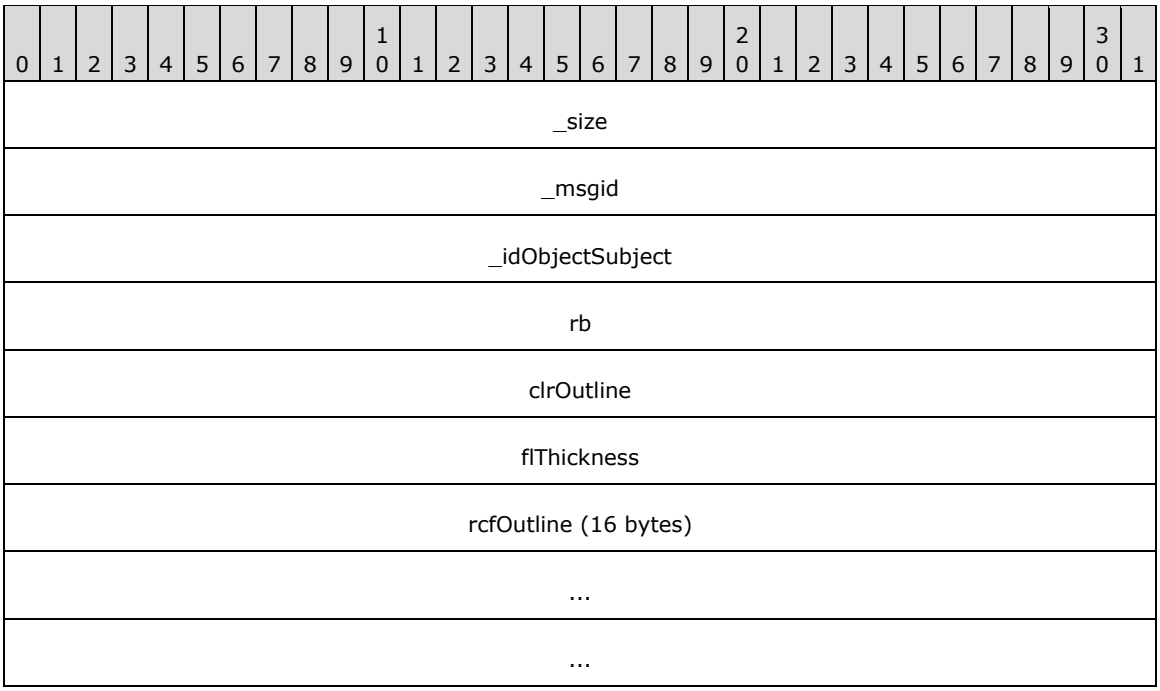
flThickness (4 bytes): A single-precision 32-bit number. The line thickness.

vStart (12 bytes): A Vector3 (section 2.2.6.3) that specifies the starting position of the line.

vEnd (12 bytes): A Vector3 (section 2.2.6.3) that specifies the end position of the line.

2.2.4.9.4 Device_DrawOutline

The Device_DrawOutline message draws a 1-pixel outline.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

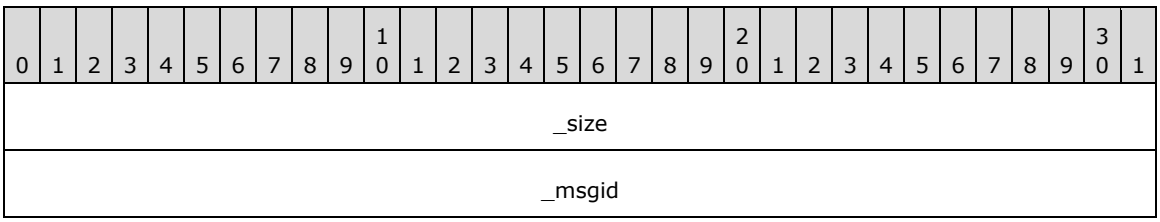
clrOutline (4 bytes): A Color (section 2.2.6.9) that specifies the color of the outline.

flThickness (4 bytes): A single-precision 32-bit number. The outline line thickness.

rcfOutline (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area to draw the outline around, in pixels.

2.2.4.9.5 Device_DrawSolid

The Device_DrawSolid message draws a solid rectangle of a given color.



_idObjectSubject
rb
clrFill
rcfFill (16 bytes)
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrFill (4 bytes): A Color (section 2.2.6.9) that specifies the color of the outline.

rcfFill (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area to draw the outline around, in pixels.

2.2.4.9.6 Device_CreateSurfacePool

The Device_CreateSurfacePool message requests that the device creates a new surface pool.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewSurface																															
sizeGutterPxl																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

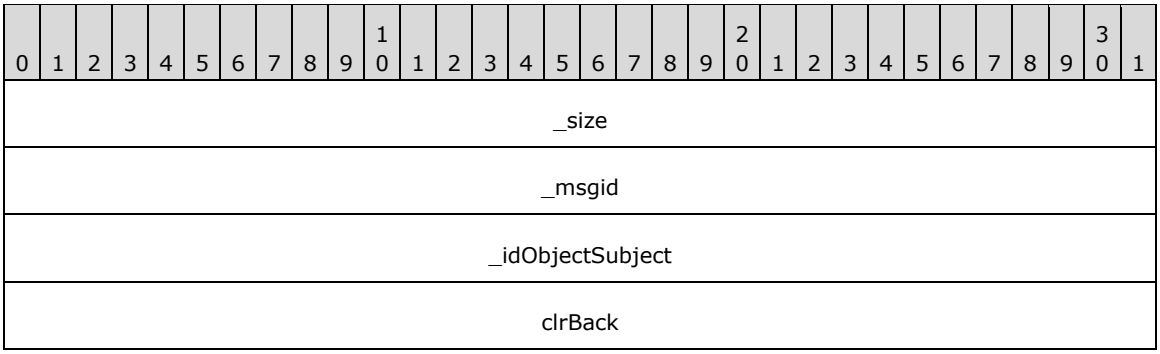
idNewSurface (4 bytes): An unsigned 32-bit integer. The ID to assign to the new surface pool.

sizeGutterPxl (8 bytes): A Size (section 2.2.6.6). The gutter around surfaces, in pixels.

2.2.4.10 Window

2.2.4.10.1 Window_SetBackgroundColor

The Window_SetBackgroundColor message changes the default background color for the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

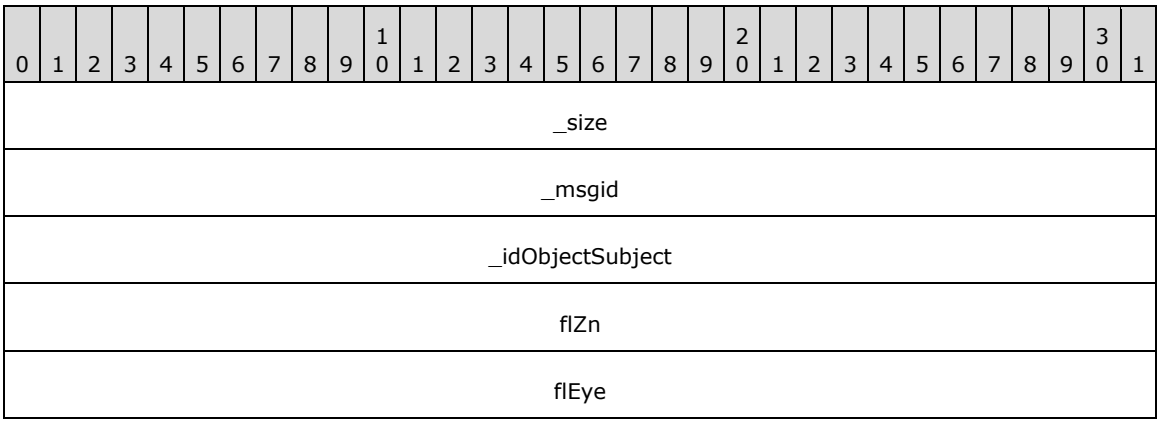
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clrBack (4 bytes): A Color (section 2.2.6.9) that specifies the color of the window's background.

2.2.4.10.2 Window_SetPerspectiveSettings

The Window_SetPerspectiveSettings message sets the viewing perspective of the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flZn (4 bytes): A single-precision 32-bit number. The distance of "eye" from "at" to the nearest plane.

flEye (4 bytes): A single-precision 32-bit number. The distance of "eye" from "at" to the furthest plane.

At: The center of the object you want to look "at".

Eye: The location of the eye (camera).

2.2.4.10.3 Window_ChangeDataBits

The Window_ChangeDataBits message changes the user-defined bits set on the window.

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
_size																															
_msgid																															
_idObjectSubject																															
nValue																															
nMask																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nValue (4 bytes): An unsigned 32-bit integer. The new value.

nMask (4 bytes): An unsigned 32-bit integer. A mask to use when changing the bits.

2.2.4.10.4 Window_SetContent

The Window_SetContent message copies the RenderOperations from the given RenderBuilder into the window.

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
_size																															
_msgid																															
_idObjectSubject																															
rbContent																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

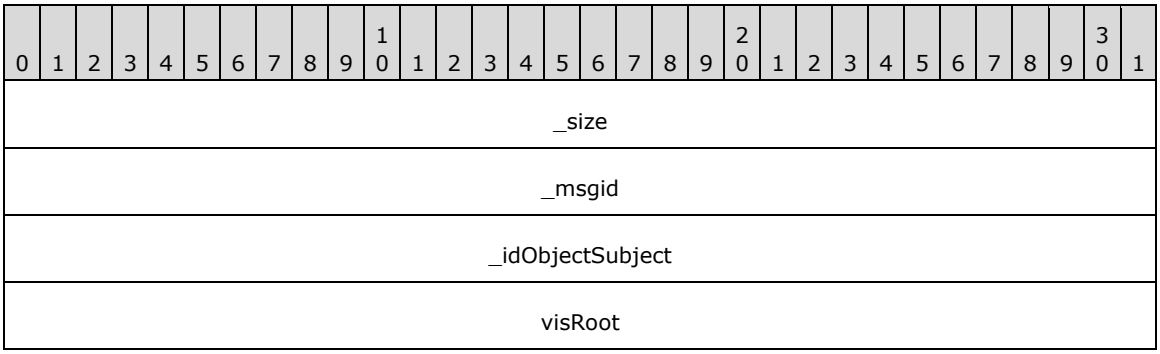
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rbContent (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder.

2.2.4.10.5 Window_SetRoot

The Window_SetRoot message changes the root visual associated with the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

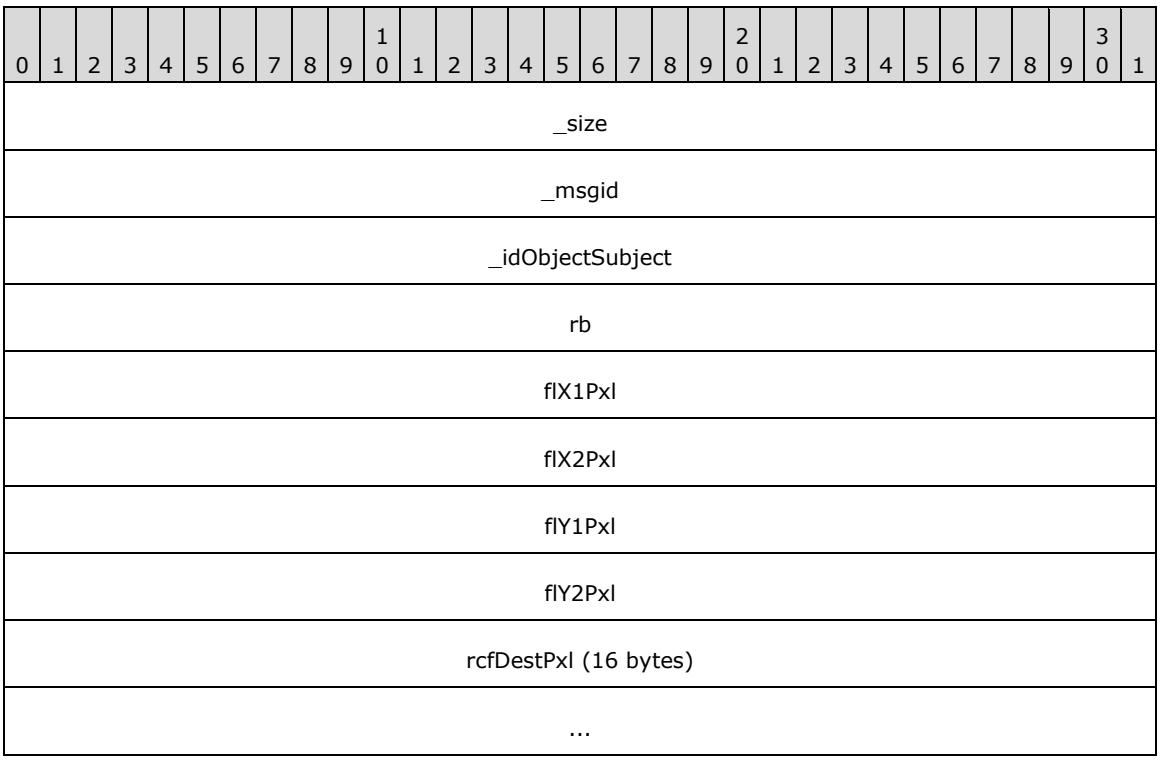
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

visRoot (4 bytes): An unsigned 32-bit integer. The ID of the new root visual.

2.2.4.11 Surface

2.2.4.11.1 Surface_DrawGrid

The Surface_DrawGrid message creates a RenderOperation to draw the surface in a grid.



...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

flX1Pxl (4 bytes): A single-precision 32-bit number. The left division, in pixels.

flX2Pxl (4 bytes): A single-precision 32-bit number. The right division, in pixels.

flY1Pxl (4 bytes): A single-precision 32-bit number. The top division, in pixels.

flY2Pxl (4 bytes): A single-precision 32-bit number. The bottom division, in pixels.

rcfDestPxl (16 bytes): A RectangleF (section 2.2.6.5). The user destination coordinates, in pixels.

2.2.4.11.2 Surface_Draw

The Surface_Draw message creates a RenderOperation to draw the surface.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															
rcfSrcPxl (16 bytes)																															
...																															
...																															
rcfDestPxl (16 bytes)																															
...																															
...																															
fNeverStretch																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

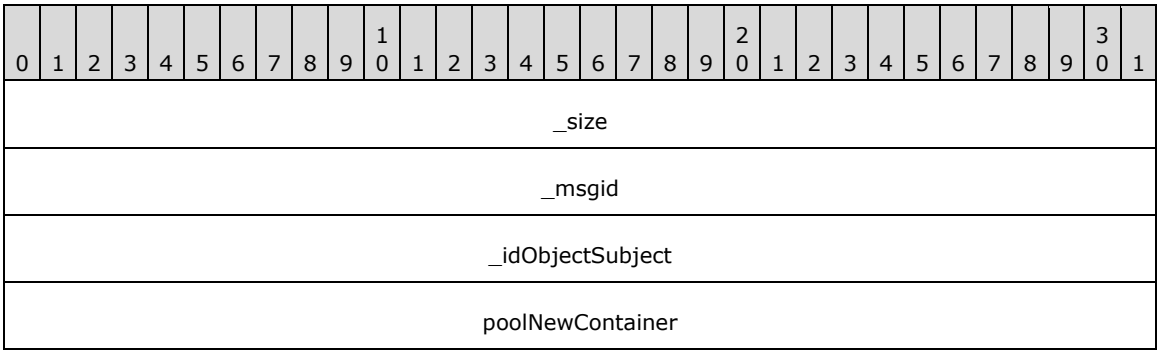
rcfSrcPxl (16 bytes): A RectangleF (section 2.2.6.5). The user source coordinates, in pixels.

rcfDestPxl (16 bytes): A RectangleF (section 2.2.6.5). The user destination coordinates, in pixels.

fNeverStretch (4 bytes): An unsigned 32-bit integer. This value MUST always be false.

2.2.4.11.3 Surface_RemapContainer

The Surface_RemapContainer message changes the container of the surface. The underlying content is not transferred. The current configuration of the surface is not changed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

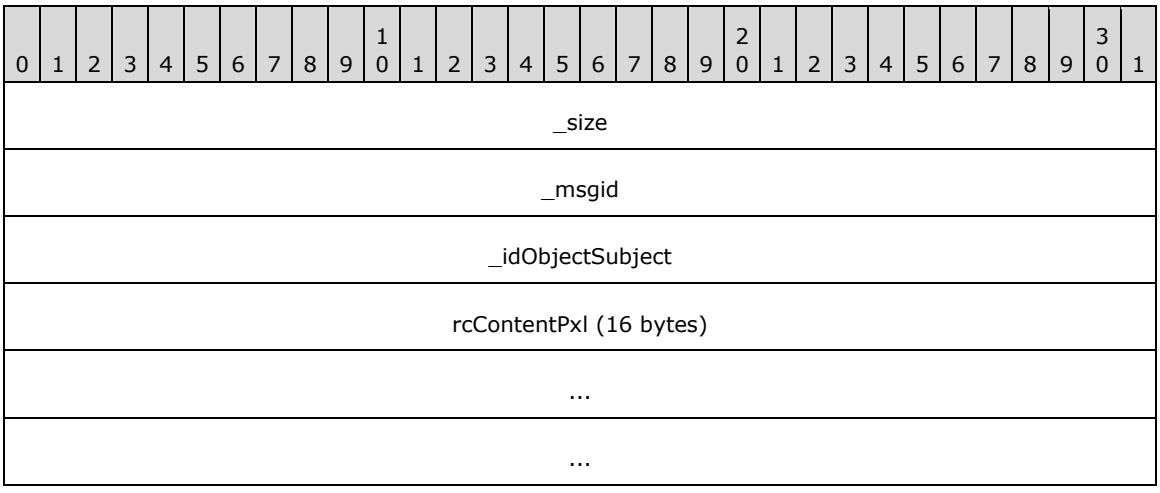
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

poolNewContainer (4 bytes): An unsigned 32-bit integer. The ID of the container SurfacePool.

2.2.4.11.4 Surface_RemapLocation

The Surface_RemapLocation message changes the requested location of the surface from the upper-left corner within the pool. The underlying content is not moved.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

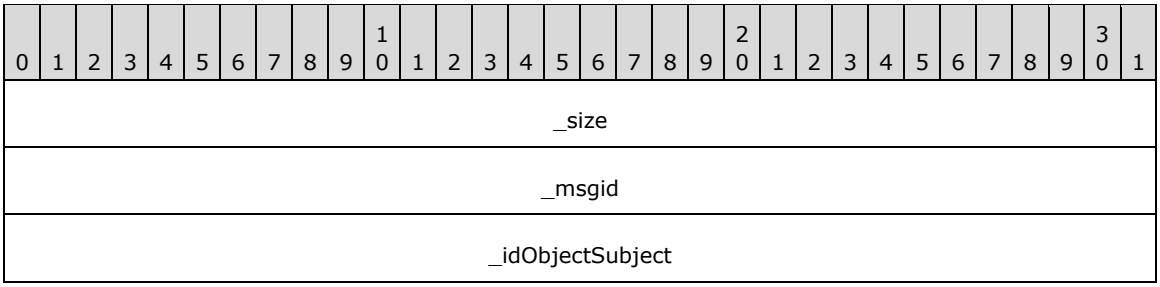
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rcContentPxl (16 bytes): A Rectangle (section 2.2.6.4). The new location within the pool, in pixels.

2.2.4.11.5 Surface_MarkContentValid

The Surface_MarkcontentValid message marks this surface as having valid content. This message enables an application to use a surface for drawing after setting the SurfacePool's underlying surface. This function can be used very carefully as it marks the content as valid, regardless of whether valid content has actually been set.



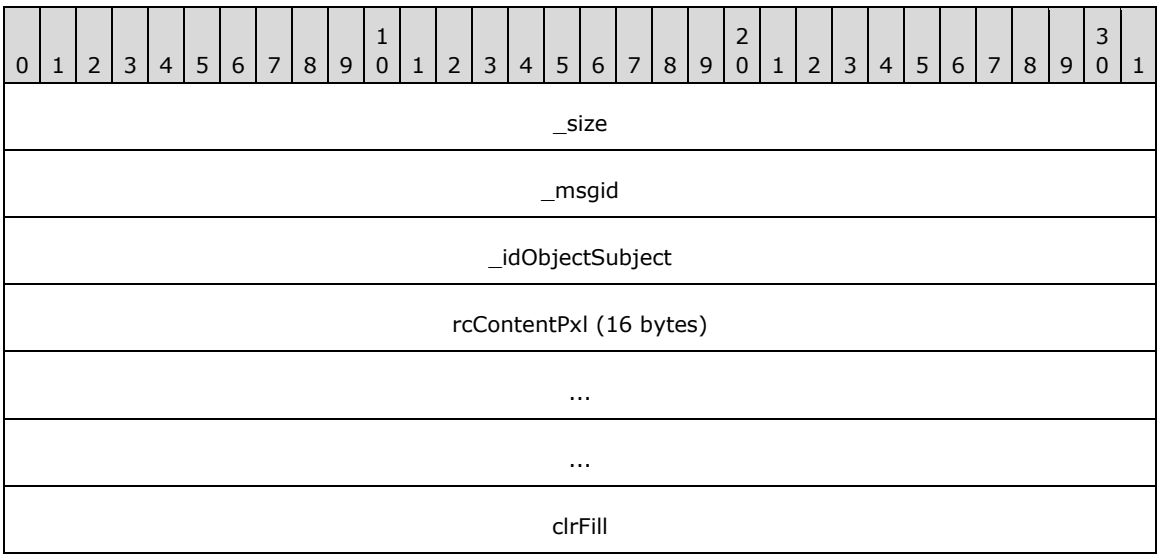
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.11.6 Surface_Clear

The Surface_Clear message empties the content of the surface, but does not change the surface's location within the SurfacePool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

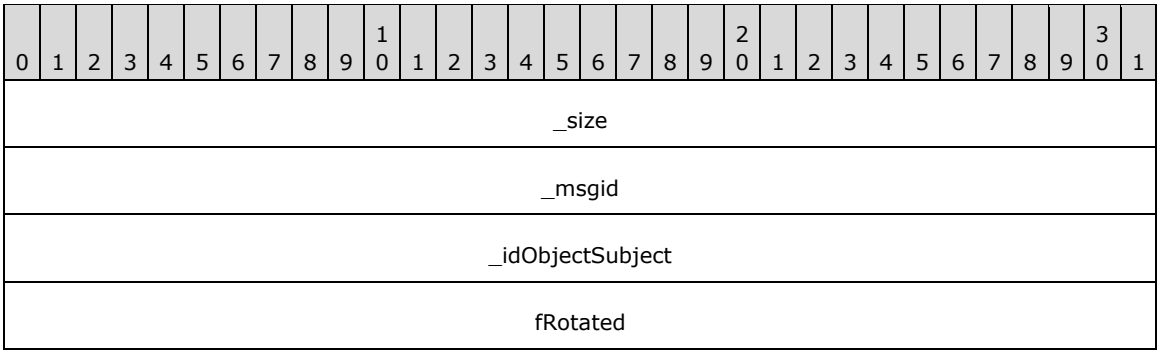
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rcContentPxl (16 bytes): A Rectangle (section 2.2.6.4). The area to clear. Use an empty area rectangle to clear the entire surface.

clrFill (4 bytes): A Color (section 2.2.6.9). The color to which to clear the rectangle.

2.2.4.11.7 Surface_SetRotation

The Surface_SetRotation message changes when the contents of the surface are rotated 90 degrees to produce a more compact representation. After changing the rotation, any content MUST be reloaded into the surface.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

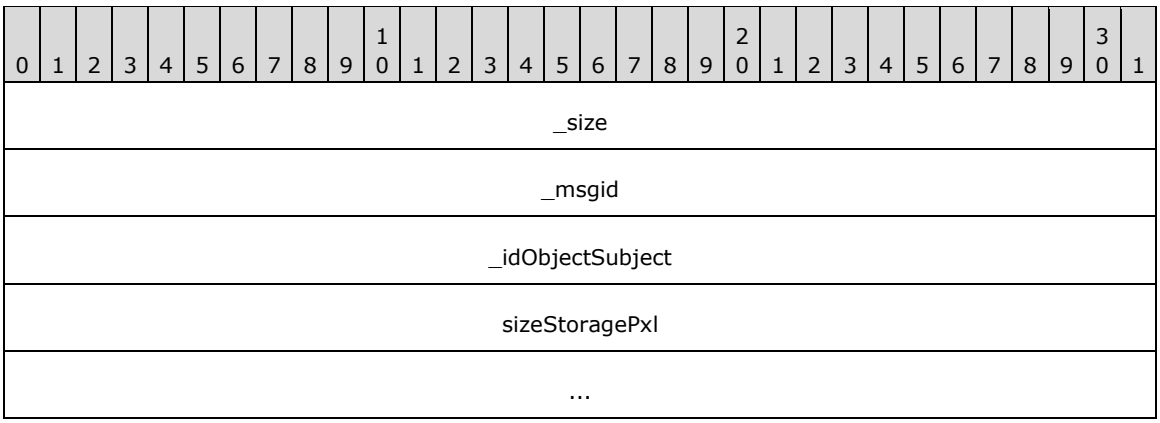
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

fRotated (4 bytes): An unsigned 32-bit integer. Indicates whether or not to rotate the surface.

2.2.4.11.8 Surface_SetStorageSize

The Surface_SetStorageSize message changes the requested physical size of the surface within the pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

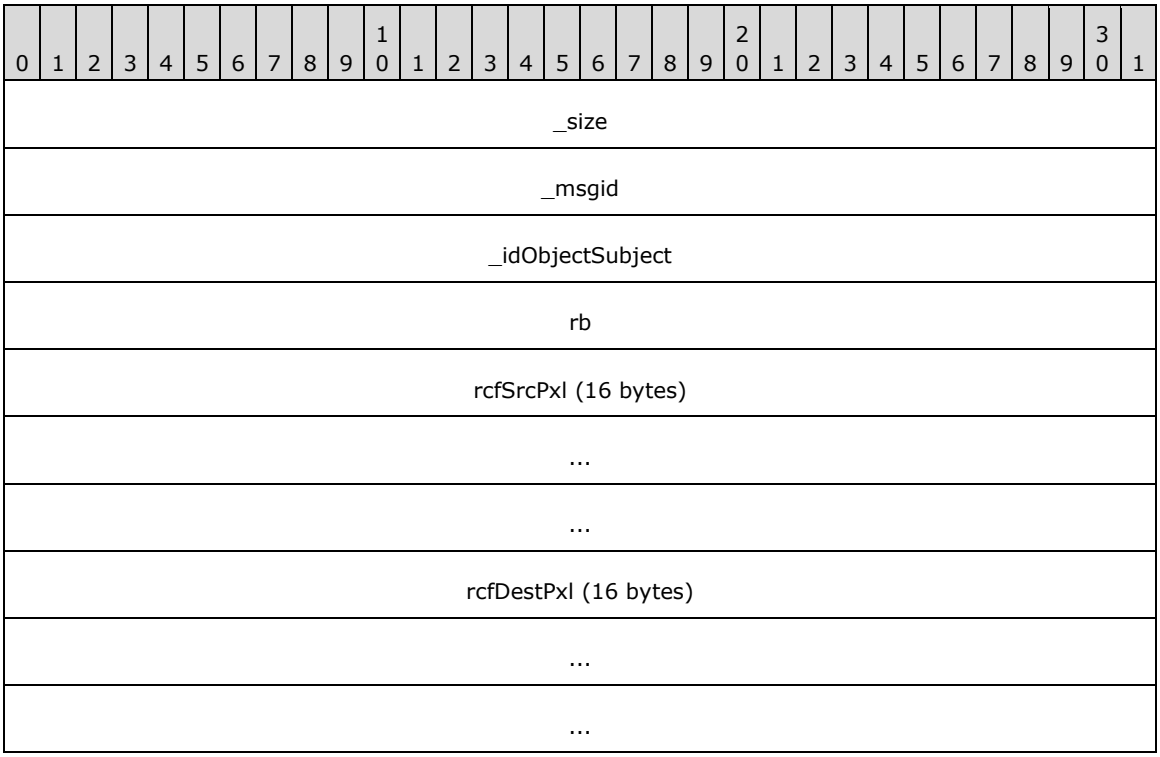
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

sizeStoragePxl (8 bytes): A Size (section 2.2.6.6). The size of the requested area, in pixels.

2.2.4.12 SurfacePool

2.2.4.12.1 SurfacePool_Draw

The SurfacePool_Draw message creates a RenderOperation to draw the surface pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

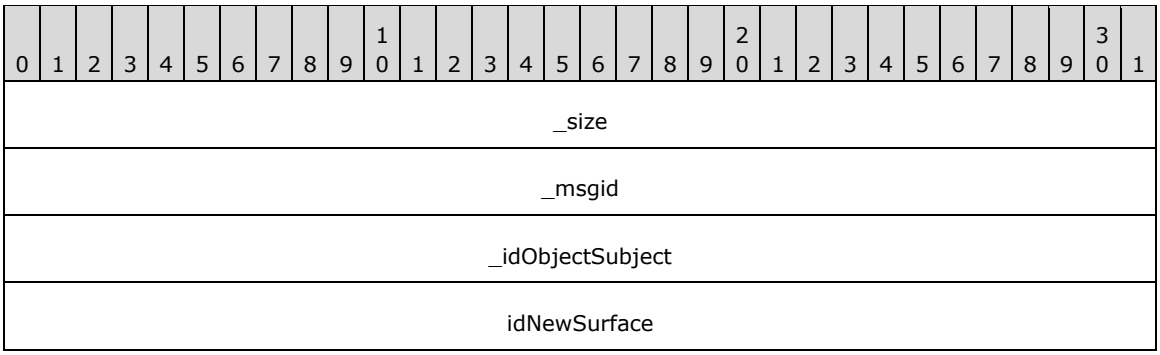
rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

rcfSrcPxl (16 bytes): A RectangleF (section 2.2.6.5). The user source coordinates, in pixels.

rcfDestPxl (16 bytes): A RectangleF (section 2.2.6.5). The user destination coordinates, in pixels.

2.2.4.12.2 SurfacePool_CreateSurface

The SurfacePool_CreateSurface message requests a new surface to be created in the pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

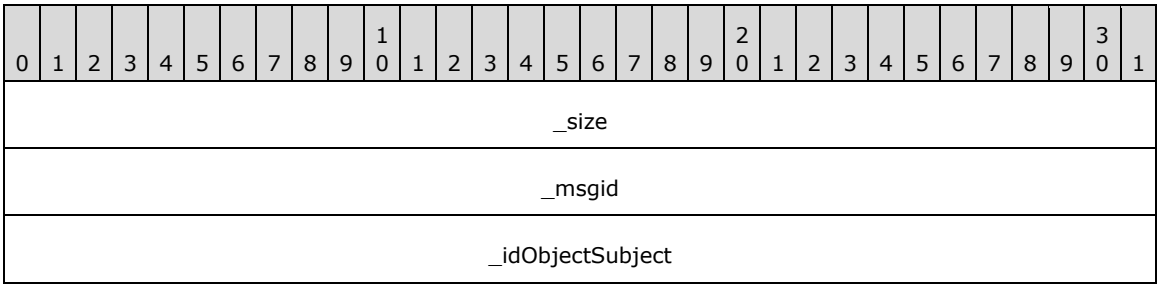
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID of the new surface to be created.

2.2.4.12.3 SurfacePool_Free

The SurfacePool_Free message releases any previously allocated or attached surfaces.



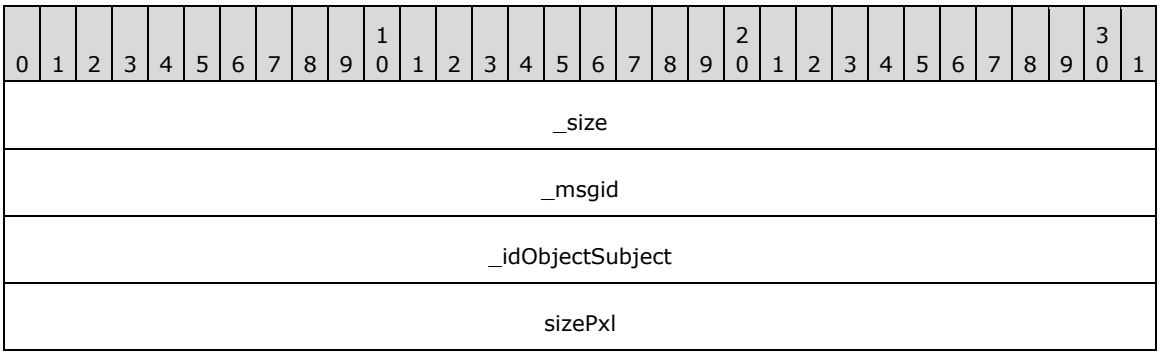
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.12.4 SurfacePool_Allocate

The SurfacePool_Allocate message allocates an underlying surface to store content.



...
nOptions

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

sizePxl (8 bytes): A Size (section 2.2.6.6). The size of the surface, in pixels.

nOptions (4 bytes): An unsigned 32-bit integer. The pixel format of the surface. Possible values are listed in the following table.

Value	Description
0	None
0x00200000	Bpp32
0x00180000	Bpp24
0x00100000	Bpp16
0x00080000	Bpp8
0x00208888	ARGB32
0x00200888	RGB32
0x00180888	RGB24
0x00101555	ARGB16-1555
0x00100555	RGB16-555
0x00100565	RGB16-565
0x21100000	YUY2
0x00080008	L8

2.2.4.12.5 SurfacePool_SetEmptyColor

The SurfacePool_SetEmptyColor message changes the color to use to draw the surface when no storage is allocated.

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
_size																															
_msgid																															

_idObjectSubject
clrFill

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clrFill (4 bytes): A Color (section 2.2.6.9). The color to use for the surface.

2.2.4.12.6 SurfacePool_SetPriority

The SurfacePool_SetPriority message changes the current priority level for this object, relative to its peers. A lower number indicates a higher priority. The default priority level is 0.

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
_size																															
_msgid																															
_idObjectSubject																															
nPriority																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nPriority (4 bytes): A signed 32-bit integer. The new priority level.

2.2.4.13 VideoPool

2.2.4.13.1 VideoPool_Draw

The VideoPool_Draw message creates a RenderOperation to draw the VideoPool.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															

rcfSrcPxl (16 bytes)
...
...
rcfDestPxl (16 bytes)
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

rcfSrcPxl (16 bytes): A RectangleF (section 2.2.6.5). The user source coordinates, in pixels.

rcfDestPxl (16 bytes): A RectangleF (section 2.2.6.5). The user destination coordinates, in pixels.

2.2.4.13.2 VideoPool_CreateSurface

The VideoPool_CreateSurface message requests a new surface to be created in the pool.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewSurface																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

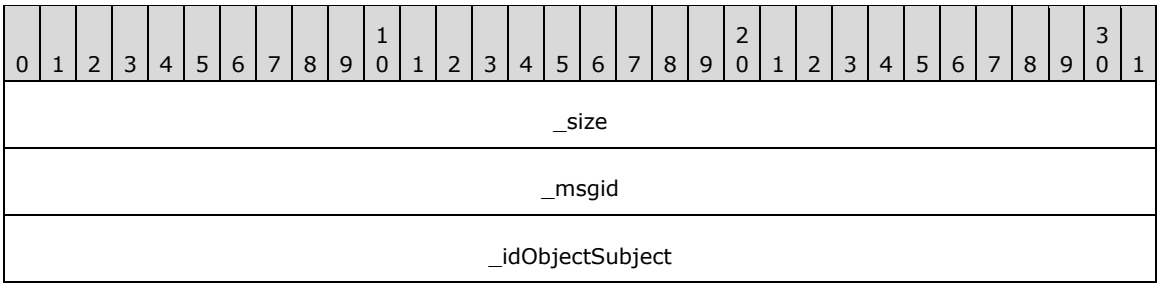
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID of the new surface to be created.

2.2.4.13.3 VideoPool_Free

The VideoPool_Free message releases any previously allocated or attached surfaces.



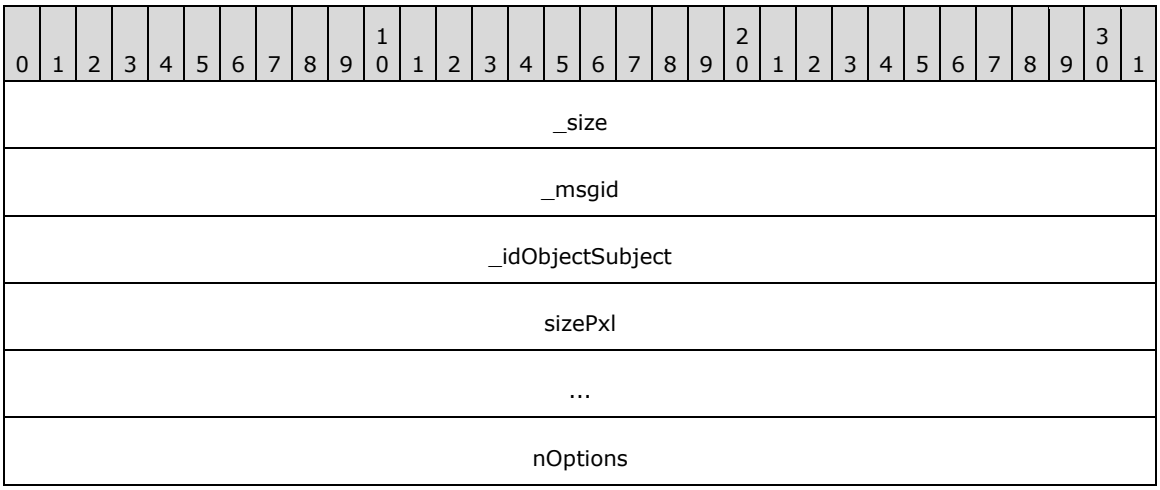
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.13.4 VideoPool_Allocate

The VideoPool_Allocate message allocates an underlying surface to store content.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

sizePxl (8 bytes): A Size (section 2.2.6.6). The size of the surface, in pixels.

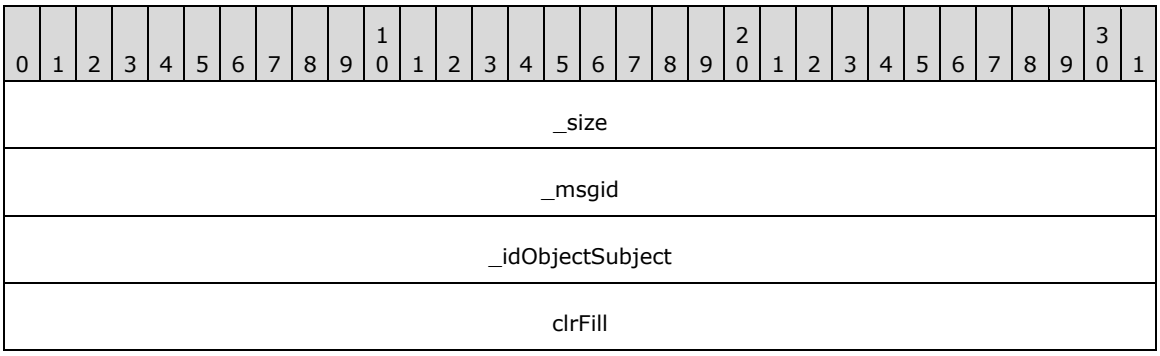
nOptions (4 bytes): An unsigned 32-bit integer. The pixel format of the surface.

Value	Description
0	Any - Any position amongst its siblings.
1	Before - Before the specified sibling.
2	Behind - Behind the specified sibling.
3	Top - The top of the parent's children list.

Value	Description
4	Bottom - The bottom of the parent's children list.

2.2.4.13.5 VideoPool_SetEmptyColor

The VideoPool_SetEmptyColor message changes the color to use to draw the surface when no storage is allocated.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

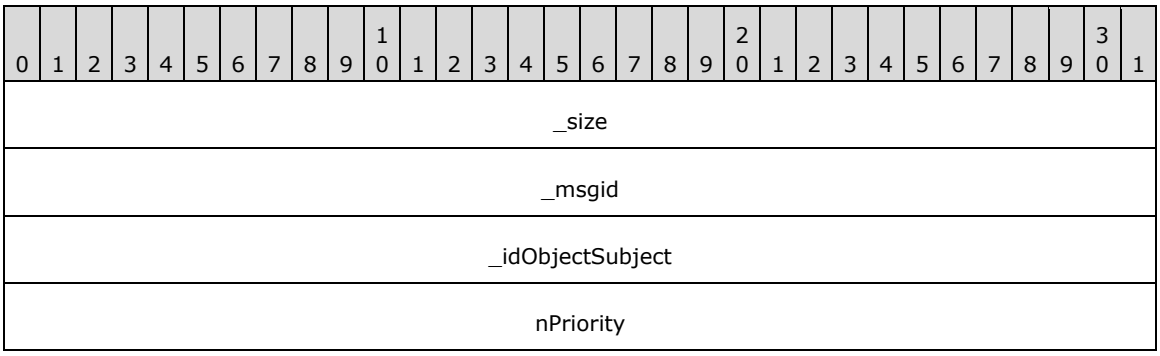
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clrFill (4 bytes): A Color (section 2.2.6.9). The color to use for the surface.

2.2.4.13.6 VideoPool_SetPriority

The VideoPool_SetPriority message changes the current priority level for this object, relative to its peers. A lower number indicates a higher priority. The default priority level is 0.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

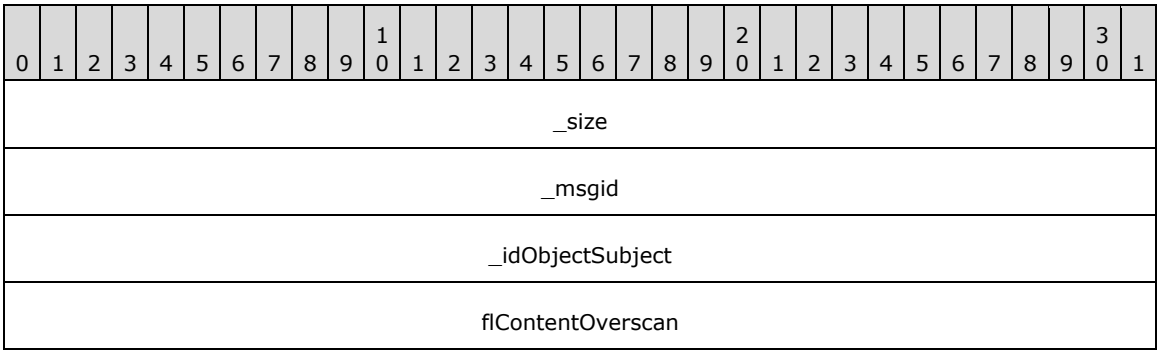
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nPriority (4 bytes): A signed 32-bit integer. The new priority level.

2.2.4.13.7 VideoPool_SetContentOverscan

The VideoPool_SetContentOverscan message sets the content overscan area for this video pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

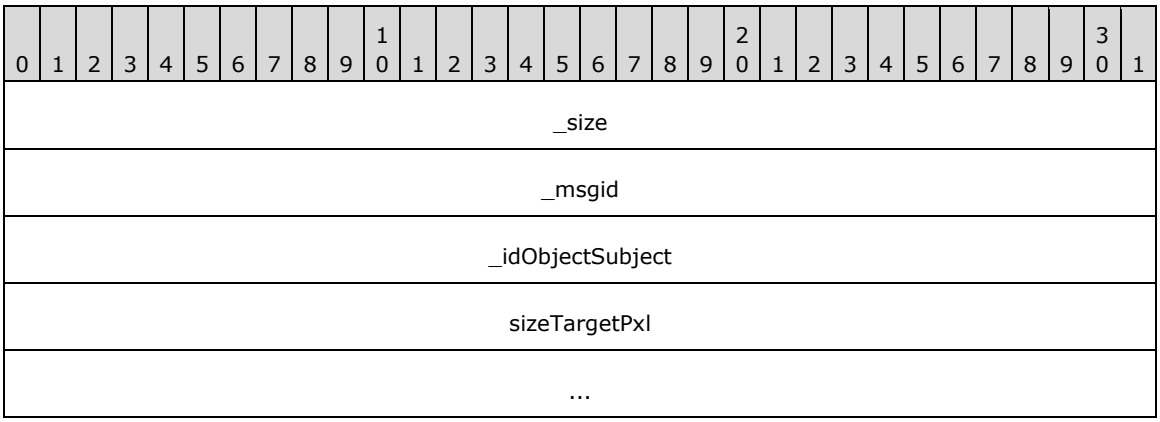
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flContentOverscan (4 bytes): A single-precision 32-bit number. The content overscan percentage.

2.2.4.13.8 VideoPool_NotifyVideoSizeChanged

The VideoPool_NotifyVideoSizeChanged message notifies the pool when the video size has changed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

sizeTargetPxl (8 bytes): A Size (section 2.2.6.6). The new video dimensions, in pixels.

2.2.4.14 Rasterizer

2.2.4.14.1 Rasterizer_LoadRawImage

The Rasterizer_LoadRawImage message loads a 32-bit raw image from the specified buffer.

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
_size																															
_msgid																															
_idObjectSubject																															
surContent																															
buffer																															
info (24 bytes)																															
...																															
...																															
offset																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

surContent (4 bytes): An unsigned 32-bit integer. The ID of the surface in which to store the content.

buffer (4 bytes): An unsigned 32-bit integer. The ID of the buffer.

info (24 bytes): An ImageHeader (section 2.2.6.7). The image information.

offset (8 bytes): A Point (section 2.2.6.8). The offset within the surface.

2.2.4.15 Gradient

2.2.4.15.1 Gradient_Pop

The Gradient_Pop message pops the gradient out of effect.

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
_size																															
_msgid																															
_idObjectSubject																															

rb

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

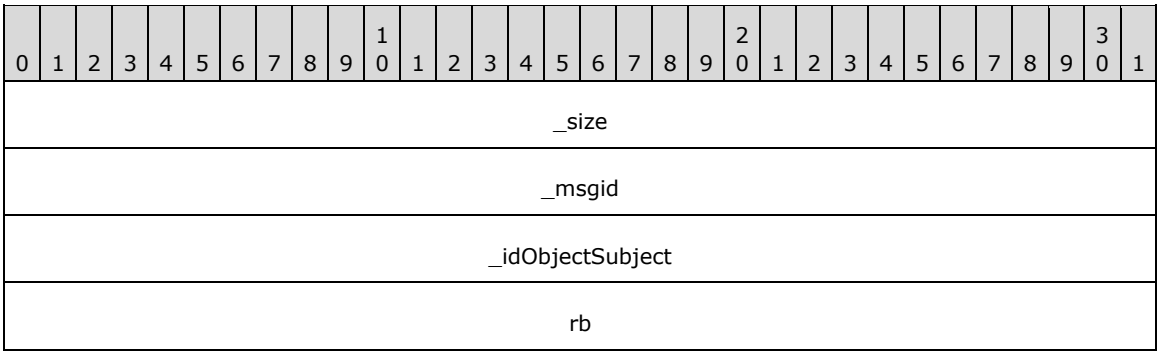
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

2.2.4.15.2 Gradient_Push

The Gradient_Push message pushes the gradient into effect.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

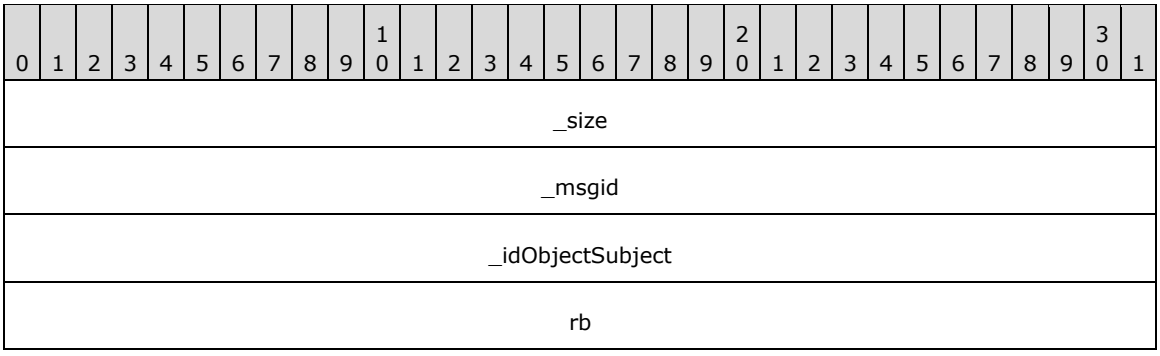
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

2.2.4.15.3 Gradient_Draw

The Gradient_Draw message signals that the gradient can be put into effect during the next render operation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

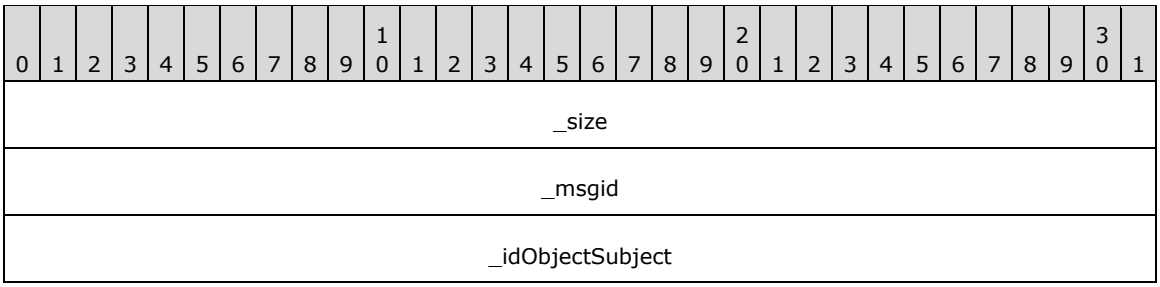
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The **_msgid** value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

2.2.4.15.4 Gradient_Clear

The Gradient_Clear message removes all values from this gradient's ramp.



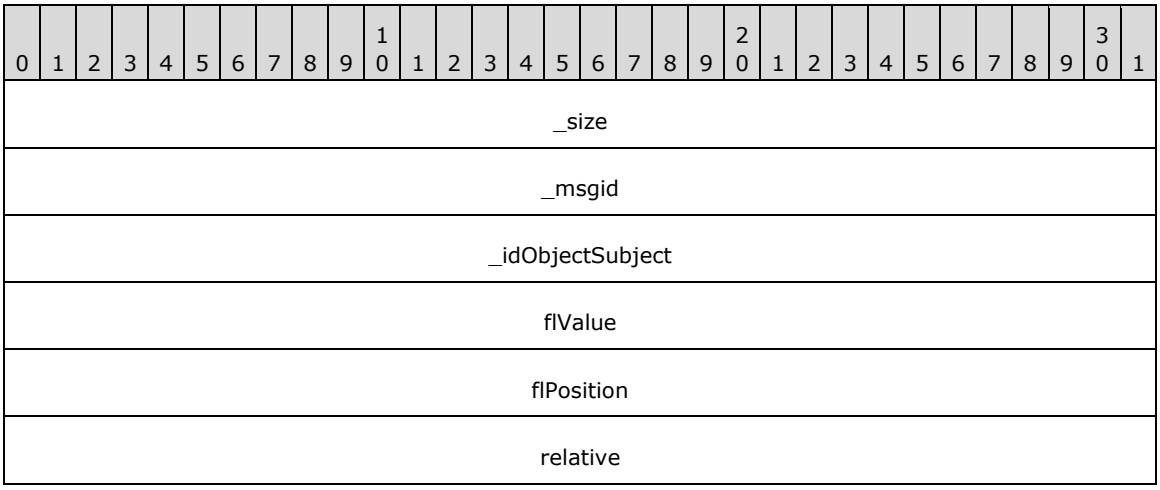
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The **_msgid** value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.15.5 Gradient_AddValue

The Gradient_AddValue message adds a value to the ramp. The position is interpreted differently depending on the orientation of the gradient and offset based on the relative value of the value.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flValue (4 bytes): A single-precision 32-bit number. The value of the gradient stop.

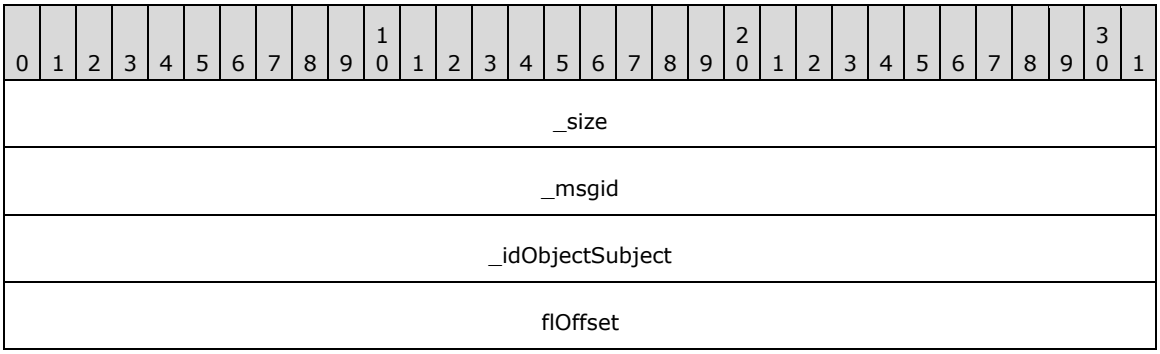
flPosition (4 bytes): A single-precision 32-bit number. The position along the gradient ramp.

relative (4 bytes): A signed 32-bit integer. The relative space of the position value. Possible values are described in the following table.

Value	Description
0	The visual's logical rectangle min.
1	The visual's logical rectangle max.
2	The mesh's min extent.
3	The mesh's max extent.
4	Global space.

2.2.4.15.6 Gradient_SetOffset

The Gradient_SetOffset message sets the offset of the gradient.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

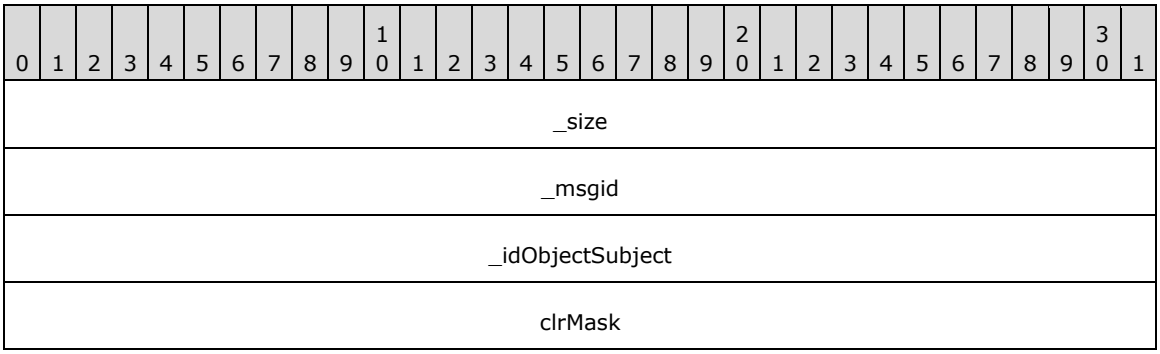
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flOffset (4 bytes): A single-precision 32-bit number. Offset value.

2.2.4.15.7 Gradient_SetColorMask

The Gradient_SetColorMask message sets the color mask that the gradient uses when applying the specified values.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

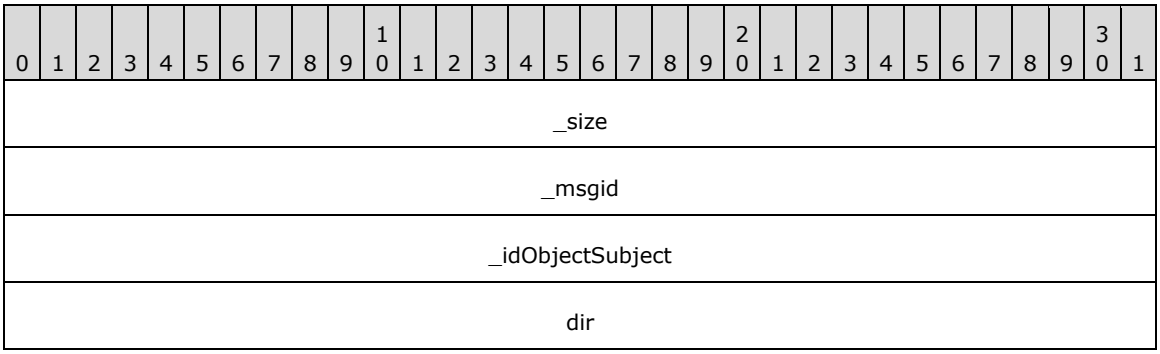
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clrMask (4 bytes): A Color (section 2.2.6.9). The color mask.

2.2.4.15.8 Gradient_SetOrientation

The Gradient_SetOrientation message sets the orientation of the gradient coordinates.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

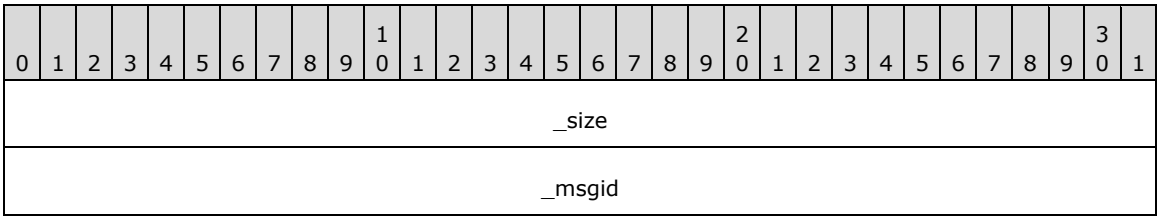
dir (4 bytes): A signed 32-bit integer. Specifies whether the gradient runs horizontally or vertically. Possible values are described in the following table.

Value	Description
0	Horizontal
1	Vertical

2.2.4.16 Line

2.2.4.16.1 Line_SetThickness

The Line_SetThickness message sets the thickness of the line.



_idObjectSubject
flThickness

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flThickness (4 bytes): A single-precision 32-bit number. The line thickness.

2.2.4.16.2 Line_SetColor

The Line_SetColor message sets the color of the line.

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
_size																															
_msgid																															
_idObjectSubject																															
clr																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clr (4 bytes): A Color (section 2.2.6.9). The color to use.

2.2.4.16.3 Line_CommitLine

The Line_CommitLine message draws the line.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

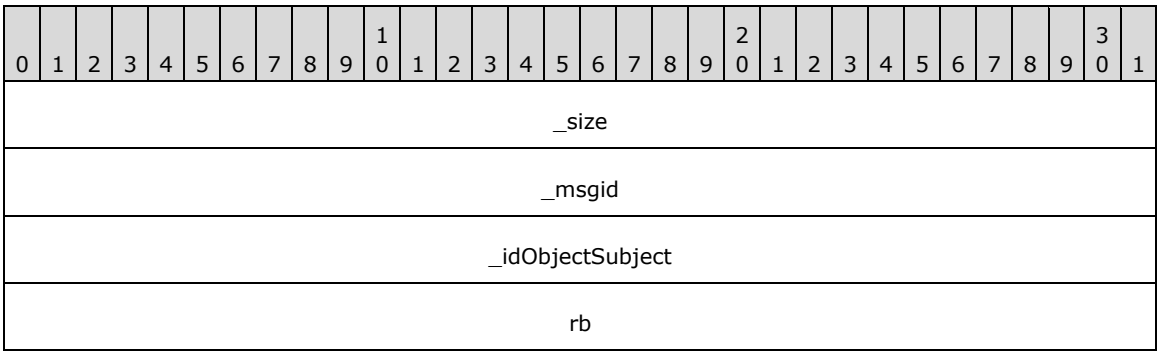
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

2.2.4.16.4 Line_DrawPoint

The Line_DrawPoint message draws a point of the line.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

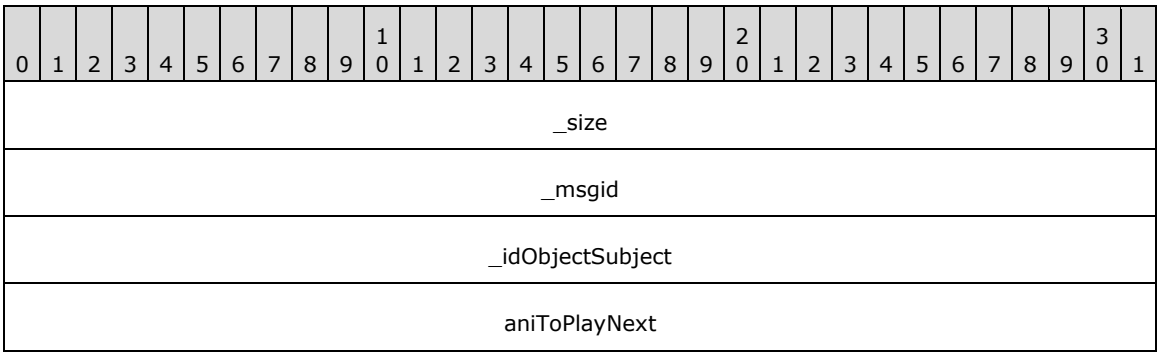
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

2.2.4.17 Animation

2.2.4.17.1 Animation_AddCompletionLink

The Animation_AddCompletionLink message arranges for an animation to be auto-played as the result of another animation completing normally. This message is useful for logically separate sequences that have to run into each other without the client application actively monitoring the playback. For example, the application can display a short "intro" animation that leads into a repeating animation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

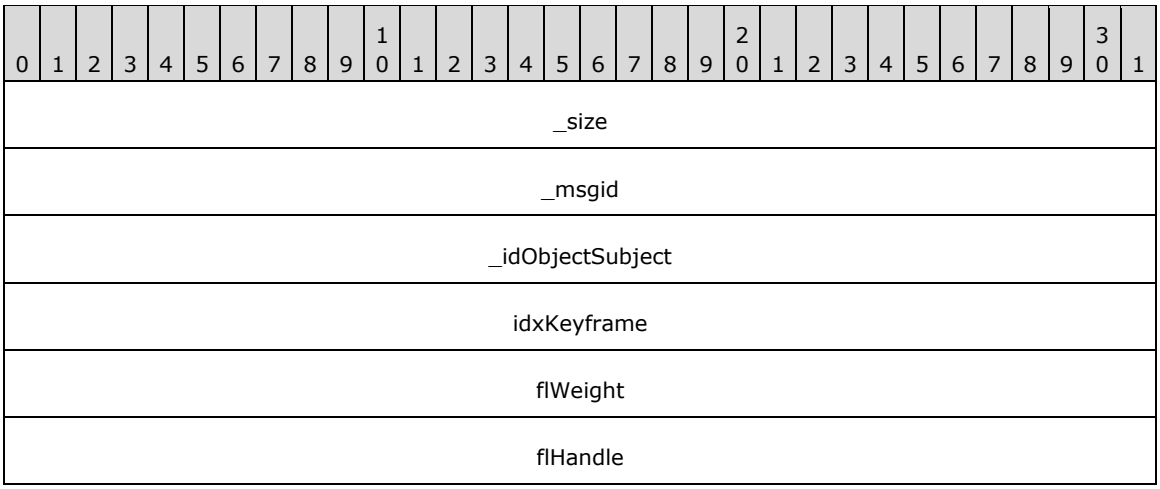
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

aniToPlayNext (4 bytes): An unsigned 32-bit integer. The ID of the animation to play next.

2.2.4.17.2 Animation_SetEaseOut

The Animation_SetEaseOut message changes the given keyframes across all sequences in the animation to an Ease Out interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

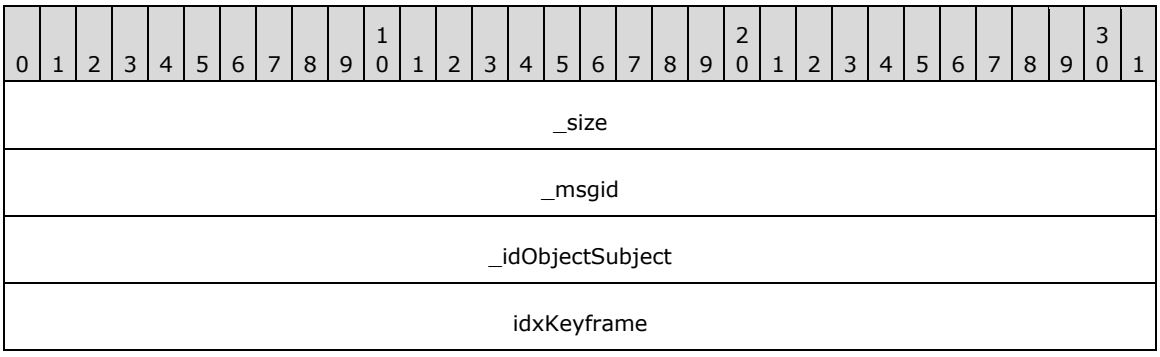
idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flWeight (4 bytes): A single-precision 32-bit number. The weight of the interpolation as compared to a linear interpolation.

flHandle (4 bytes): A single-precision 32-bit number. The percentage of progress where the interpolation changes from exponential to linear. This value is between 0.0 and 1.0 (non-inclusive).

2.2.4.17.3 Animation_SetEaseIn

The Animation_SetEaseIn message changes the given keyframes across all sequences in the animation to use an Ease In interpolation.



flWeight
flHandle

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flWeight (4 bytes): A single-precision 32-bit number. The weight of the interpolation as compared to a linear interpolation.

flHandle (4 bytes): A single-precision 32-bit number. The percentage of progress where the interpolation changes from linear to logarithmic. This value is between 0.0 and 1.0 (non-inclusive).

2.2.4.17.4 Animation_SetBezier

The Animation_SetBezier message changes the given keyframes across all sequences in the animation to use a Bezier interpolation.

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
_size																															
_msgid																															
_idObjectSubject																															
idxKeyframe																															
flHandle1																															
flHandle2																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

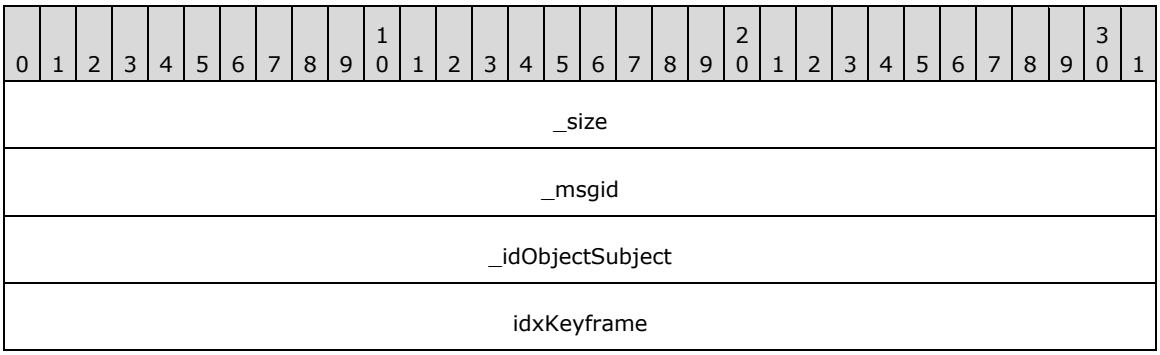
idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flHandle1 (4 bytes): A single-precision 32-bit number. The first control handle for the Bezier curve.

flHandle2 (4 bytes): A single-precision 32-bit number. The second control handle for the Bezier curve.

2.2.4.17.5 Animation_SetCosine

The Animation_SetCosine message changes the given keyframes across all sequences in the animation to use a cosine interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

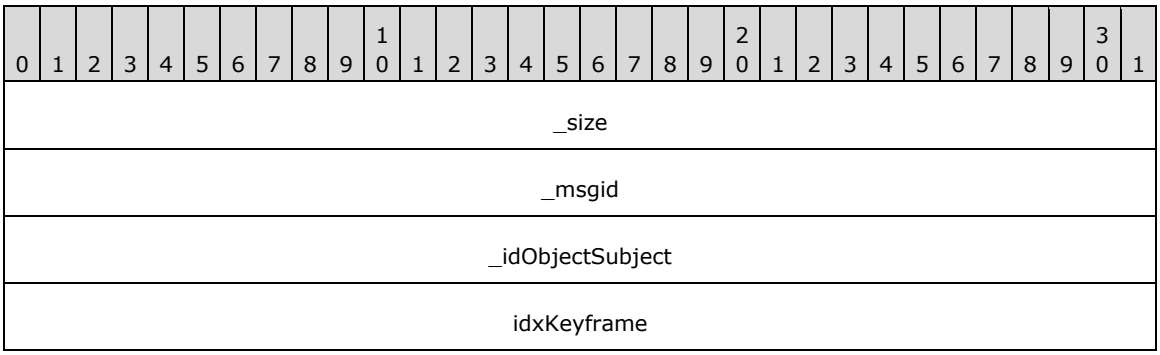
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

2.2.4.17.6 Animation_SetSine

The Animation_SetSine message changes the given keyframes across all sequences in the animation to use a sine interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

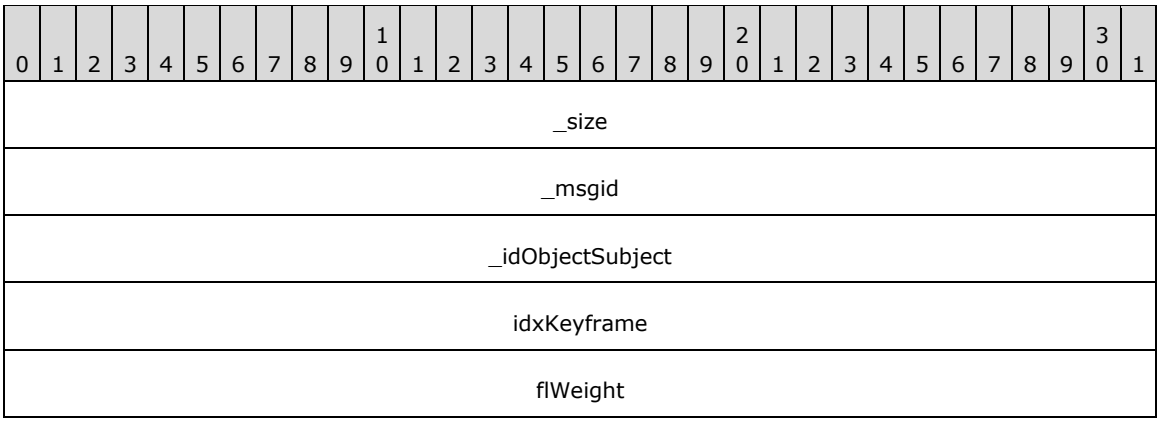
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

2.2.4.17.7 Animation_SetSCurve

The Animation_SetSCurve message changes the given keyframes across all sequences in the animation to use an S-curve interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

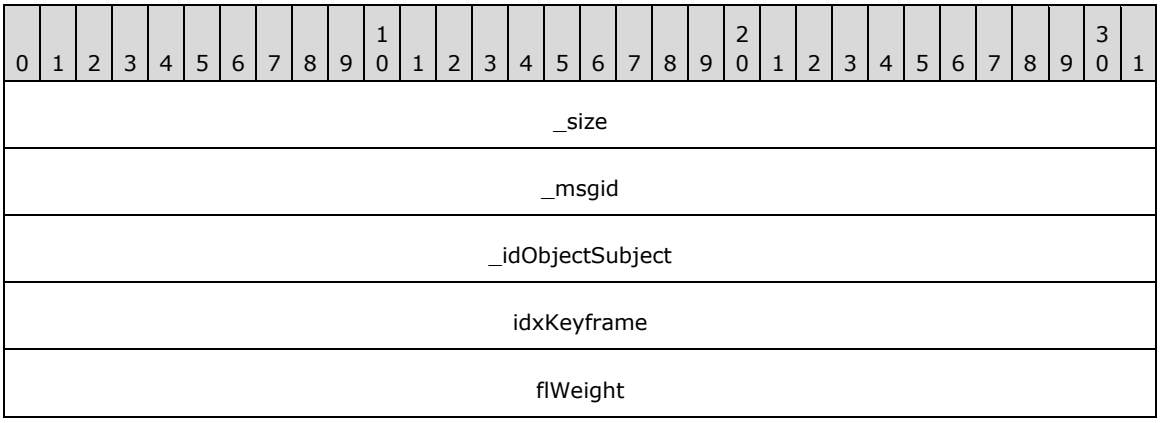
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flWeight (4 bytes): A single-precision 32-bit number. The weight of the interpolation as compared to a linear interpolation.

2.2.4.17.8 Animation_SetLogarithmic

The Animation_SetLogarithmic message changes the given keyframes across all sequences in the animation to use a logarithmic interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

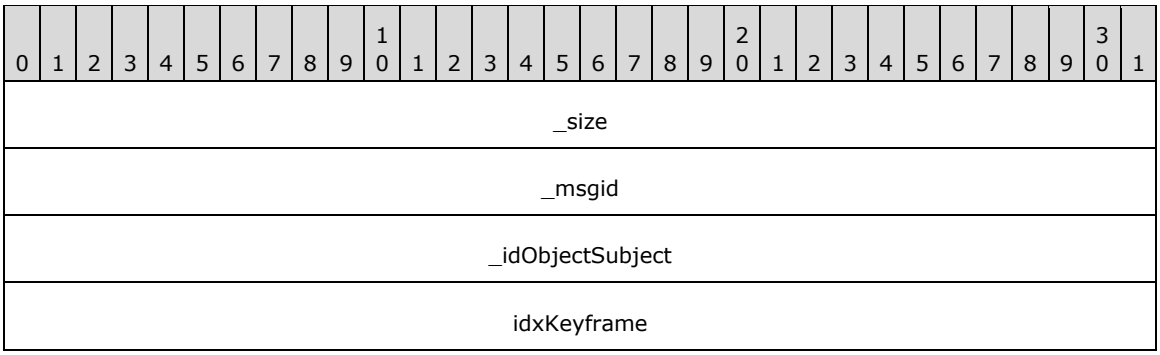
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flWeight (4 bytes): A single-precision 32-bit number. The weight of the interpolation as compared to a linear interpolation.

2.2.4.17.9 Animation_SetLinear

The Animation_SetLinear message changes the given keyframes across all sequences in the animation to use a linear interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

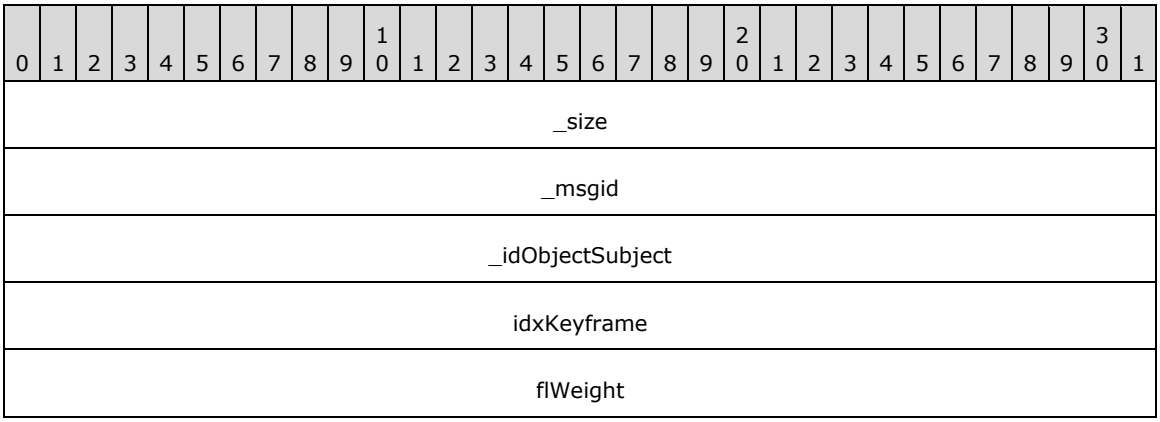
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

2.2.4.17.10 Animation_SetExponential

The Animation_SetExponential message changes the given keyframes across all sequences in the animation to use an exponential interpolation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

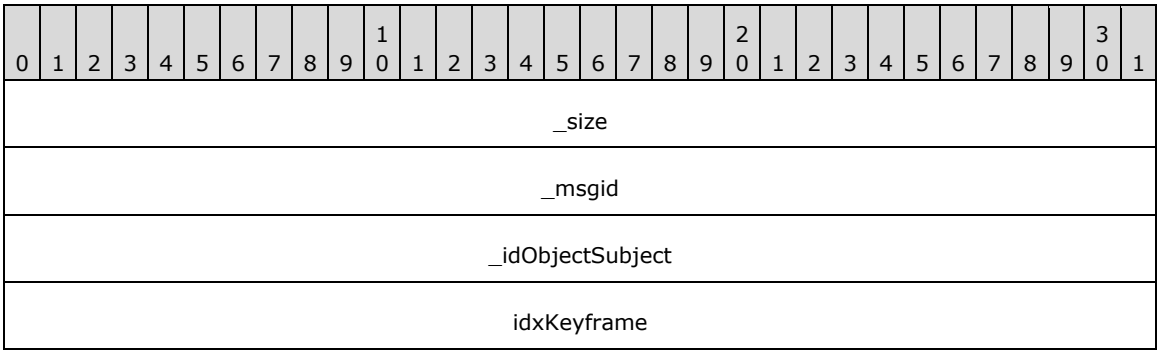
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flWeight (4 bytes): A single-precision 32-bit number. The weight of the interpolation as compared to a linear interpolation.

2.2.4.17.11 Animation_SetDynamicRotation

The Animation_SetDynamicRotation message creates a new DynamicAnimationState that is evaluated when the animation starts. It "fills-in" specific keyframe rotation values depending on current information.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

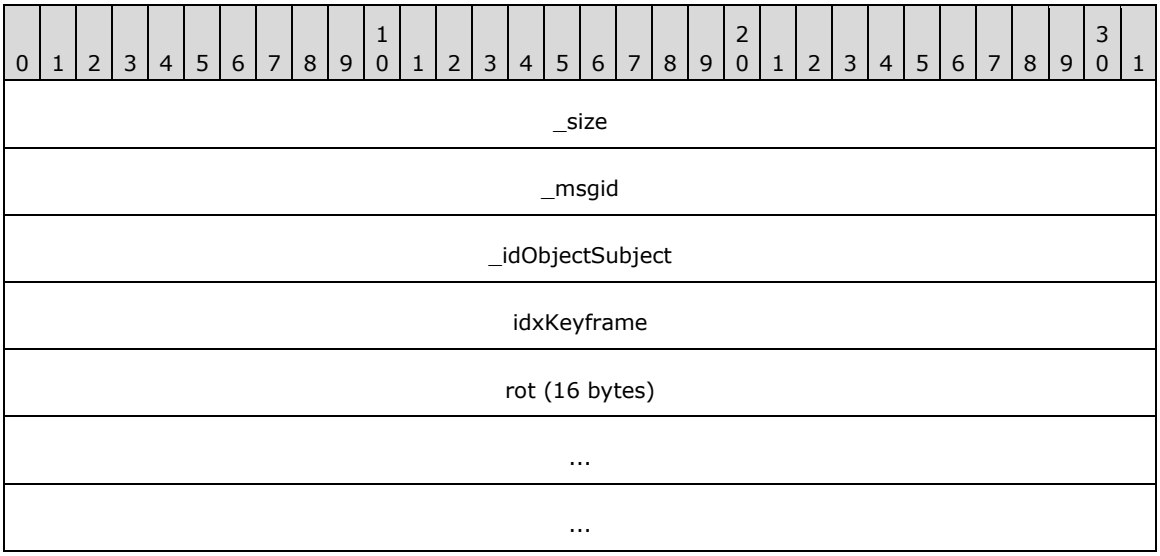
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

2.2.4.17.12 Animation_SetRotation

The Animation_SetRotation message sets the sequence components of an animation to correspond to the given rotation component values.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

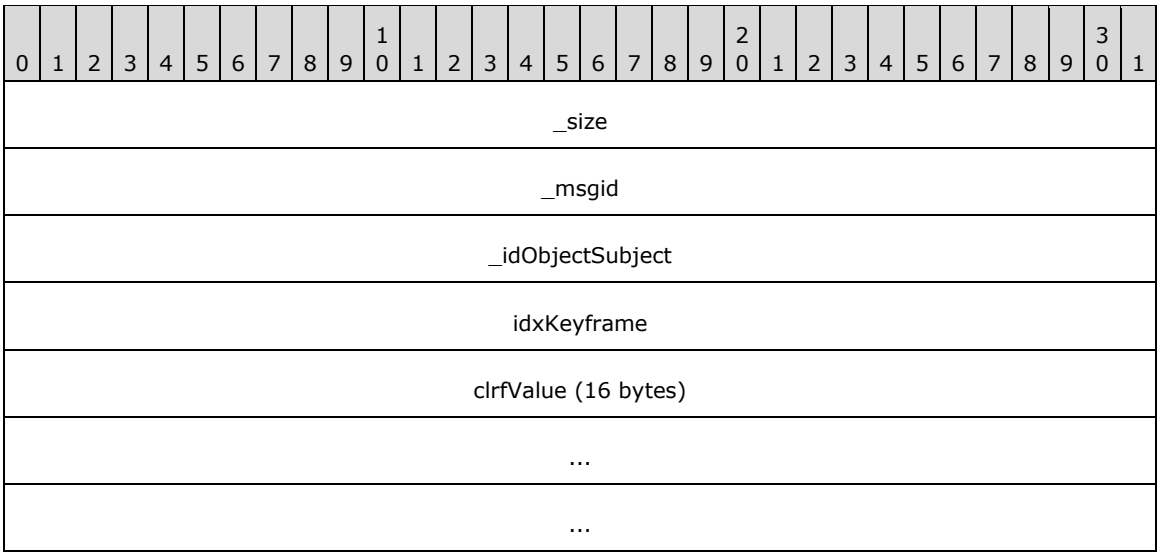
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

rot (16 bytes): A Rotation (section 2.2.6.2). The rotation to apply at the keyframe.

2.2.4.17.13 Animation_SetColorF

The Animation_SetColorF message sets the color to apply to the keyframe.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

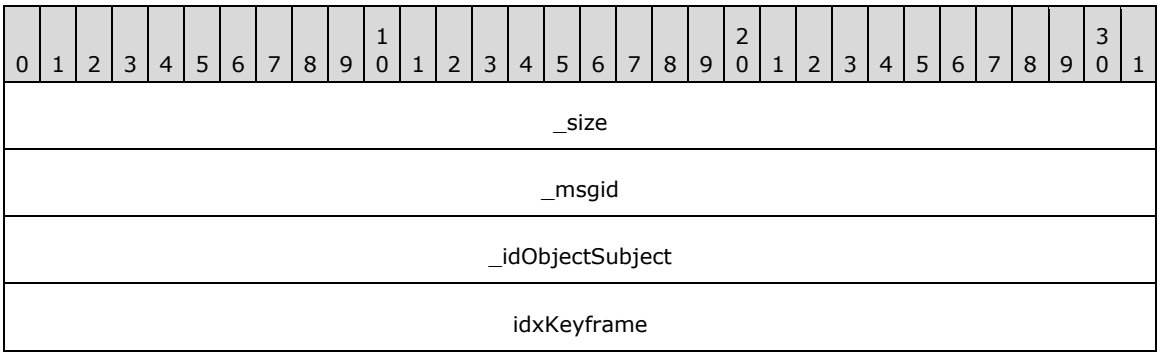
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

clrfValue (16 bytes): A ColorF (section 2.2.6.10). The color to apply at the keyframe.

2.2.4.17.14 Animation_SetDynamicARGBColor

The Animation_SetDynamicARGBColor message creates a new DynamicAnimationState that is evaluated when the animation starts. It "fills-in" specific keyframe ARGB values depending on current information.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

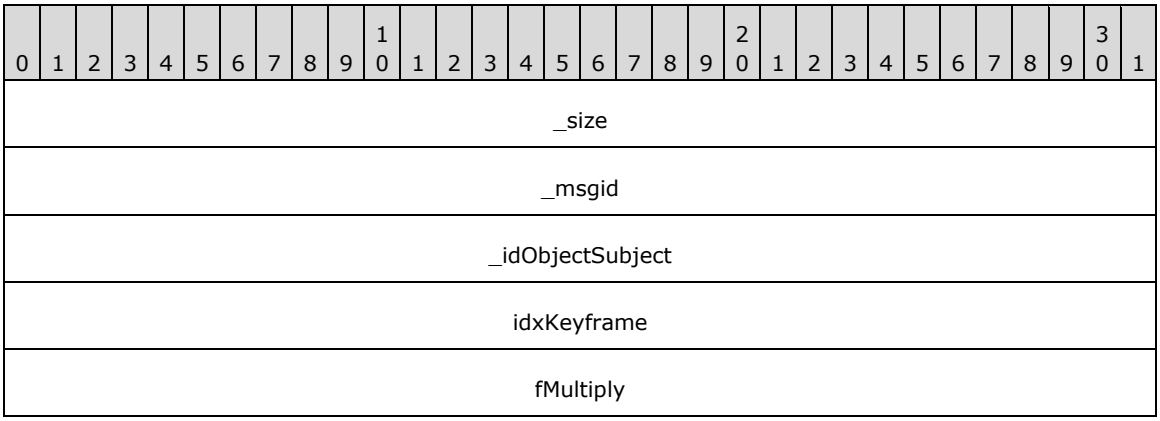
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000D for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

2.2.4.17.15 Animation_SetDynamicRGBColor

The Animation_SetDynamicRGBColor message creates a new DynamicAnimationState that is evaluated when the animation starts. It "fills-in" specific keyframe RGB values depending on current information.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000E for this message.

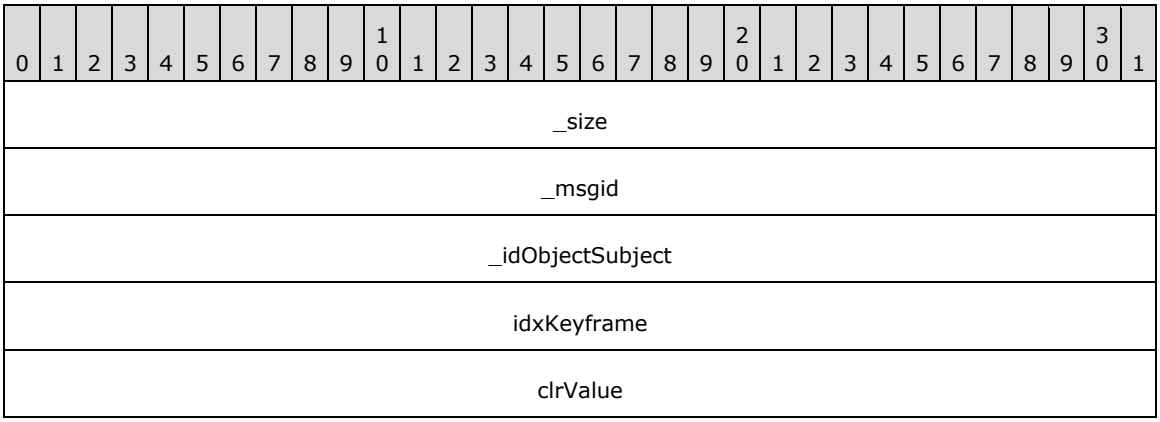
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

fMultiply (4 bytes): An unsigned 32-bit integer. Indicates whether the values can be multiplied or added.

2.2.4.17.16 Animation_SetARGBColor

The Animation_SetARGBColor message sets the ARGB color of the keyframe.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000F for this message.

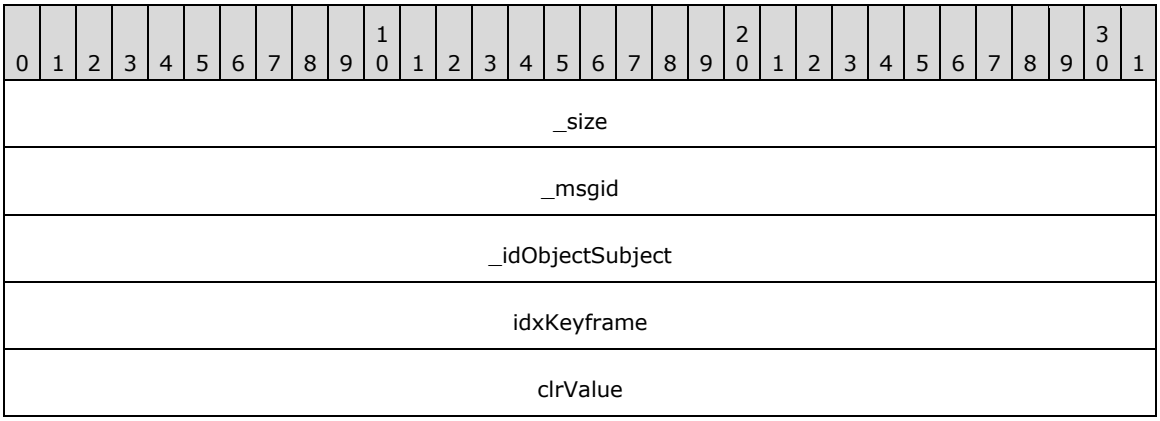
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

clrValue (4 bytes): A Color (section 2.2.6.9).The new ARBG color of the keyframe.

2.2.4.17.17 Animation_SetRGBColor

The Animation_SetRGBColor message sets the RGB color of the keyframe.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000010 for this message.

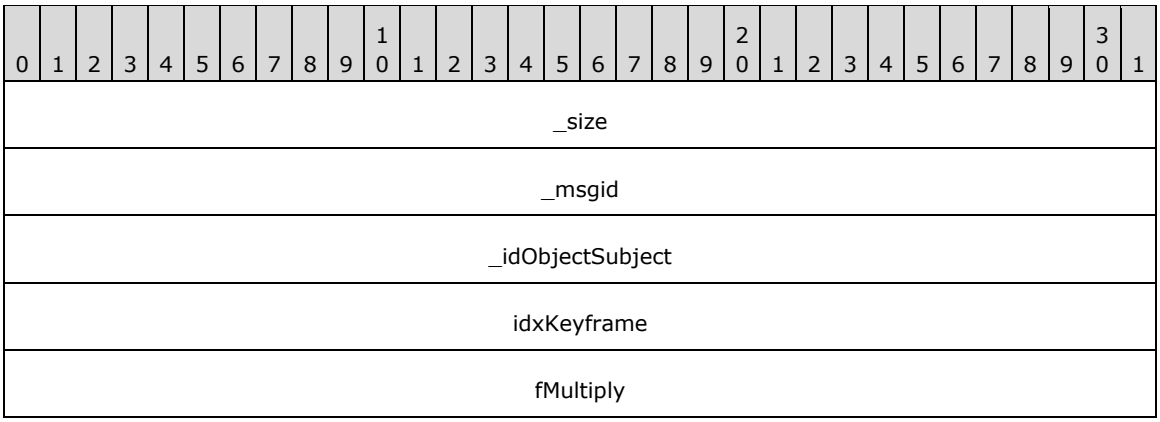
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

clrValue (4 bytes): A Color (section 2.2.6.9). The new RBG color of the keyframe.

2.2.4.17.18 Animation_SetDynamicVector3

The Animation_SetDynamicVector3 message creates a new DynamicAnimationState that is evaluated when the animation starts. It "fills-in" specific keyframe Vector3 values depending on current information.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000011 for this message.

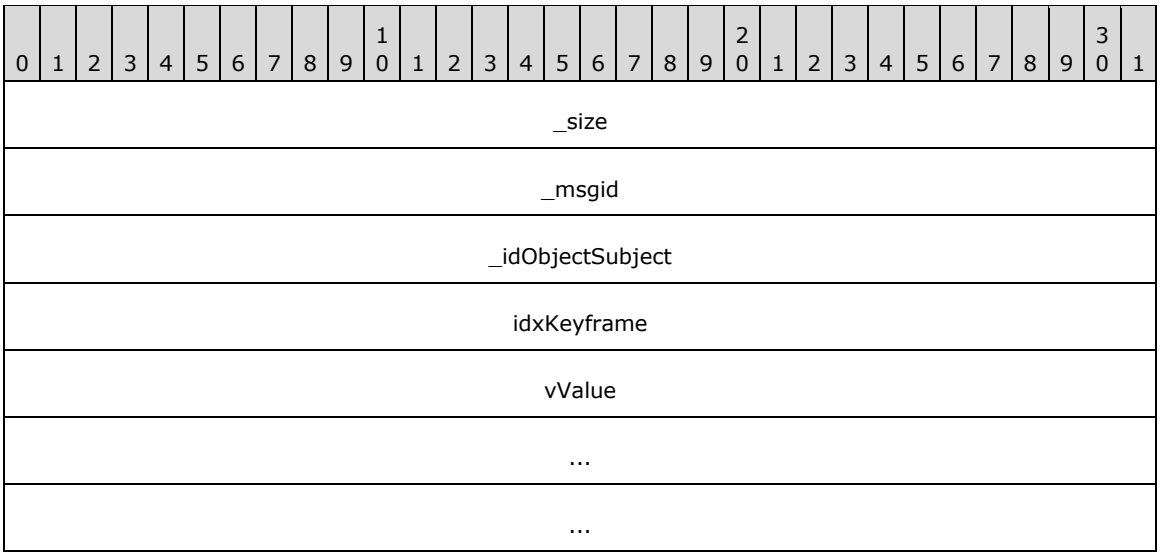
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

fMultiply (4 bytes): An unsigned 32-bit integer. Indicates whether the values can be multiplied or added.

2.2.4.17.19 Animation_SetVector3

The Animation_SetVector3 message sets the sequence components of an animation to correspond to the given vector component values.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000012 for this message.

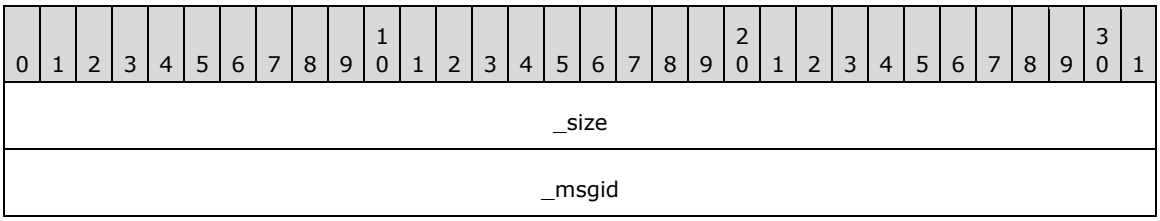
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

vValue (12 bytes): A Vector3 (section 2.2.6.3). The new Vector3 value of the keyframe.

2.2.4.17.20 Animation_SetDynamicFloat

The Animation_SetDynamicFloat message creates a new DynamicAnimationState that is evaluated when the animation starts. It "fills-in" specific keyframe float values depending on current information.



_idObjectSubject
idxKeyframe
fMultiply

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000013 for this message.

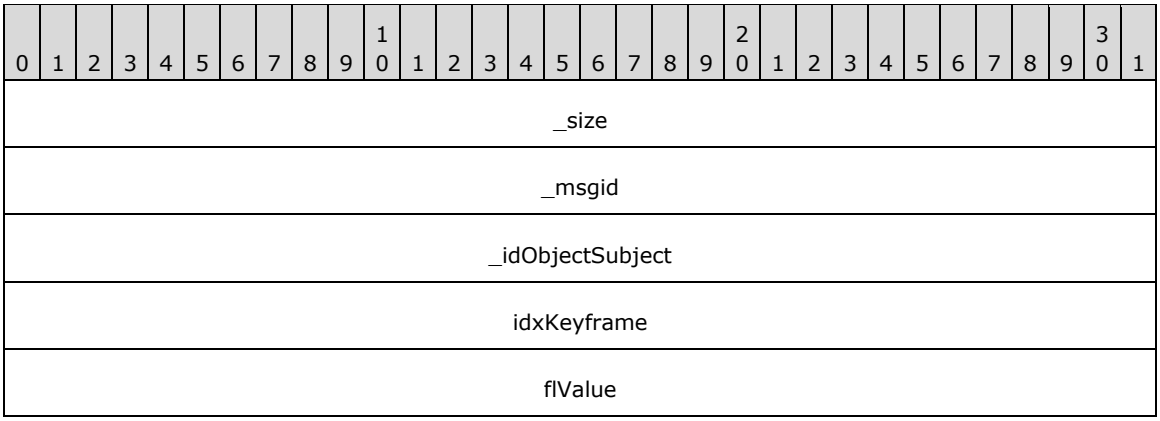
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

fMultiply (4 bytes): An unsigned 32-bit integer. Indicates whether the values can be multiplied or added.

2.2.4.17.21 Animation_SetFloat

The Animation_SetFloat message sets the sequence component of an animation to correspond to the given float value.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000014 for this message.

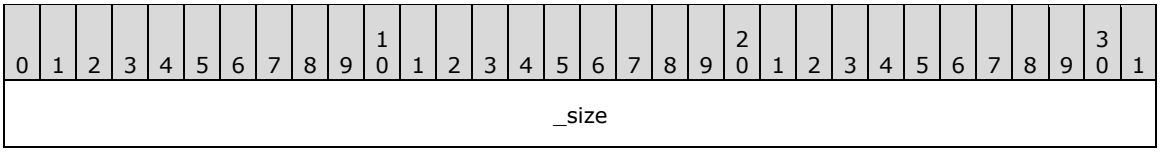
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flValue (4 bytes): A single-precision 32-bit number. The new float value of the keyframe.

2.2.4.17.22 Animation_RemoveCallback

The Animation_RemoveCallback message unregisters the specified callback.



_msgid
_idObjectSubject
_objcb
_ctxcb

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000015 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.17.23 Animation_AddCallback

The Animation_AddCallback message registers the given callback to be notified on different animation events. The callback is notified asynchronously even if it is implemented on the same thread as this animation object.

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
_size																															
_msgid																															
_idObjectSubject																															
_objcb																															
_ctxcb																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000016 for this message.

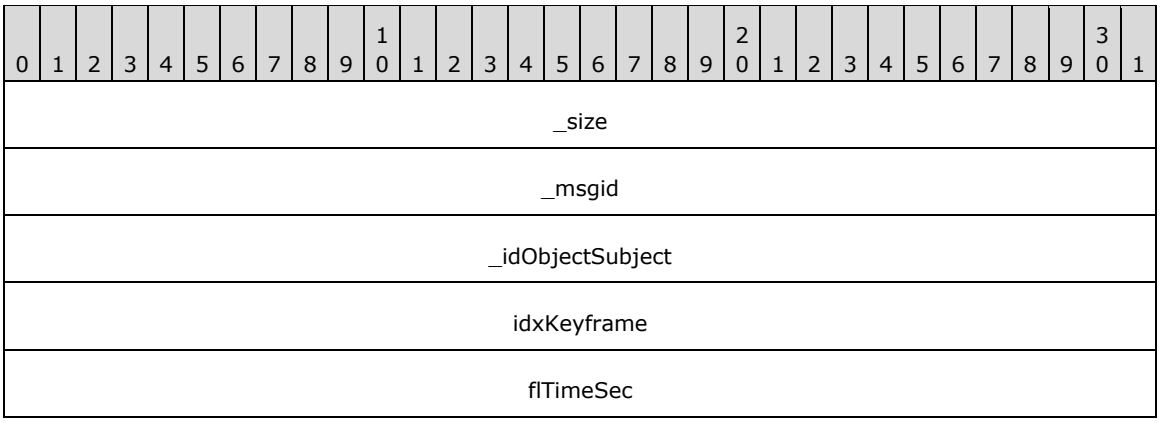
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.17.24 Animation_AddKeyframe

The Animation_AddKeyframe message adds a new keyframe at the specified index. If a keyframe already exists at the specified index, the existing keyframe is moved down.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000017 for this message.

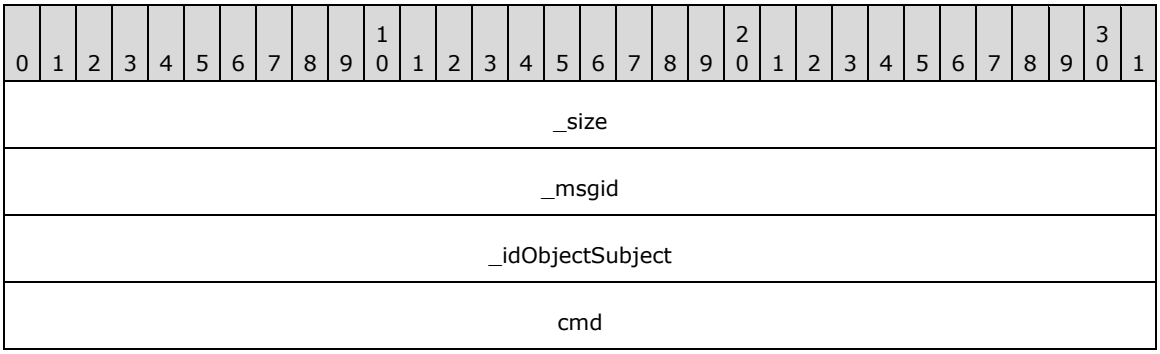
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flTimeSec (4 bytes): A single-precision 32-bit number. The keyframe time, in seconds.

2.2.4.17.25 Animation_Stop

The Animation_Stop message stops the animation that is playing. When the animation is not playing, time is not passed to the individual sequences and therefore their progress does not change. The sequences can be safely modified during this time.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000018 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

cmd (4 bytes): A signed 32-bit integer. A post-stop processing command.

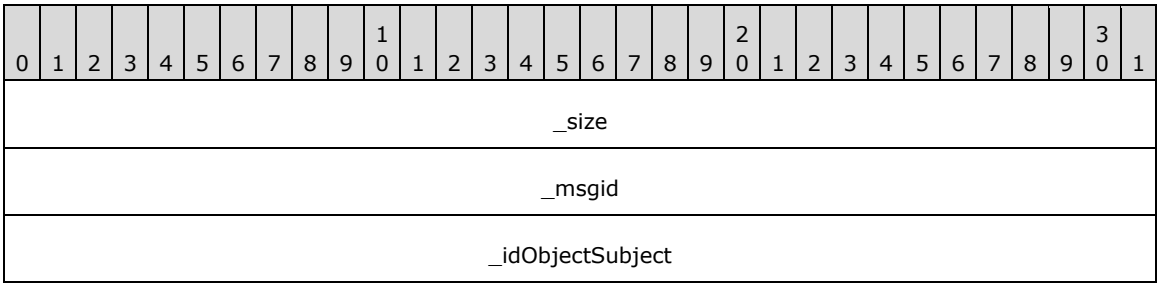
Possible values are described in the following table.

Value	Description
0x00000000	Do not move the position.

Value	Description
0x00000001	Reset the position to the beginning.
0x00000002	Advance the position to the end.

2.2.4.17.26 Animation_Play

The Animation_Play message starts the animation that is playing. While the animation is playing, time is passed to the individual sequences by advancing their timers and changing progress. The sequences cannot be modified while playing.



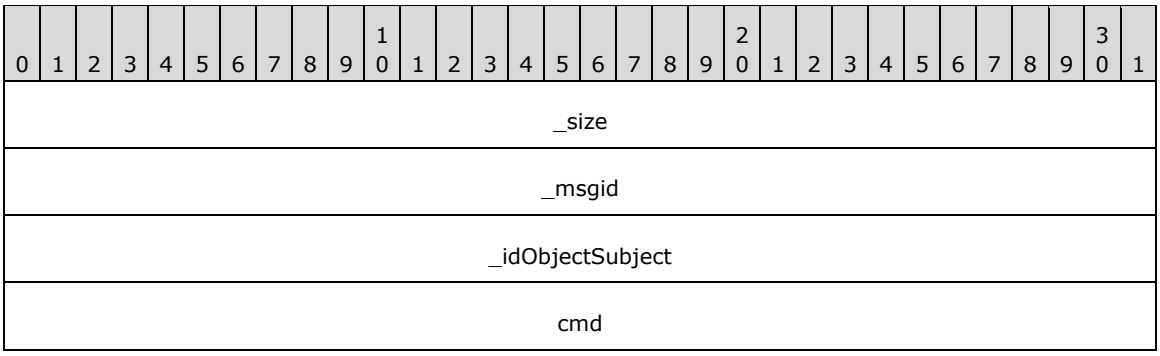
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000001A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.17.27 Animation_SetStopCommand

The Animation_SetStopCommand message changes the action to take when the animation is stopped. This message allows the subject being animated to be left in a determined state.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000001B for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

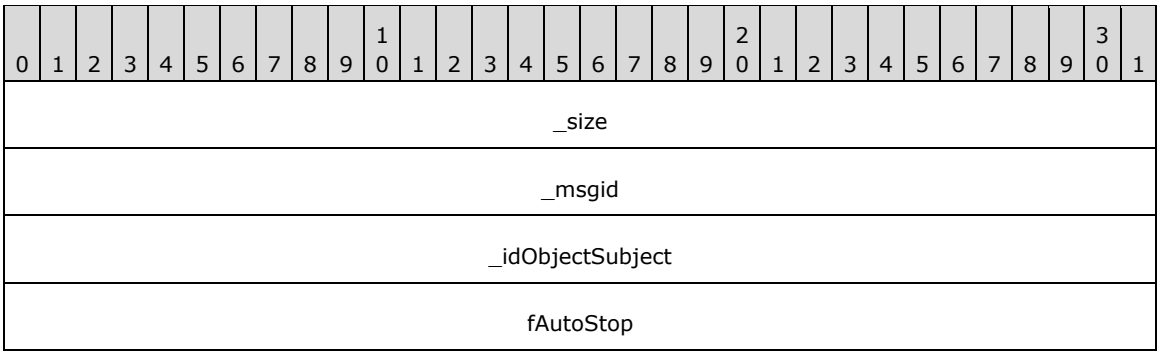
cmd (4 bytes): A signed 32-bit integer. A post-stop processing command.

Possible values are described in the following table.

Value	Description
0x00000000	Do not move the position.
0x00000001	Reset the position to the beginning.
0x00000002	Advance the position to the end.

2.2.4.17.28 Animation_SetAutoStop

The Animation_SetAutoStop message changes whether the animation automatically stops playback when each of the sequences is complete.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

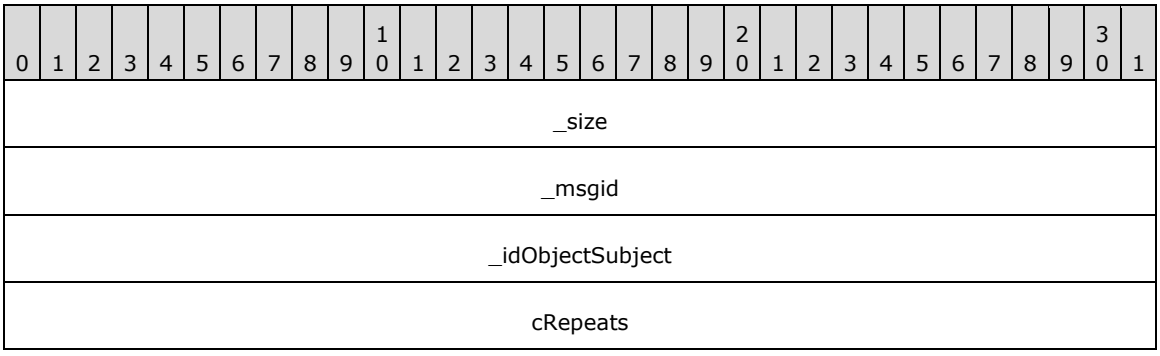
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000001D for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

fAutoStop (4 bytes): An unsigned 32-bit integer. The auto-stop value.

2.2.4.17.29 Animation_SetRepeatCount

The Animation_SetRepeatCount message changes the number of times the given animation repeats before completing.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

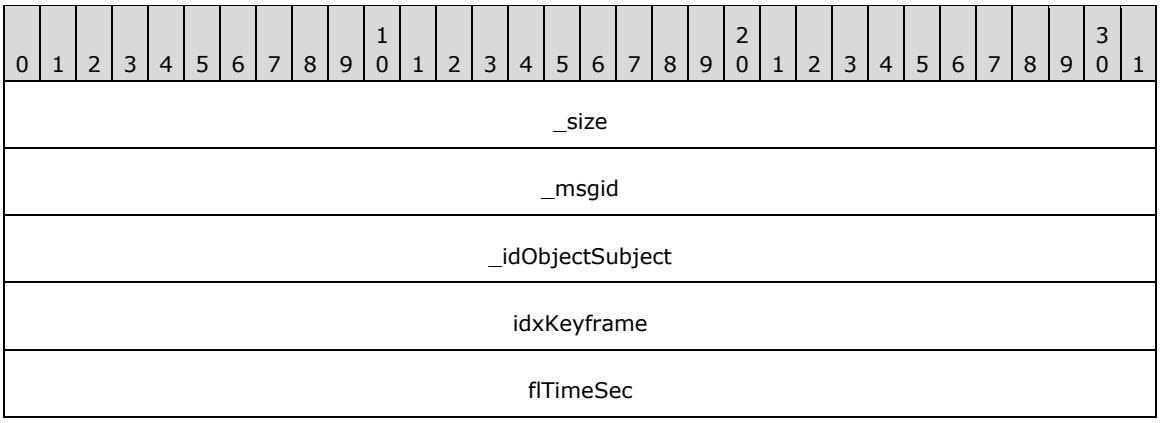
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000001E for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

cRepeats (4 bytes): A signed 32-bit integer. The number of times to repeat the animation.

2.2.4.17.30 Animation_SetKeyframeTime

The Animation_SetKeyframeTime message changes the given keyframe's time.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000021 for this message.

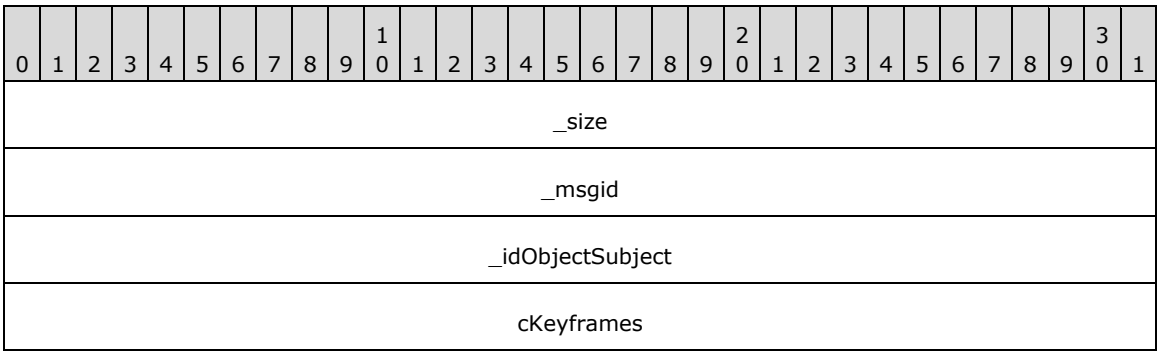
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idxKeyframe (4 bytes): A signed 32-bit integer. The index of the keyframe to modify.

flTimeSec (4 bytes): A single-precision 32-bit number. The new time of the keyframe, in seconds.

2.2.4.17.31 Animation_SetKeyframeCount

The Animation_SetKeyframeCount message changes the number of common keyframes in the animation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000023 for this message.

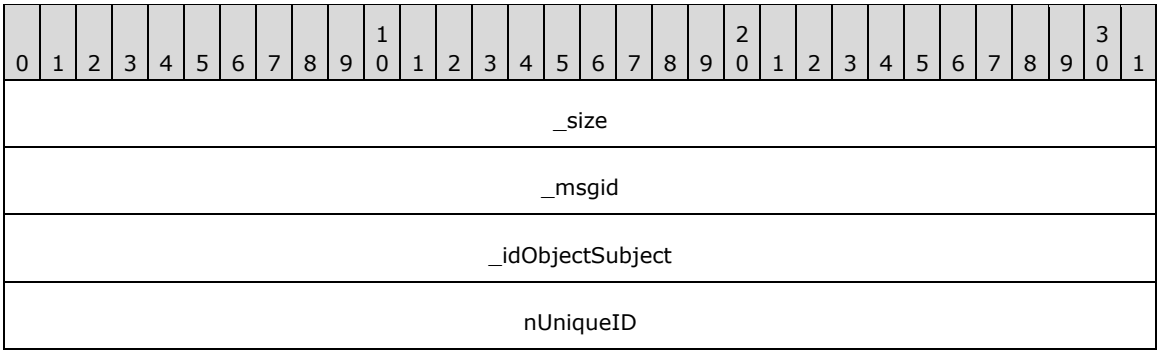
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

cKeyframes (4 bytes): A signed 32-bit integer. The new number of keyframes.

2.2.4.18 DynamicSurfaceFactory

2.2.4.18.1 DynamicSurfaceFactory_CloseInstance

The DynamicSurfaceFactory_CloseInstance message closes a DynamicSurface instance.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

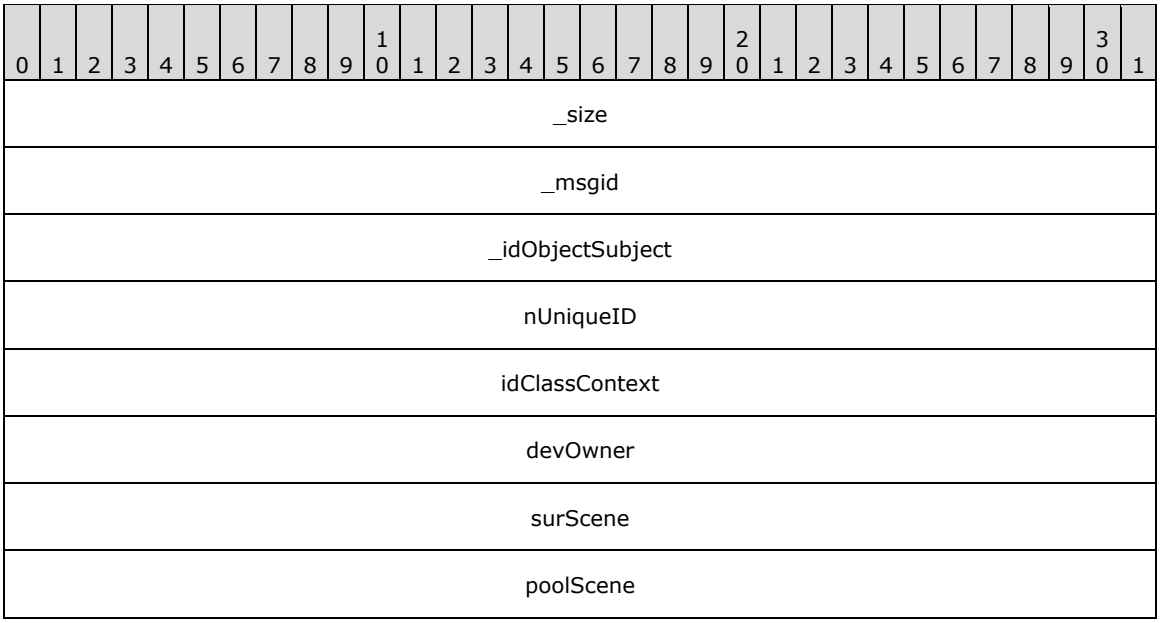
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nUniqueID (4 bytes): A signed 32-bit integer. The ID of the DynamicSurface instance.

2.2.4.18.2 DynamicSurfaceFactory_CreateVideoInstance

The DynamicSurfaceFactory_CreateVideoInstance message constructs a new pull-style DynamicSurface instance.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nUniqueID (4 bytes): A signed 32-bit integer. The ID of the DynamicSurface instance.

idClassContext (4 bytes): An unsigned 32-bit integer. ID of ClassObject for context.

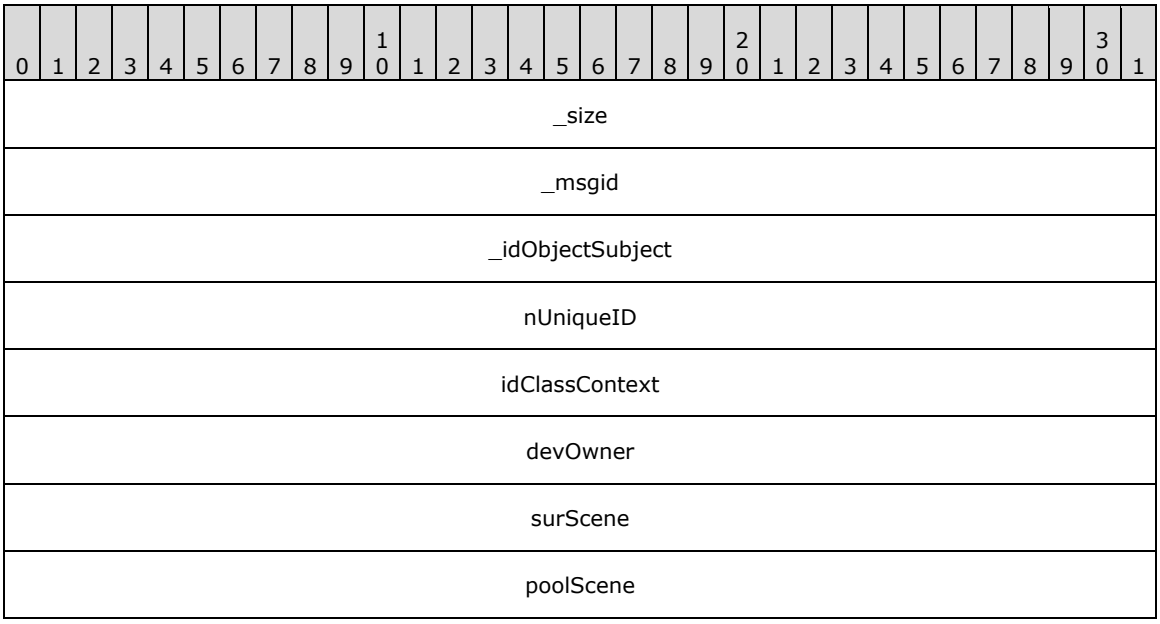
devOwner (4 bytes): An unsigned 32-bit integer. The ID of the device to use.

surScene (4 bytes): An unsigned 32-bit integer. The ID of the surface to display.

poolScene (4 bytes): An unsigned 32-bit integer. The ID of the VideoPool that contains content.

2.2.4.18.3 DynamicSurfaceFactory_CreateSurfaceInstance

The DynamicSurfaceFactory_CreateSurfaceInstance message creates a new surface instance.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nUniqueID (4 bytes): A signed 32-bit integer. The ID of the DynamicSurface instance.

idClassContext (4 bytes): An unsigned 32-bit integer. ID of ClassObject for context.

devOwner (4 bytes): An unsigned 32-bit integer. The ID of the device to use.

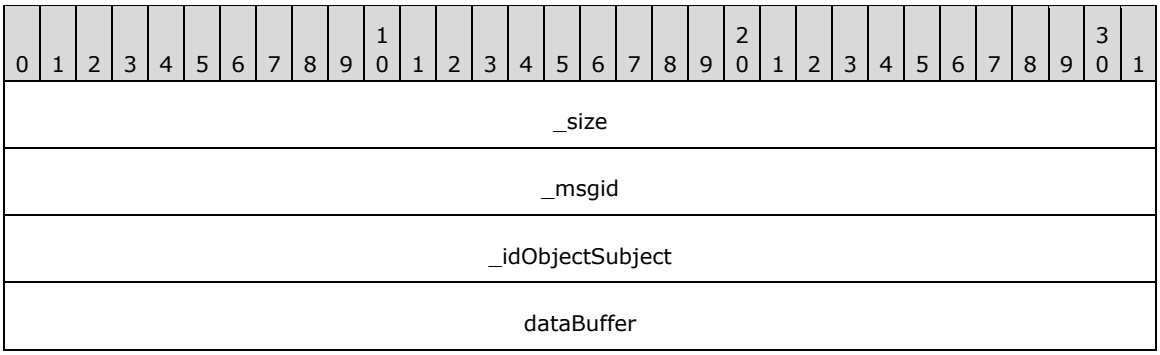
surScene (4 bytes): An unsigned 32-bit integer. The ID of the surface to display.

poolScene (4 bytes): An unsigned 32-bit integer. The ID of the VideoPool that contains content.

2.2.4.19 SoundBuffer

2.2.4.19.1 SoundBuffer_LoadSoundData

The SoundBuffer_LoadSoundData message loads the specified sound data into a sound buffer.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

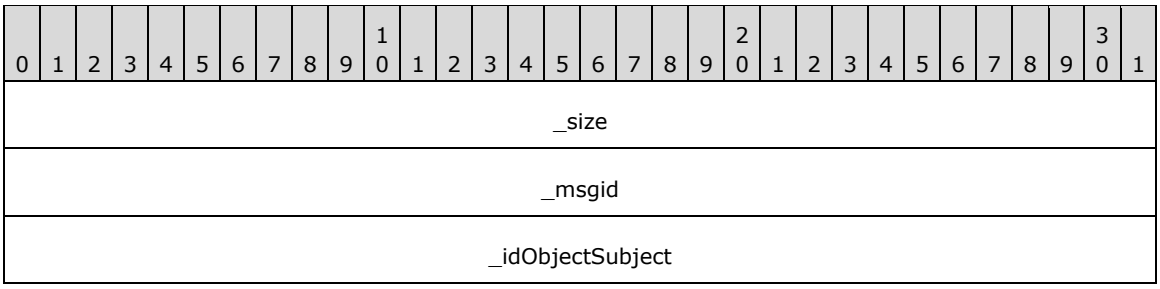
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

dataBuffer (4 bytes): An unsigned 32-bit integer. The ID of the DataBuffer that contains the sound data.

2.2.4.20 Sound

2.2.4.20.1 Sound_Stop

The Sound_Stop message stops sound playback if necessary, and releases the lock previously acquired when Sound_Play was called.



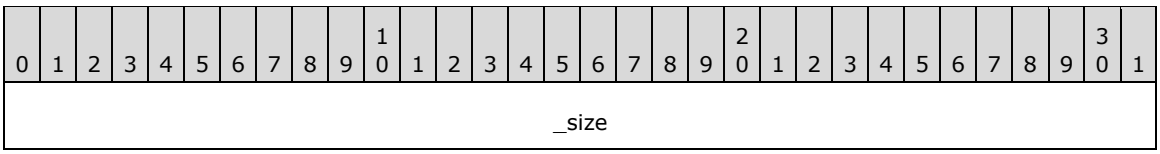
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.20.2 Sound_Play

The Sound_Play message starts sound playback. If the sound is already playing, playback is restarted. The object is locked while the sound is being played.



_msgid
_idObjectSubject

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.21 SoundDevice

2.2.4.21.1 SoundDevice_CreateSound

The SoundDevice_CreateSound message creates a sound object and associates it with the specified SoundBuffer.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewSound																															
soundBuffer																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSound (4 bytes): An unsigned 32-bit integer. The ID of the new sound object.

soundBuffer (4 bytes): An unsigned 32-bit integer. The ID of the SoundBuffer to associate with the sound.

2.2.4.21.2 SoundDevice_CreateSoundBuffer

The SoundDevice_CreateSoundBuffer message creates a SoundBuffer and associates it with the SoundDevice.

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
_size																															
_msgid																															

_idObjectSubject	
idNewBuffer	
info (22 bytes)	
...	
...	
...	objcb
...	ctxcb
...	

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewBuffer (4 bytes): A signed 32-bit integer. The ID of the new SoundBuffer object.

info (22 bytes): A SoundHeader (section 2.2.6.11). Information about the SoundBuffer to be created.

objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.21.3 SoundDevice_EvictExternalResources

The SoundDevice_EvictExternalResources message releases all driver-specific resources used by the object.

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
_size																															
_msgid																															
_idObjectSubject																															

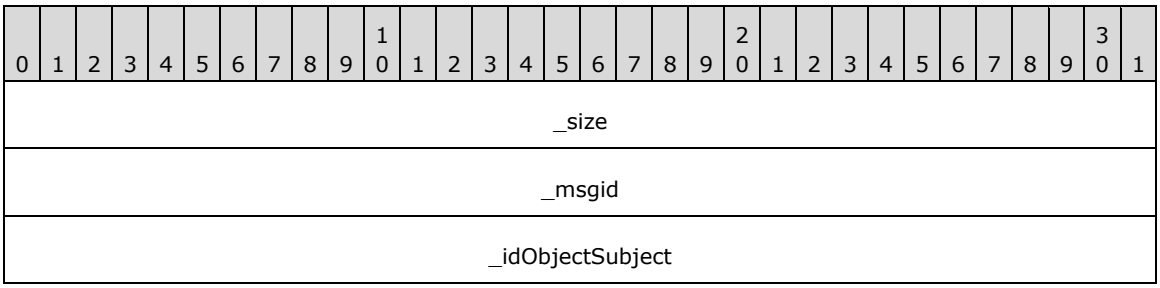
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.21.4 SoundDevice_CreateExternalResources

The SoundDevice_CreateExternalResources message creates the driver-specific resources that the object requires.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

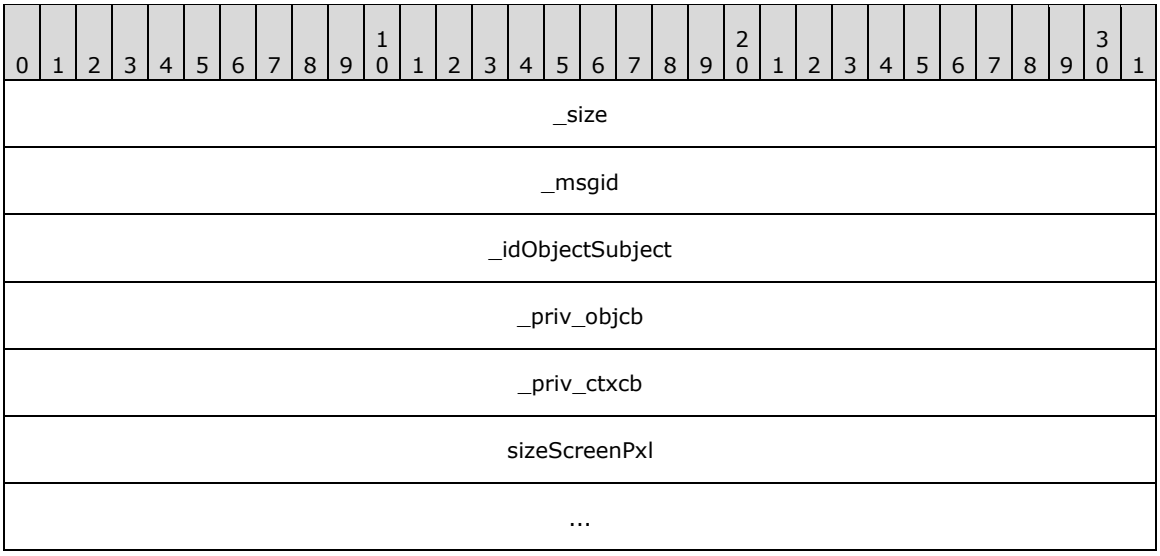
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.22 XeDevice

2.2.4.22.1 XeDevice_Create

The XeDevice_Create message completes construction of a new device. Anything that could potentially return an error is handled in this second stage.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000E for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

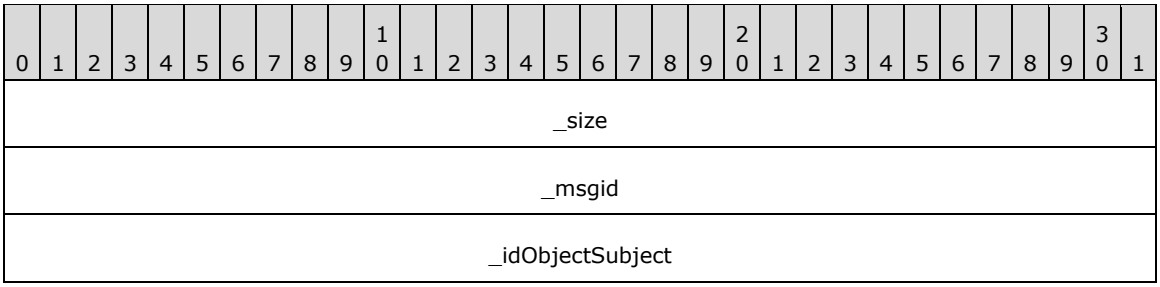
_priv_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_priv_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

sizeScreenPxl (8 bytes): A Size (section 2.2.6.6). The requested screen resolution, in pixels.

2.2.4.22.2 XeDevice_Stop

The XeDevice_Stop message stops rendering the current generation on this device. Any time rendering has to stop, this count is increased. For rendering to continue, the application MUST restart the new generation, when ready. This allows the application to setup any state before displaying to the user.



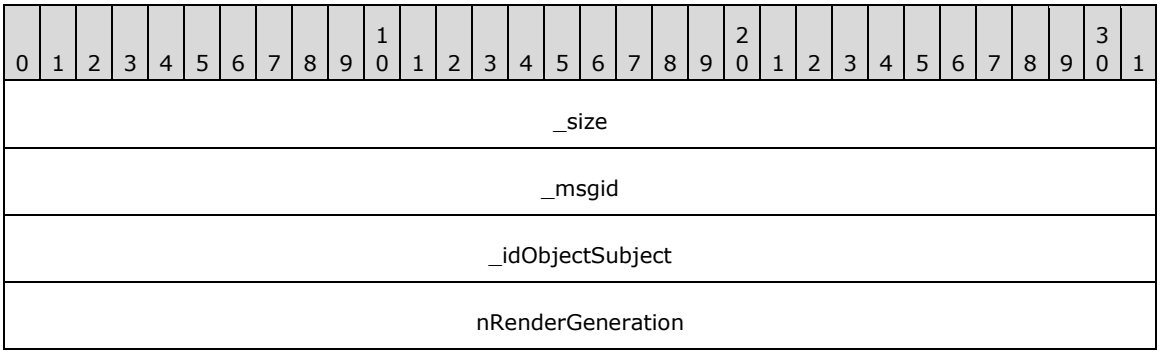
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.22.3 XeDevice_Restart

The XeDevice_Restart message restarts a previously stopped rendering generation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

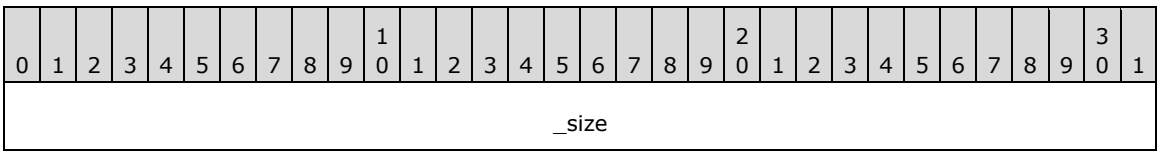
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nRenderGeneration (4 bytes): An unsigned 32-bit integer. The render generation to restart.

2.2.4.22.4 XeDevice_DrawLine

The XeDevice_DrawLine message draws a line of the given color.



_msgid
_idObjectSubject
rb
clrLine
flThickness
vStart
...
...
vEnd
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrLine (4 bytes): A Color (section 2.2.6.9). The color of the line.

flThickness (4 bytes): A single-precision 32-bit number. The line thickness.

vStart (12 bytes): A Vector3 (section 2.2.6.3) that specifies the starting position of the line.

vEnd (12 bytes): A Vector3 (section 2.2.6.3) that specifies the end position of the line.

2.2.4.22.5 XeDevice_DrawOutline

The XeDevice_DrawOutline message draws a 1-pixel outline.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															

clrOutline
flThickness
rcfOutline (16 bytes)
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrOutline (4 bytes): A Color (section 2.2.6.9). The color of the outline.

flThickness (4 bytes): A single-precision 32-bit number. The outline thickness.

rcfOutline (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area around which to draw the outline, in pixels.

2.2.4.22.6 XeDevice_DrawSolid

The XeDevice_DrawSolid message draws a solid rectangle of the given color.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewBuffer																															
rb																															
clrFill																															
rcfFill (16 bytes)																															
...																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewBuffer (4 bytes): A signed 32-bit integer. The ID of the new SoundBuffer object.

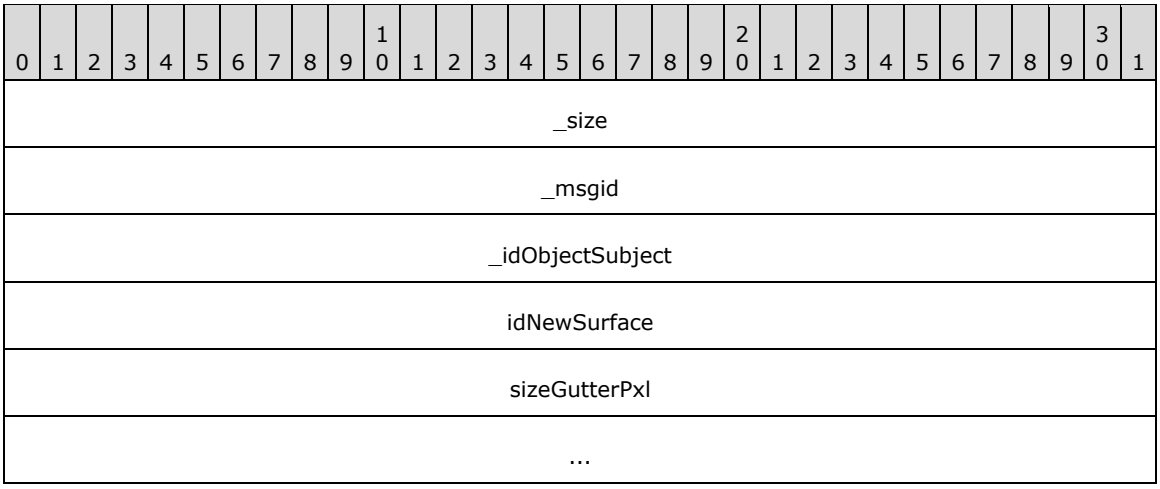
rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrFill (4 bytes): A Color (section 2.2.6.9). The color of the outline.

rcfFill (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area around which to draw the outline, in pixels.

2.2.4.22.7 XeDevice_CreateSurfacePool

The XeDevice_CreateSurfacePool message has the device create a new surface pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

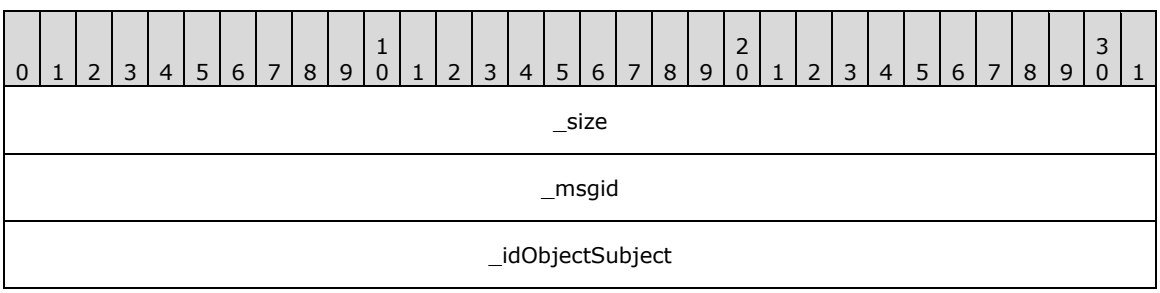
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID to assign to the new surface pool.

sizeGutterPxl (8 bytes): A Size (section 2.2.6.6). The gutter around surfaces, in pixels.

2.2.4.22.8 XeDevice_CreateVideoPool

The XeDevice_CreateVideoPool message has the device create a new video pool.



_priv_objcbOwner
_priv_ctxcbOwner
idNewSurface

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

_priv_objcbOwner (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_priv_ctxcbOwner (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID of the new surface.

2.2.4.22.9 XeDevice_CreateLine

The XeDevice_CreateLine message has the device create a new line.

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
_size																															
_msgid																															
_idObjectSubject																															
idLine																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idLine (4 bytes): An unsigned 32-bit integer. The ID of the new line.

2.2.4.22.10 XeDevice_CreateGradient

The XeDevice_CreateGradient message has the device create a new gradient.

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
_size																															
_msgid																															

_idObjectSubject
idNewGradient

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewGradient (4 bytes): An unsigned 32-bit integer. The ID of the new gradient.

2.2.4.22.11 XeDevice_DrawNotify

The XeDevice_DrawNotify message sets up so the profiler is notified of when the content in this render builder reaches the screen.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															
uId																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

uId (4 bytes): An unsigned 32-bit integer. An ID to use for the notification.

2.2.4.22.12 XeDevice_EndVideoSurfaceAllocation

The XeDevice_EndVideoSurfaceAllocation message closes a session that is previously started by an XeDevice_BeginVideoSurfaceAllocation message whereby an external component has to allocate video memory. When the session is closed, all surfaces are restored and the device becomes available for rendering.

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
_size																															
_msgid																															

_idObjectSubject

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.22.13 XeDevice_BeginVideoSurfaceAllocation

The XeDevice_BeginVideoSurfaceAllocation message frees video memory for an external component to allocate local video memory. The caller is responsible for sending an XeDevice_EndVideoSurfaceAllocation message when finished. During this time, the device becomes unavailable for rendering.

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
_size																															
_msgid																															
_idObjectSubject																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000C for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.22.14 XeDevice_Enter3DMode

The XeDevice_Enter3DMode message creates a RenderOperation to draw the main 3d scene. This message allows the application to control what operations are executed before and after the main scene starts to render.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000D for this message.

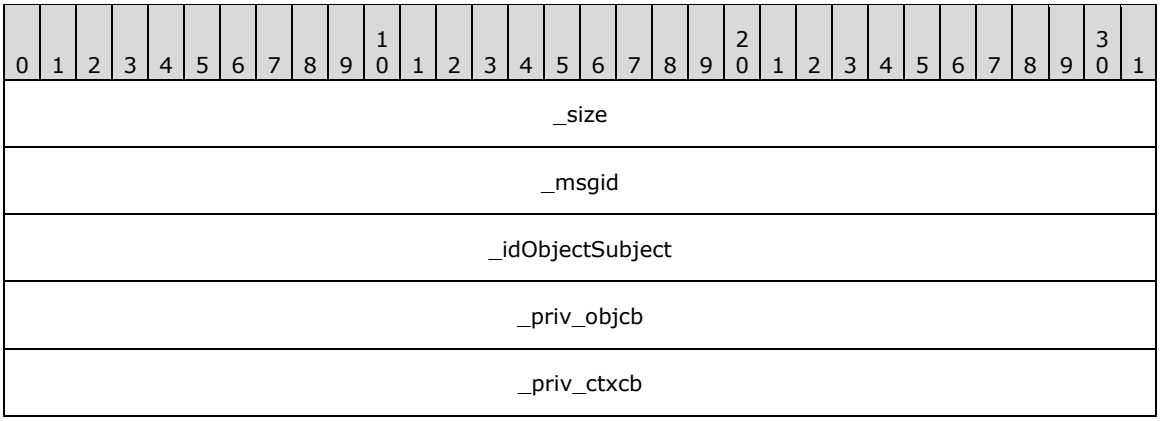
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

2.2.4.23 HostWindow

2.2.4.23.1 HostWindow_Create

The HostWindow_Create message completes construction of a new HostWindow. Anything that could potentially return an error is handled in this second stage.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

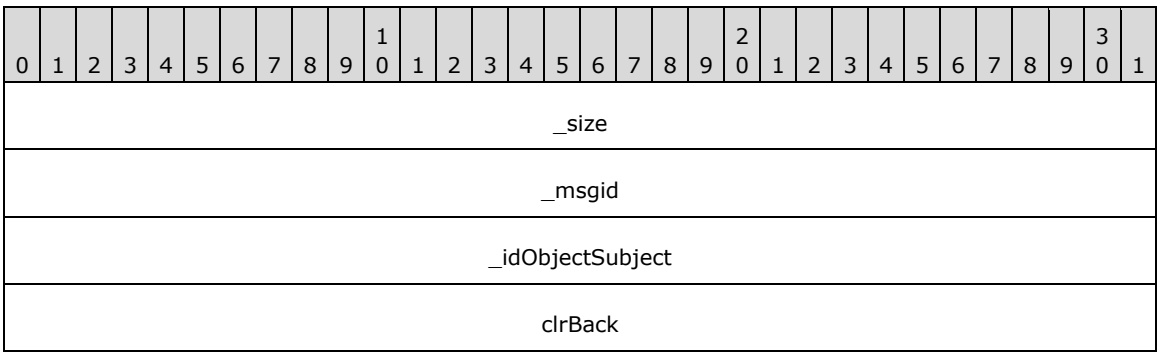
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

_priv_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_priv_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.23.2 HostWindow_SetBackgroundColor

The HostWindow_SetBackgroundColor message changes the default background color for the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

clrBack (4 bytes): A Color (section 2.2.6.9). The color of the window's background.

2.2.4.23.3 HostWindow_SetPerspectiveSettings

The HostWindow_SetPerspectiveSettings message sets the viewing perspective of the window.

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
_size																															
_msgid																															
_idObjectSubject																															
flZn																															
flEye																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flZn (4 bytes): A single-precision 32-bit number. The distance of "eye" from "at" to the nearest plane.

flEye (4 bytes): A single-precision 32-bit number. The distance of "eye" from "at" to the furthest plane.

At: The center of the object you want to look "at".

Eye: The location of the eye (camera).

2.2.4.23.4 HostWindow_ChangeDataBits

The HostWindow_ChangeDataBits message changes the user-defined bits set on the window.

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
_size																															
_msgid																															
_idObjectSubject																															
nValue																															
nMask																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

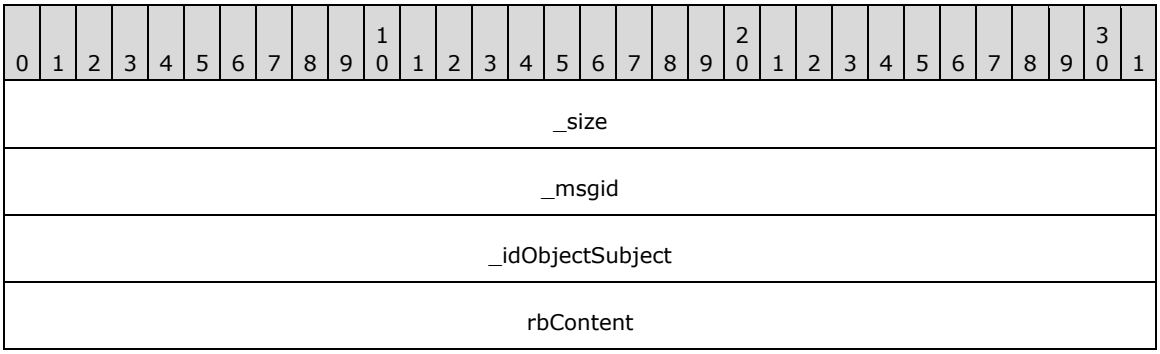
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nValue (4 bytes): An unsigned 32-bit integer. The new value.

nMask (4 bytes): An unsigned 32-bit integer. A mask to use when changing the bits.

2.2.4.23.5 HostWindow_SetContent

The HostWindow_SetContent message copies the RenderOperations from the given RenderBuilder into the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

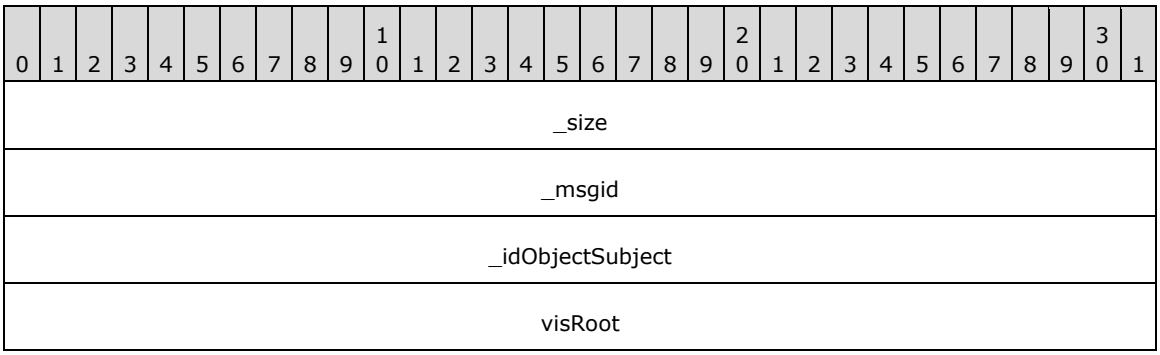
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rbContent (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder.

2.2.4.23.6 HostWindow_SetRoot

The HostWindow_SetRoot message changes the root visual associated with the window.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

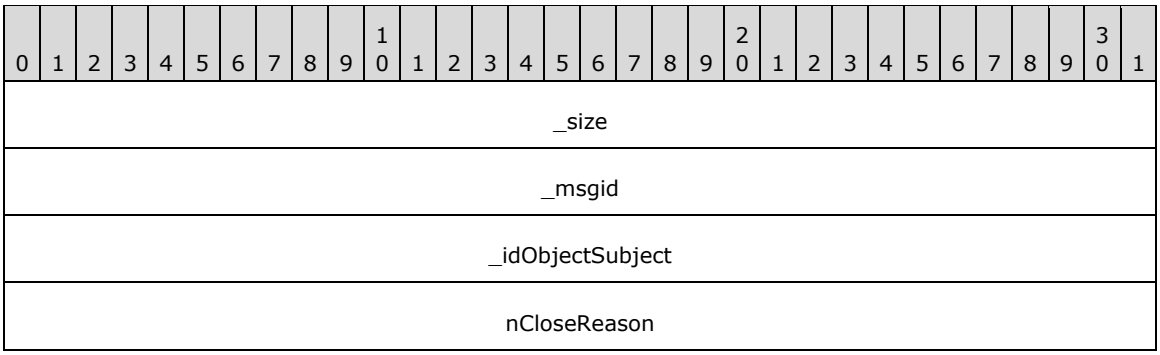
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000008 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

visRoot (4 bytes): An unsigned 32-bit integer. The ID of the new root visual.

2.2.4.23.7 HostWindow_SetCloseReason

The HostWindow_SetCloseReason message sets the reason the window is being closed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

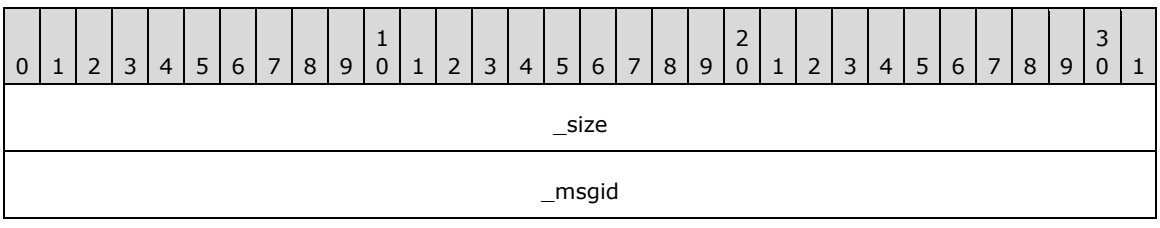
nCloseReason (4 bytes): A signed 32-bit integer. The ID of close reason. Possible values are described in the following table.

Value	Description
0xFFFFFFFF	Unknown Reason.
0x00000000	Externally Forced.
0x00000001	User Requested.
0x00000002	Auto Restart.
0x00000003	Renderer Requested.
0x00000004	Generic Error.

2.2.4.24 XAudSoundDevice

2.2.4.24.1 XAudSoundDevice_Create

The XAudSoundDevice_Create message completes construction of a new SoundDevice. Anything that could potentially return an error is handled in this second stage.



_idObjectSubject

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000006 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.24.2 XAudSoundDevice_CreateSound

The XAudSoundDevice_CreateSound message creates a sound object and associates it with the specified SoundBuffer.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewSound																															
soundBuffer																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSound (4 bytes): An unsigned 32-bit integer. The ID of the new sound object.

soundBuffer (4 bytes): An unsigned 32-bit integer. The ID of the SoundBuffer to associate with the sound.

2.2.4.24.3 XAudSoundDevice_CreateSoundBuffer

The XAudSoundDevice_CreateSoundBuffer message creates a SoundBuffer and associates it with the SoundDevice.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewBuffer																															

info (22 bytes)	
...	
...	
...	_priv_objcb
...	_priv_ctxcb
...	

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewBuffer (4 bytes): A signed 32-bit integer. The ID of the new SoundBuffer object.

info (22 bytes): A SoundHeader (section 2.2.6.11). Information about the SoundBuffer to be created.

_priv_objcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_priv_ctxcb (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

2.2.4.24.4 XAudSoundDevice_EvictExternalResources

The XAudSoundDevice_EvictExternalResources message releases all driver-specific resources used by the object.

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
_size																															
_msgid																															
_idObjectSubject																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.24.5 XAudSoundDevice_CreateExternalResources

The XAudSoundDevice_CreateExternalResources message creates the driver-specific resources that the object requires.

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
_size																															
_msgid																															
_idObjectSubject																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.24.6 XAudSoundDevice_SetMute

The XAudSoundDevice_SetMute message mutes or unmutes the sound device.

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
_size																															
_msgid																															
_idObjectSubject																															
fMuted																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

fMuted (4 bytes): An unsigned 32-bit integer. Indicates whether the sound device can be muted.

2.2.4.24.7 XAudSoundDevice_SetVolume

The XAudSoundDevice_SetVolume message sets the master volume level for all sounds played with the sound device.

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
_size																															
_msgid																															
_idObjectSubject																															

flVolume

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The nMsg value is 0x00000036 for this message.

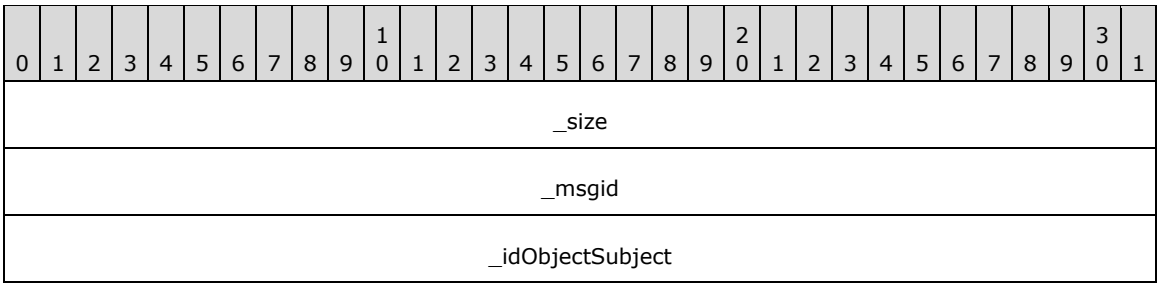
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

flVolume (4 bytes): A single-precision 32-bit number. The volume level. The value MUST be within the range of 0.0 and 1.0.

2.2.4.25 Dx9Device

2.2.4.25.1 Dx9Device_Stop

The Dx9Device_Stop message stops rendering the current generation on this device. Any time rendering has to stop, this count is increased. For rendering to continue, the application MUST restart the new generation, when ready. This message allows the application to set up any state before displaying it to the user.



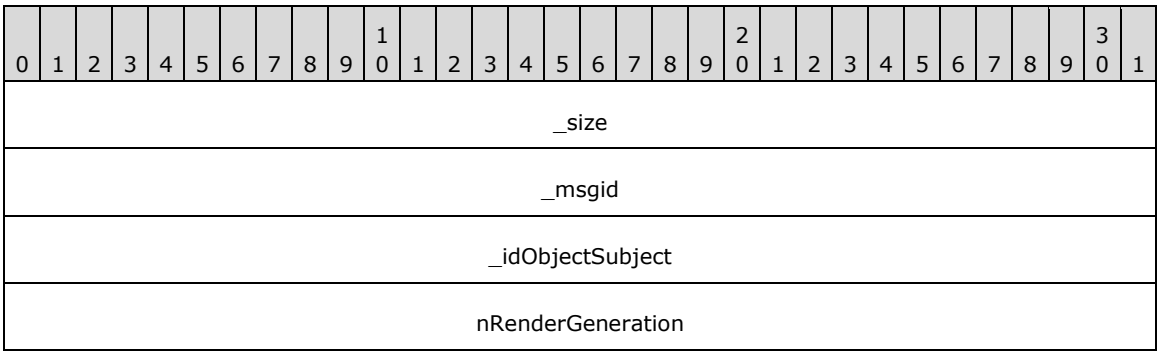
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.25.2 Dx9Device_Restart

The Dx9Device_Restart message restarts a previously stopped rendering generation.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

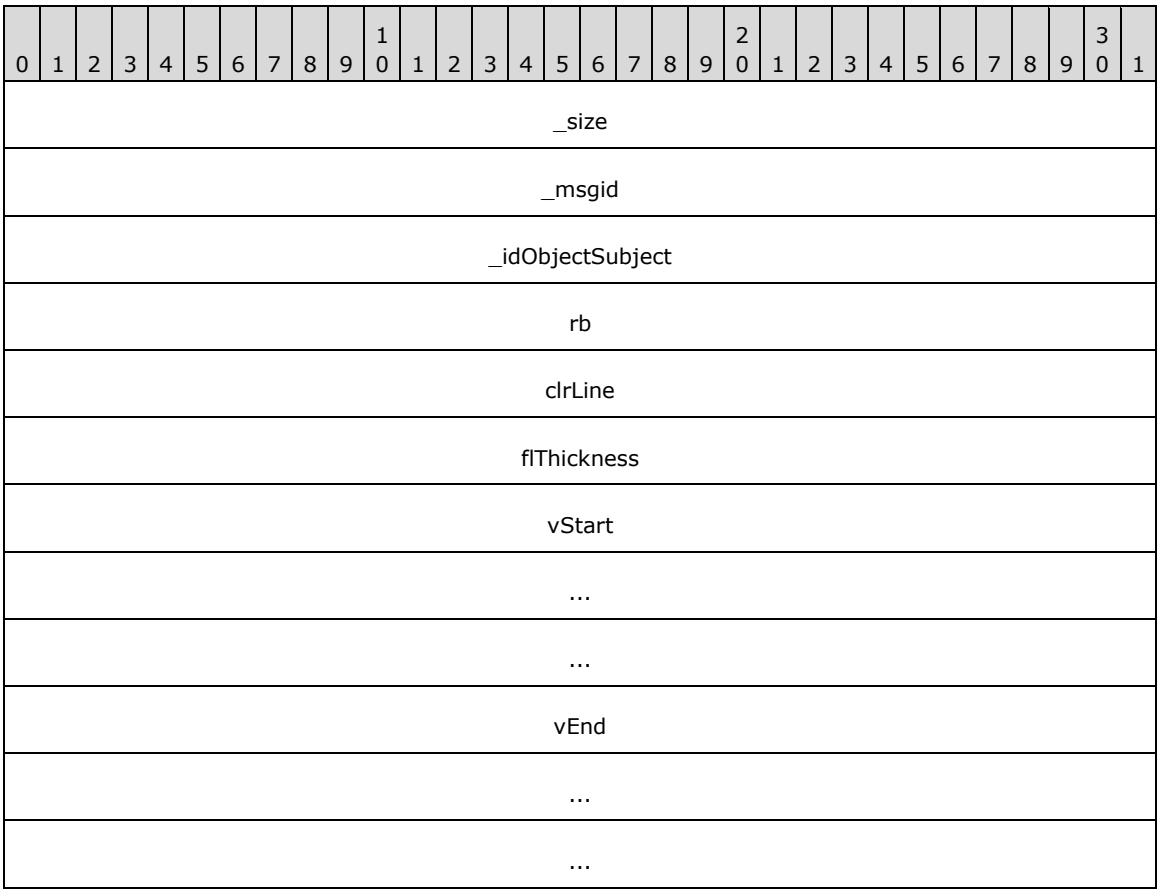
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

nRenderGeneration (4 bytes): An unsigned 32-bit integer. The render generation to restart.

2.2.4.25.3 Dx9Device_DrawLine

The Dx9Device_DrawLine message draws a line of the given color.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrLine (4 bytes): A Color (section 2.2.6.9). The color of the line.

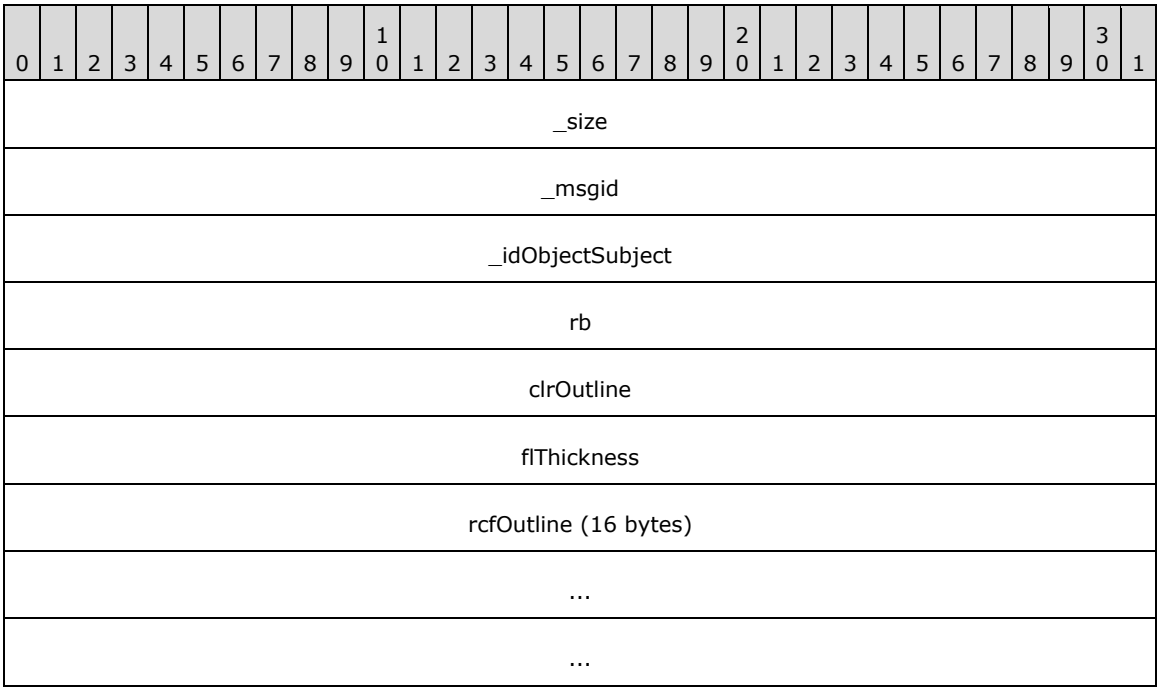
flThickness (4 bytes): A single-precision 32-bit number. The line thickness.

vStart (12 bytes): A Vector3 (section 2.2.6.3) that specifies the starting position of the line.

vEnd (12 bytes): A Vector3 (section 2.2.6.3) that specifies the end position of the line.

2.2.4.25.4 Dx9Device_DrawOutline

The Dx9Device_DrawOutline message draws a 1-pixel outline.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

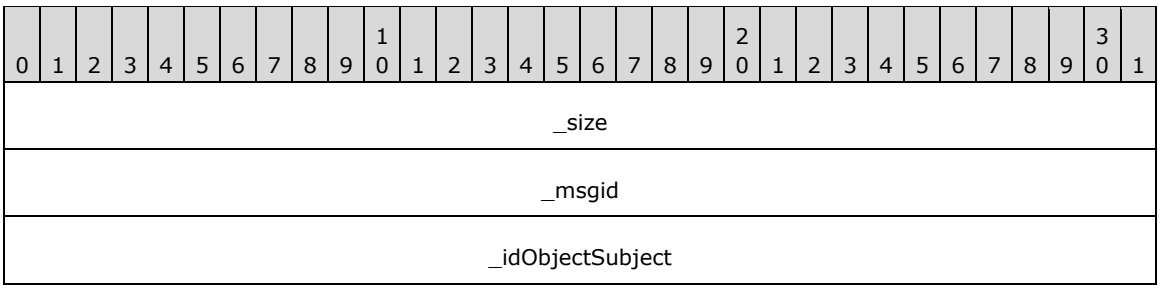
clrOutline (4 bytes): A Color (section 2.2.6.9). The color of the outline.

flThickness (4 bytes): A single-precision 32-bit number. The outline thickness.

rcfOutline (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area around which to draw the outline, in pixels.

2.2.4.25.5 Dx9Device_DrawSolid

The Dx9Device_DrawSolid message draws a solid rectangle of the given color.



idNewBuffer
rb
clrFill
rcfFill (16 bytes)
...
...

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000004 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewBuffer (4 bytes): A signed 32-bit integer. The ID of the new SoundBuffer object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the render builder to use.

clrFill (4 bytes): A Color (section 2.2.6.9). The color of the outline.

rcfFill (16 bytes): A RectangleF (section 2.2.6.5) that specifies the area around which to draw the outline, in pixels.

2.2.4.25.6 Dx9Device_CreateSurfacePool

The Dx9Device_CreateSurfacePool message has the device create a new surface pool.

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
_size																															
_msgid																															
_idObjectSubject																															
idNewSurface																															
sizeGutterPxl																															
...																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000005 for this message.

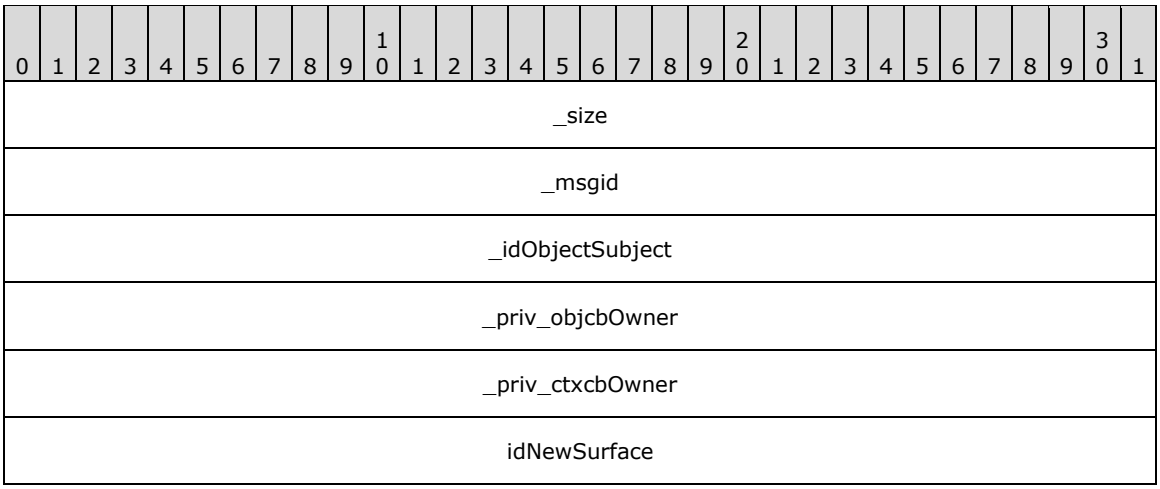
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID to assign to the new surface pool.

sizeGutterPxl (8 bytes): A Size (section 2.2.6.6). The gutter around surfaces, in pixels.

2.2.4.25.7 Dx9Device_CreateVideoPool

The Dx9Device_CreateVideoPool message has the device create a new video pool.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000007 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

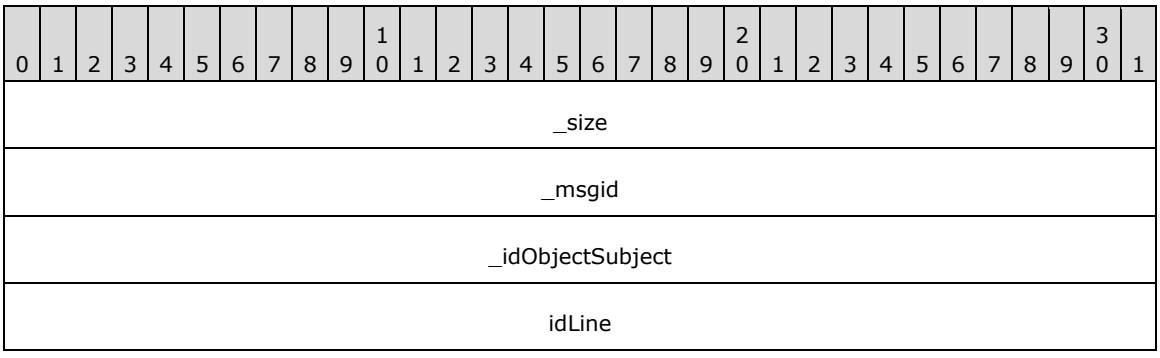
_priv_objcbOwner (4 bytes): An unsigned 32-bit integer. The ID of the owner's callback.

_priv_ctxcbOwner (4 bytes): An unsigned 32-bit integer. The ID of the owner's context.

idNewSurface (4 bytes): An unsigned 32-bit integer. The ID of the new surface.

2.2.4.25.8 Dx9Device_CreateLine

The Dx9Device_CreateLine message has the device create a new line.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

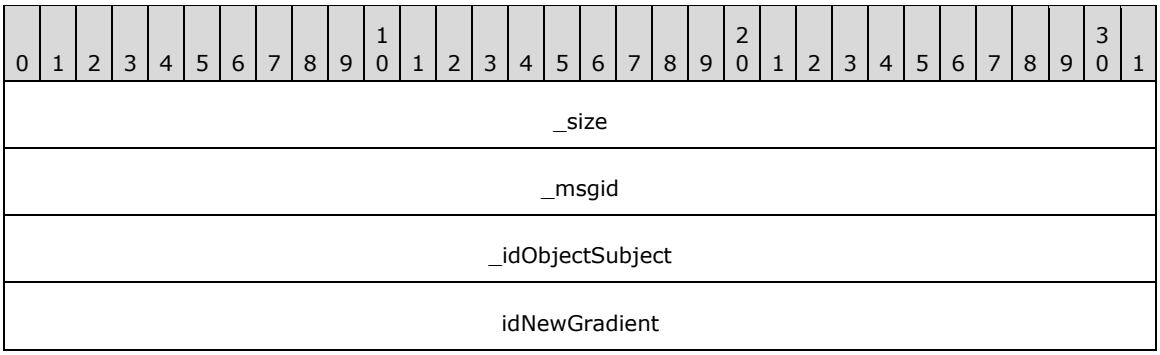
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idLine (4 bytes): An unsigned 32-bit integer. The ID of the new line.

2.2.4.25.9 Dx9Device_CreateGradient

The Dx9Device_CreateGradient message has the device create a new gradient.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

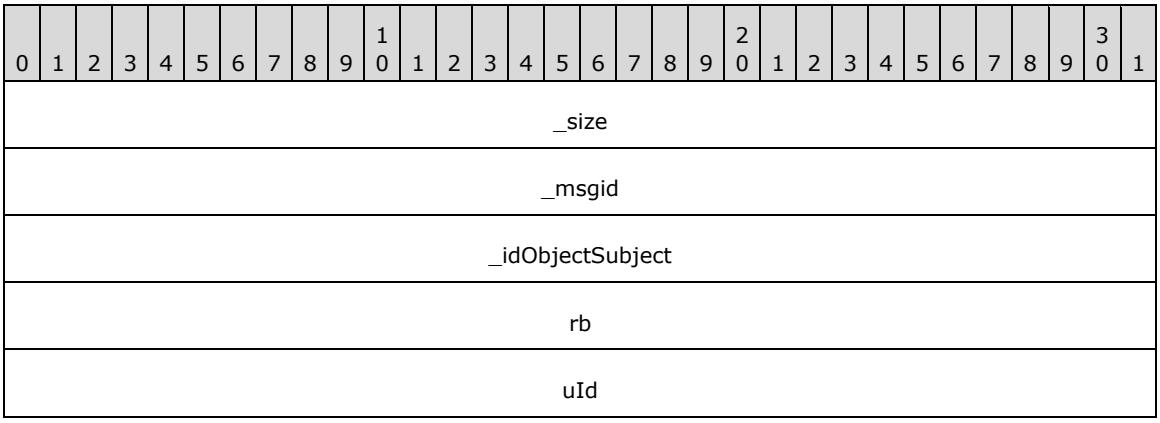
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000009 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idNewGradient (4 bytes): An unsigned 32-bit integer. The ID of the new gradient.

2.2.4.25.10 Dx9Device_DrawNotify

The Dx9Device_DrawNotify message sets up so the profiler is notified when the content in the render builder reaches the screen.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000A for this message.

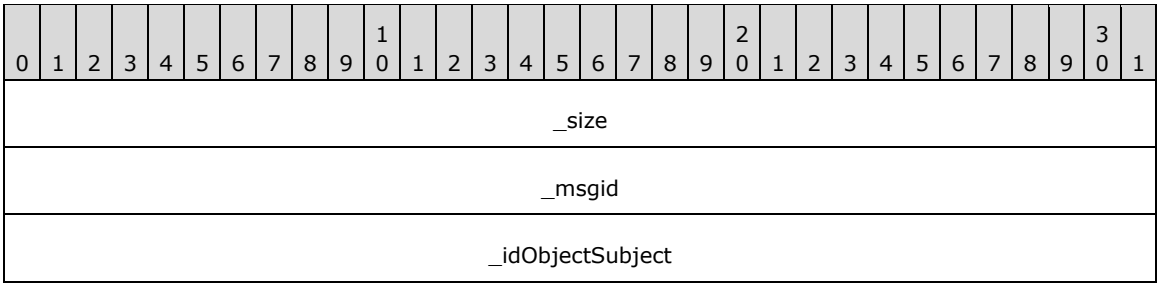
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

uId (4 bytes): An unsigned 32-bit integer. An ID to use for the notification.

2.2.4.25.11 Dx9Device_EndVideoSurfaceAllocation

The Dx9Device_EndVideoSurfaceAllocation message closes a session previously started by a Dx9Device_BeginVideoSurfaceAllocation message whereby an external component has to allocate video memory. When the session is closed, all surfaces are restored and the device becomes available for rendering.



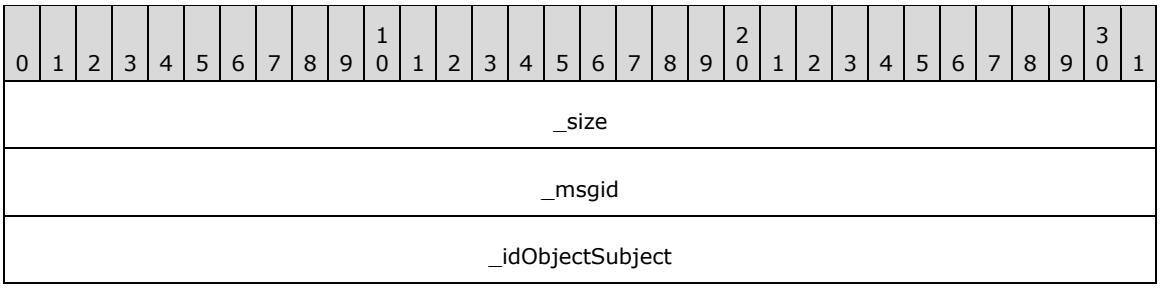
_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000B for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.25.12 Dx9Device_BeginVideoSurfaceAllocation

The Dx9Device_BeginVideoSurfaceAllocation message frees video memory for an external component to allocate local video memory. The caller is responsible for sending a Dx9Device_EndVideoSurfaceAllocation message when finished. During this time, the device becomes unavailable for rendering.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000C for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

2.2.4.25.13 Dx9Device_Enter3DMode

The Dx9Device_Enter3DMode message creates a RenderOperation to draw the main 3d scene. This message allows the application to control what operations are executed before and after the main scene starts to render.

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
_size																															
_msgid																															
_idObjectSubject																															
rb																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x0000000D for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

rb (4 bytes): An unsigned 32-bit integer. The ID of the RenderBuilder to use.

2.2.5 Callback Messages

Callbacks are the messages sent by the server to the client and function in the same manner as the regular messages. They are often sent a single message buffer. The header for these messages is explained in section 2.2.5.1

2.2.5.1 LocalAnimationCallback_OnComplete

The LocalAnimationCallback_OnComplete message notifies the listener that the animation has stopped.

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
_size																															
_msgid																															
_idObjectSubject																															
target																															
flAnimationProgress																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

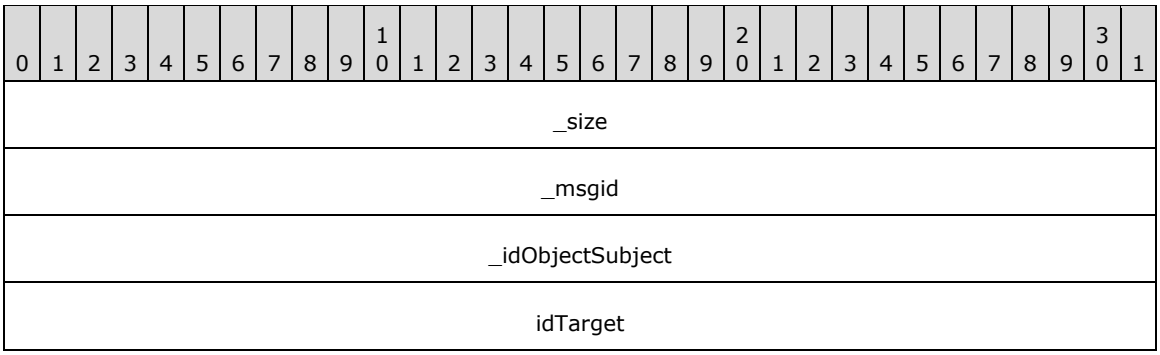
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the animation.

flAnimationProgress (4 bytes): A single-precision 32-bit number. The percentage of animation sequence that completed when the animation stopped.

2.2.5.2 LocalSoundBufferCallback_OnSoundBufferReady

The LocalSoundBufferCallback_OnSoundBufferReady message notifies the listener that the SoundBuffer is ready.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

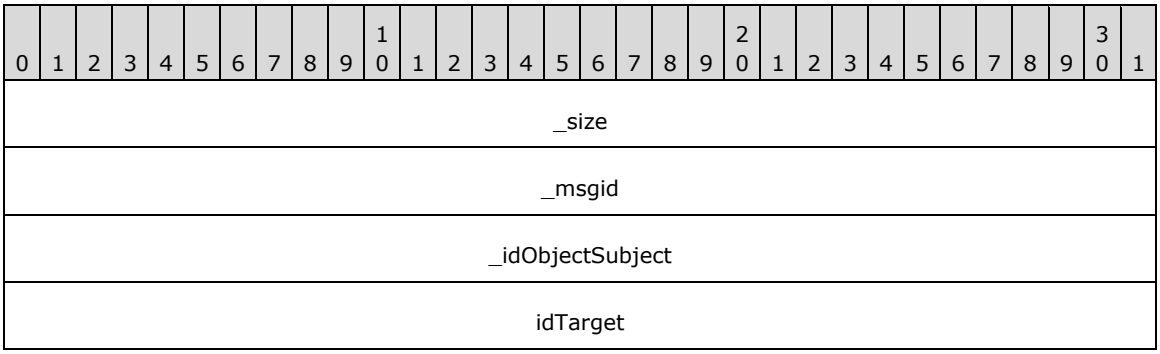
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idTarget (4 bytes): An unsigned 32-bit integer. The ID of the SoundBuffer.

2.2.5.3 LocalSoundBufferCallback_OnSoundBufferLost

The LocalSoundBufferCallback_OnSoundBufferLost message notifies the listener that the SoundBuffer is no longer usable.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

idTarget (4 bytes): An unsigned 32-bit integer. The ID of the SoundBuffer.

2.2.5.4 LocalHostWindowCallback_OnRawExtenderInput

The LocalHostWindowCallback_OnRawExtenderInput message notifies the listener that input has been received from an extender device. Virtual key codes are passed, as opposed to scan codes, which require knowledge of specific keyboard layouts to work properly in the various locales.

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
_size																															
_msgid																															
_idObjectSubject																															
target																															
vk																															
isKeyUp																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the HostWindow.

vk (4 bytes): A signed 32-bit integer. The virtual key code.

isKeyUp (4 bytes): An unsigned 32-bit integer. Indicates whether the key is up.

2.2.5.5 LocalHostWindowCallback_OnEndKeyboardInput

The LocalHostWindowCallback_OnEndKeyboardInput message notifies the listener that keyboard input has ended, and instructs the listener to resume the conversion of all keyboard input to remote control input, which undoes the effect of a LocalHostWindowCallback_OnBeginKeyboardInput message.

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
_size																															
_msgid																															
_idObjectSubject																															
target																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

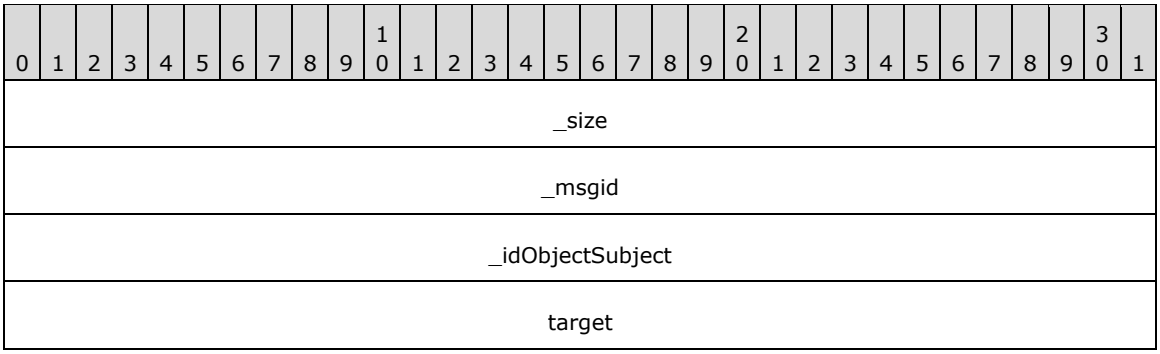
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000001 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the HostWindow.

2.2.5.6 LocalHostWindowCallback_OnBeginKeyboardInput

The LocalHostWindowCallback_OnBeginKeyboardInput message notifies the listener that subsequent keyboard input can be converted to remote control input, until it is signaled by a LocalHostWindowCallback_OnEndKeyboardInput message.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

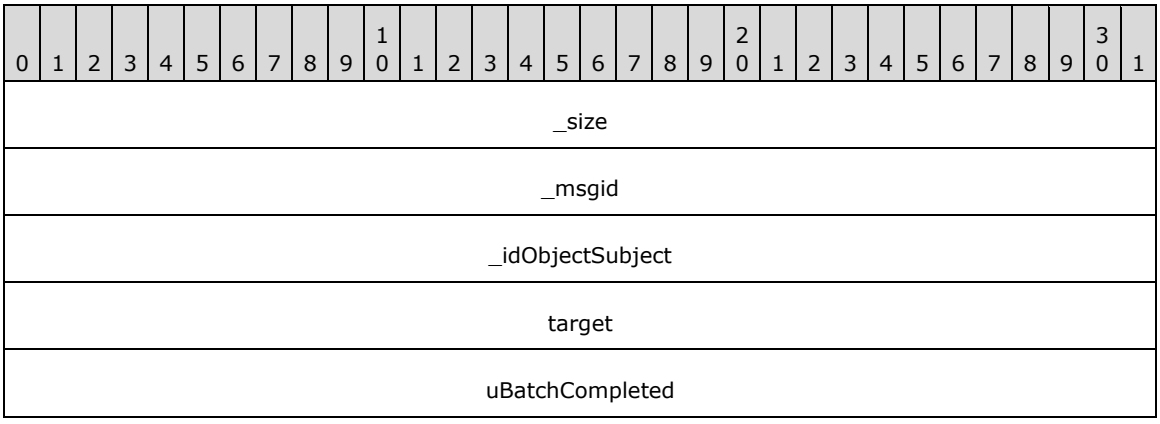
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the HostWindow.

2.2.5.7 LocalRenderPortCallback_OnBatchProcessed

The LocalRenderPortCallback_OnBatchProcessed message notifies the listener that a message batch was processed.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

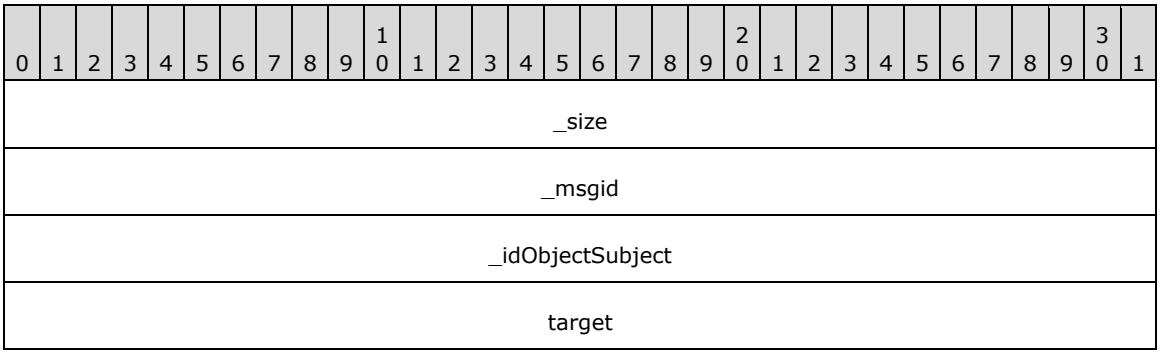
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the target object that requested the notification.

uBatchCompleted (4 bytes): An unsigned 32-bit integer. The ID of the batch that was processed.

2.2.5.8 LocalRenderPortCallback_OnPingReply

The LocalRenderPortCallback_OnPingReply message notifies the listener that the ping was received.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

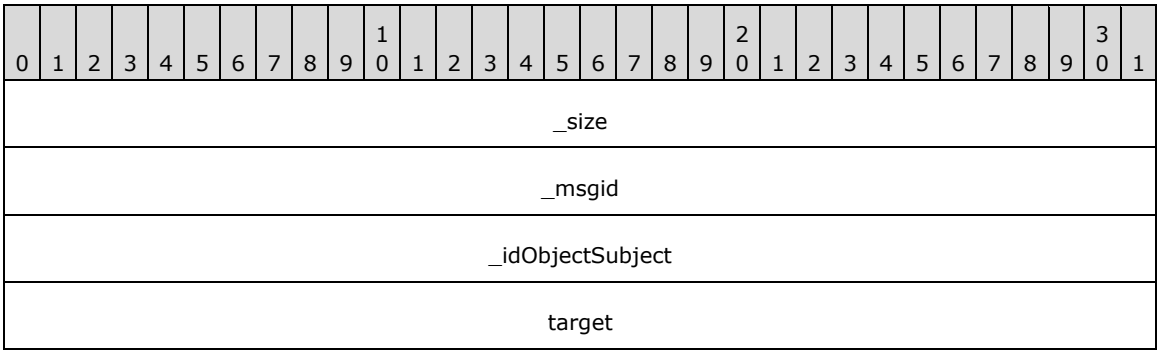
_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the target object that requested the ping reply.

2.2.5.9 LocalDataBufferCallback_OnComplete

The LocalDataBufferCallback_OnComplete message notifies the listener that the contained data is no longer required. The sender can then free the memory.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the DataBuffer.

2.2.5.10 LocalDeviceCallback_OnSurfacePoolAllocation

The LocalDeviceCallback_OnSurfacePoolAllocation message notifies the listener that a SurfacePool attempted to allocate storage.

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
_size																															
_msgid																															
_idObjectSubject																															
target																															
idSurfacePool																															
nResult																															

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000000 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the device.

idSurfacePool (4 bytes): An unsigned 32-bit integer. The ID of the SurfacePool.

nResult (4 bytes): A signed 32-bit integer. The result of the attempt to allocate SurfacePool storage. Possible values are described in the following table.

Value	Description
0x00000000	The storage was not allocated.
0x00000001	The storage has been requested.
0x00000002	The storage content cannot be moved.
0x00000003	There is not enough memory available for allocation.

2.2.5.11 LocalDeviceCallback_OnLostDevice

The LocalDeviceCallback_OnLostDevice message notifies the listener of when the device transitions between available and not available.

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
_size																															
_msgid																															
_idObjectSubject																															

target
cRenderGeneration
fLost

_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000002 for this message.

_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

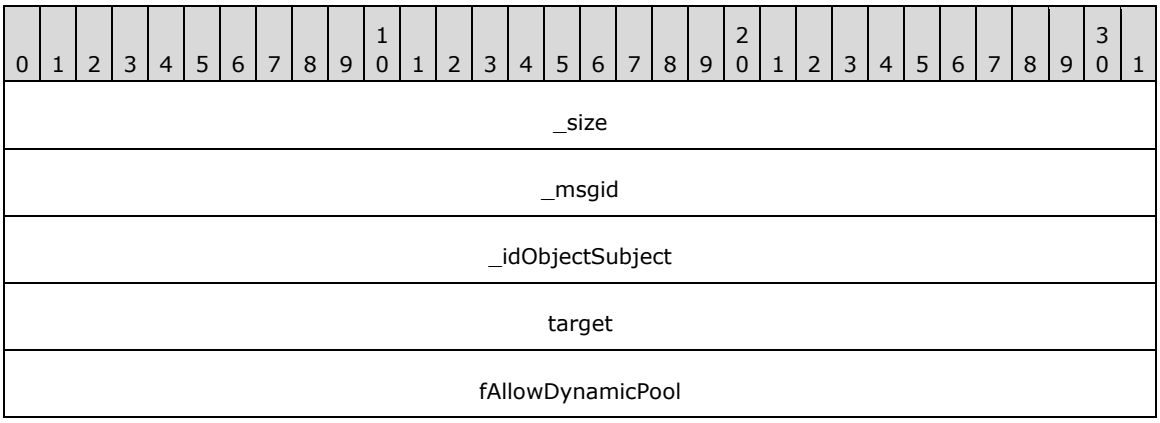
target (4 bytes): An unsigned 32-bit integer. The ID of the device.

cRenderGeneration (4 bytes): An unsigned 32-bit integer. The render generation.

fLost (4 bytes): An unsigned 32-bit integer. Indicates whether the device is available for rendering.

2.2.5.12 LocalDeviceCallback_OnCreated

The LocalDeviceCallback_OnCreated message notifies the listener that a new device has been created.



_size (4 bytes): An unsigned 32-bit integer. The total message size, in bytes.

_msgid (4 bytes): A signed 32-bit integer. The message ID that is unique to the specific target. The _msgid value is 0x00000003 for this message.

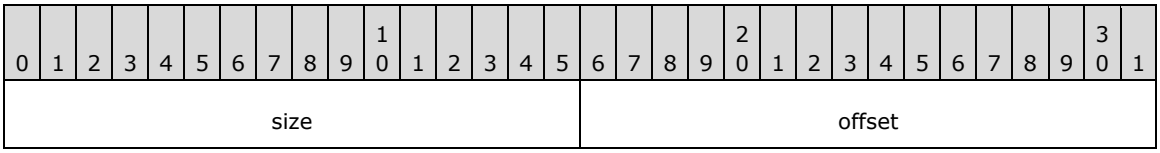
_idObjectSubject (4 bytes): An unsigned 32-bit integer. The ID of the target object.

target (4 bytes): An unsigned 32-bit integer. The ID of the device.

fAllowDynamicPool (4 bytes): An unsigned 32-bit integer. Indicates whether multiple surfaces are allowed within pools.

2.2.6 Common Structures

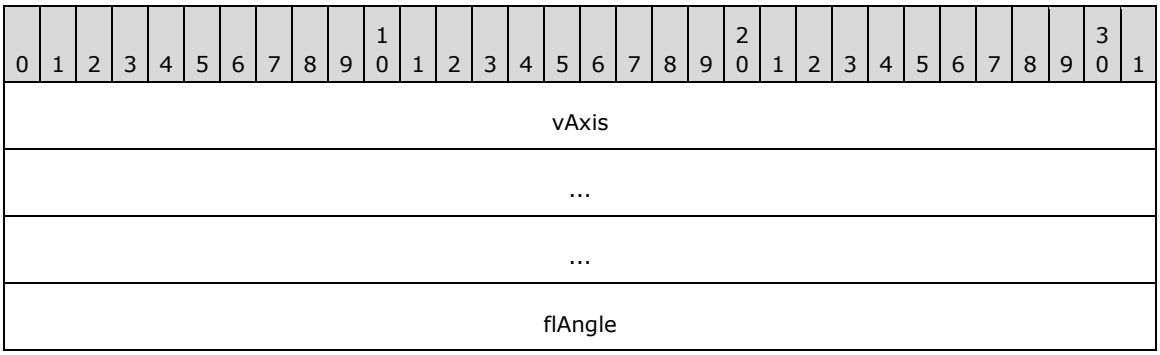
2.2.6.1 BLOBREF



size (2 bytes): An unsigned 16-bit integer. The size of the BLOB.

offset (2 bytes): An unsigned 16-bit integer. The offset of the BLOB within the message.

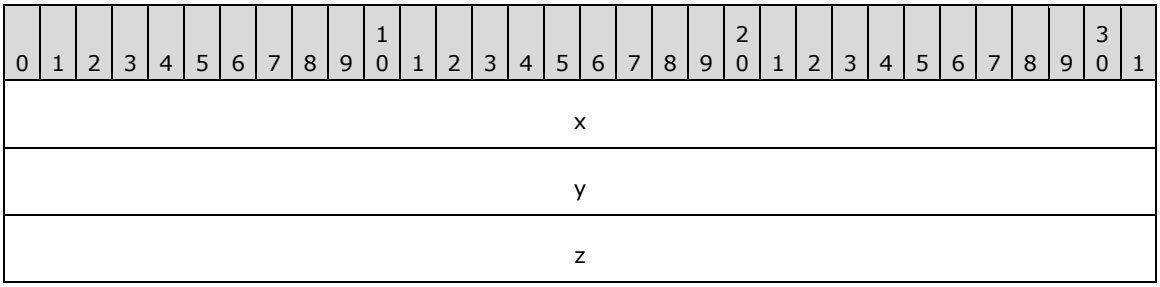
2.2.6.2 Rotation



vAxis (12 bytes): A Vector3 (section 2.2.6.3). The axes to which the rotation applies.

flAngle (4 bytes): A single-precision 32-bit number. The degree of rotation.

2.2.6.3 Vector3

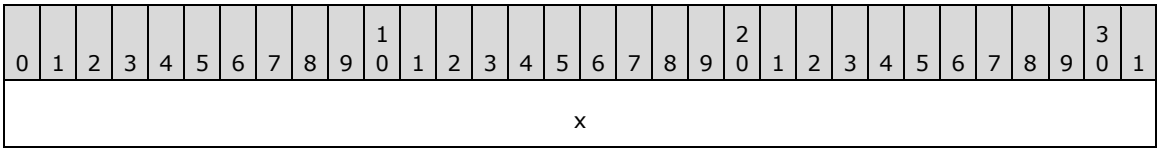


x (4 bytes): A single-precision 32-bit number. The value on the x-axis.

y (4 bytes): A single-precision 32-bit number. The value on the y-axis.

z (4 bytes): A single-precision 32-bit number. The value on the z-axis.

2.2.6.4 Rectangle



y
width
height

x (4 bytes): A signed 32-bit integer. The x-coordinate of the upper-left corner.

y (4 bytes): A signed 32-bit integer. The y-coordinate of the upper-left corner.

width (4 bytes): A signed 32-bit integer. The width of the rectangle.

height (4 bytes): A signed 32-bit integer. The height of the rectangle.

2.2.6.5 RectangleF

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
x																															
y																															
width																															
height																															

x (4 bytes): A single-precision 32-bit number. The x-coordinate of the upper-left corner.

y (4 bytes): A single-precision 32-bit number. The y-coordinate of the upper-left corner.

width (4 bytes): A single-precision 32-bit number. The width of the rectangle.

height (4 bytes): A single-precision 32-bit number. The height of the rectangle.

2.2.6.6 Size

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
width																															
height																															

width (4 bytes): A single-precision 32-bit number. The horizontal component of the size.

height (4 bytes): A single-precision 32-bit number. The vertical component of the size.

2.2.6.7 ImageHeader

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
sizeActualPxl																															

...
sizeOriginalPxl
...
nStride
nFormat

sizeActualPxl (8 bytes): A Size (section 2.2.6.6). The size, in pixels, including the border.

sizeOriginalPxl (8 bytes): A Size (section 2.2.6.6). The original size, in pixels.

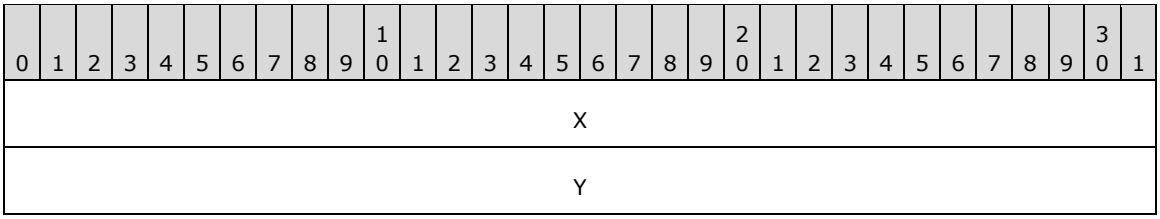
nStride (4 bytes): A signed 32-bit integer. The stride between scan lines.

nFormat (4 bytes): A signed 32-bit integer. The pixel format.

Possible values are described in the following table.

Value	Description
0x00000000	None
0x00200000	Bpp32
0x00180000	Bpp24
0x00100000	Bpp16
0x00080000	Bpp8
0x00208888	ARGB32
0x00200888	RGB32
0x00180888	RGB24
0x00101555	ARGB16-1555
0x00100555	RGB16-555
0x00100565	RGB16-565
0x21100000	YUY2
0x00080008	L8

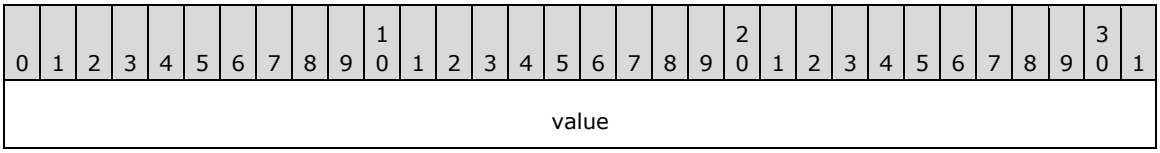
2.2.6.8 Point



X (4 bytes): A signed 32-bit integer. The x-coordinate.

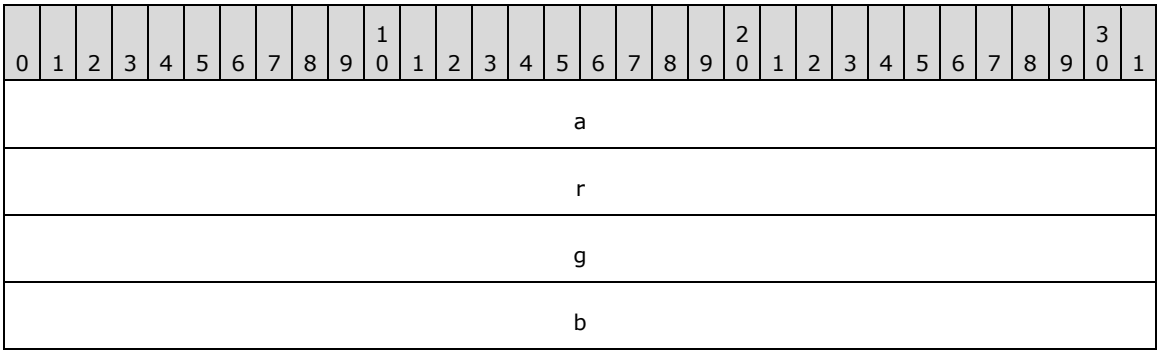
Y (4 bytes): A signed 32-bit integer. The y-coordinate.

2.2.6.9 Color



value (4 bytes): An unsigned 32-bit integer. The color value.

2.2.6.10 ColorF



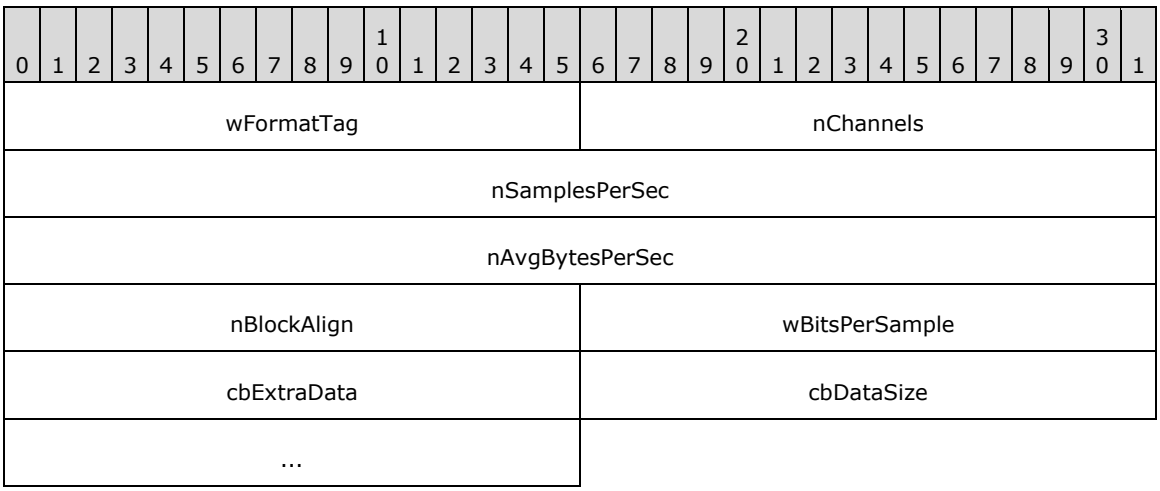
a (4 bytes): A single-precision 32-bit number. The alpha component value.

r (4 bytes): A single-precision 32-bit number. The red component value.

g (4 bytes): A single-precision 32-bit number. The green component value.

b (4 bytes): A single-precision 32-bit number. The blue component value.

2.2.6.11 SoundHeader



wFormatTag (2 bytes): An unsigned 16-bit number. The waveform audio format type.

nChannels (2 bytes): An unsigned 16-bit number. The number of channels of audio data.

nSamplesPerSec (4 bytes): An unsigned 32-bit number. The sample frequency at which each channel can be played or recorded.

nAvgBytesPerSec (4 bytes): An unsigned 32-bit number. The required average data transfer rate in bytes per second.

nBlockAlign (2 bytes): An unsigned 16-bit number. The block alignment, in bytes.

wBitsPerSample (2 bytes): An unsigned 16-bit number. The number of bits per sample for the format type.

cbExtraData (2 bytes): An unsigned 16-bit number.

cbDataSize (4 bytes): An unsigned 32-bit number. The sound data size, in bytes.

3 Protocol Details

3.1 Server Details (User Interface)

Upon establishment of a transport connection, the following handshake sequence is used to start communication:

1. Server writes and client waits for RemoteServerInformation.
2. Both sides of the connection are ready to send commands.
3. The server continuously sends rendering commands to the client.
4. The server communicates with a ShutDown command that it will cease transmissions.

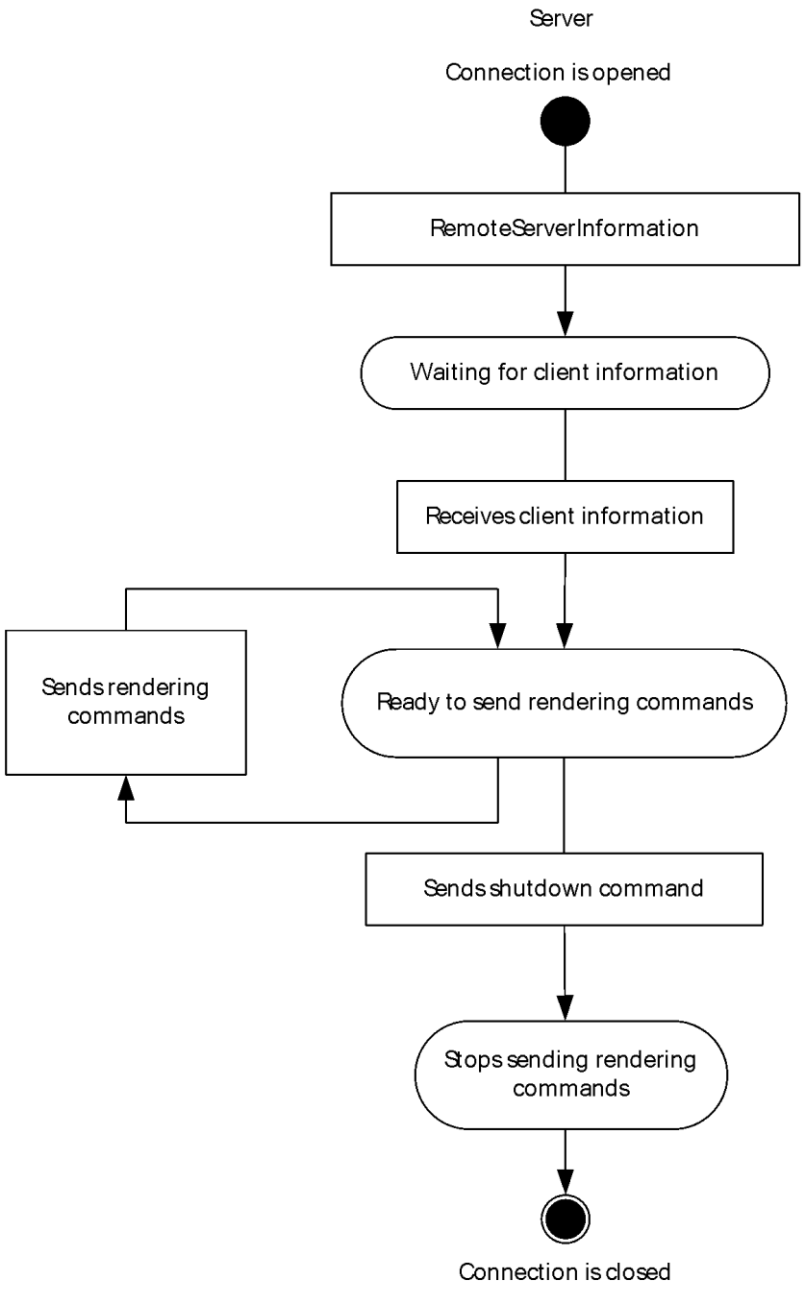


Figure 4: Server-Side Message Sequence

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Processing Events and Sequencing Rules

3.1.5.1 Common Processing Rules

3.1.5.1.1 Header Fields

The **size** field indicates the total message size, including the payload.

The **_msgid** field is a way to indicate which action to take on the target object. The server is expected to take a particular action based on the msgid and the _idObjectSubject.

The **_idObjectSubject** field refers to an object that was previously created by sending a Broker_CreateObject message. The server is expected to keep references to objects until it receives a message to destroy the object.

The common header fields are specified in section 2.2.4.

3.1.5.1.2 Error Handling

If an error occurs while processing a message, the connection is immediately terminated. No details of the error are sent between the client and server.

3.1.5.2 DataBuffer

The DataBuffer object manages bulk data that is sent to the server and allows the client to listen for when the data has been consumed (for example, by being loaded into surfaces or sound buffers).

The DataBuffer is the only non-global object type whose creation is not managed by a factory (such as the broker). Instead, the DataBuffer instance is created implicitly during transport of the bulk data (see section 2.2.3.1).

3.1.5.2.1 Processing DataBuffer_RegisterOwner

The DataBuffer_RegisterOwner message registers the owner of the buffer.

The fields of the DataBuffer_RegisterOwner are specified in section 2.2.4.1.1.

The common processing rules are specified in section 3.1.5.

3.1.5.3 ContextRelay

The ContextRelay messges allow the client to manage a context "alias" on the server. This is required to properly route callbacks when multiple clients share a single connection to the server. The typical use scenario is one where one client application serves as a host for one or more isolated plugin applications, which are also clients. Each plugin application has its own client context ID, but only the host application has a connection to the server, so by default only the server manages the route back to the host.

The host uses the context relay to maintain a list of context IDs that get routed over its connection, which informs the server that when it has a message for context D, route it via context A's connection. The host application then distributes these messages to the appropriate plugin on its end of the connection. Note that the idContextDest in the BufferInfo structure is not affected. It still contains the true destination context for the buffer. A master client MUST examine this value to know when to forward buffers on to secondary clients.

3.1.5.3.1 ContextRelay_Create

The ContextRelay_Create message creates a transport bridge to relay messages from a remote application to an existing context. The **protocol** field MUST be one of the values listed in the following table.

Value	Description
0x00000001	RDP Virtual Channel
0x00000002	TCP
0x00000003	UDP
0x00000004	Named Pipes

The **stServer** field is a valid machine name.

The **stSession** field is provided if the protocol is Named Pipes.

The fields of the ContextRelay_Create are specified in section 2.2.4.2.1.

The common header fields are specified in section 2.2.4.

3.1.5.3.2 ContextRelay_UnlinkContext

The ContextRelay_UnlinkContext message disassociates the specified context alias from an existing context.

The fields of the ContextRelay_UnlinkContext are specified in section 2.2.4.2.2.

The common header fields are specified in section 2.2.4.

3.1.5.3.3 ContextRelay_LinkContext

The ContextRelay_LinkContext message links the specified context alias to an existing context.

The fields of the ContextRelay_LinkContext are specified in section 2.2.4.2.3.

The common header fields are specified in section 2.2.4.

3.1.5.4 Broker

The broker is a global service used by the client to access types and create and destroy object instances on the server. The broker's class handle is prenegotiated on connect (see section 2.2.1.2). As a client is initializing after a successful connection, it sends CreateClass requests to the broker for each additional class it intends to use on the server.

3.1.5.4.1 Broker_DestroyObject

The Broker_DestroyObject message destroys a previously created object. It is expected that the object is destroyed immediately.

The fields of the Broker_DestroyObject are specified in section 2.2.4.3.1.

The common header fields are specified in section 2.2.4.

3.1.5.4.2 Broker_CreateObject

The **idObjectClass** field value, of the Broker_CreateObject message, refers to a class that was previously created by sending a Broker_CreateClass message.

The **idObjectNew** field value is unique for the given context.

The fields of the Broker_CreateObject are specified in section 2.2.4.3.2.

The common header fields are specified in section 2.2.4.

3.1.5.4.3 Broker_CreateClass

The Broker_CreateClass message creates a new object that is used to identify a Class.

The **stClassName** field is the name of the class to instantiate.

The **idObjectClass** field value is unique for the given context .

The fields of the Broker_CreateClass are specified in section 2.2.4.3.3.

The common header fields are specified in section 2.2.4.

3.1.5.5 Context

The context class is used to manage sets of related objects on behalf of the clients. Logically, a context represents a messaging endpoint in the protocol. All object instances live within contexts (for example, at the endpoints). When a client is initializing, a unique group in the object id space is allocated for it.

When a secondary client terminates due to an error, the master client instructs the server to clean up the resources of the secondary context by sending a Context_DestroyGroup message.

The context class can also manually forward individual messages. For example, clients use the Context_ForwardMessage message with LocalRenderPortCallback to implement synchronization.

3.1.5.5.1 Context_ForwardMessage

The Context_ForwardMessage message forwards the given message to the given object.

The fields of the Context_ForwardMessage are specified in section 2.2.4.4.1.

The common header fields are specified in section 2.2.4.

3.1.5.5.2 Context_DestroyGroup

The Context_DestroyGroup message destroys a collection of objects, including the objects themselves, in the given context.

The fields of the Context_DestroyGroup are specified in section 2.2.4.4.2.

The common header fields are specified in section 2.2.4.

3.1.5.5.3 Context_CreateGroup

The Context_CreateGroup message creates a collection of objects within the given context.

The fields of the Context_CreateGroup are specified in section 2.2.4.4.3.

The common header fields are specified in section 2.2.4.

3.1.5.6 RenderBuilder

The RenderBuilder object is a holding entity where a list of rendering operations is accumulated (such as a metafile). Several objects in the protocol accept a RenderBuilder as a parameter to their drawing commands. They are said to "draw into" the builder by storing whatever information is necessary to execute a particular rendering operation. The complete list of operations can then be atomically transferred to a visual.

When a client is refreshing the rendering commands for multiple visuals in a scene, the client can reuse a single RenderBuilder instance for several updates (using RenderBuilder_Clear in between to reset the builder if it is not empty).

3.1.5.6.1 RenderBuilder_Create

The RenderBuilder_Create message indicates whether the render operations can occur pre-scene or in-scene.

Possible values are listed in the following table.

Value	Description
0x00000000	Pre-scene
0x00000001	In-scene

The fields of the RenderBuilder_Create are specified in section 2.2.4.5.1.

The common header fields are specified in section 2.2.4.

3.1.5.6.2 RenderBuilder_Clear

The RenderBuilder_Clear message empties the contents of this RenderBuilder, which allows it to be used for painting another object.

The fields of the RenderBuilder_Clear are specified in section 2.2.4.5.2.

The common header fields are specified in section 2.2.4.

3.1.5.7 Visual

The Visual object represents a node in a rendering tree. Visuals have a coordinate space relative to their parent (for example, translation, rotation, and scale). Visuals have properties such as logical bounds, visibility status, and alpha transparency. Visuals can contain a list of rendering operations to perform as the scene is traversed. Visuals can also contain a list of child visuals. Clients construct trees of visuals, assign their properties, and attach rendering operations in order to present user interfaces.

3.1.5.7.1 Visual_Create

The Visual_Create message completes construction of a new visual.

The fields of the Visual_Create are specified in section 2.2.4.6.1.

The common header fields are specified in section 2.2.4.

3.1.5.7.2 Visual_ChangeDataBits

The Visual_ChangeDataBits message changes the user-defined bits set on the target visual.

The fields of the Visual_ChangeDataBits are specified in section 2.2.4.6.2.

The common header fields are specified in section 2.2.4.

3.1.5.7.3 Visual_ChangeParent

The Visual_ChangeParent message changes the parent and z-order inside the sub-tree.

The **nOrder** field indicates the place to add the visual, relative to the sibling.

Possible values are listed in the following table.

Value	Description
0x00000000	Any - Any position amongst its siblings.
0x00000001	Before - Before the specified sibling.
0x00000002	Behind - Behind the specified sibling.
0x00000003	Top - The top of the parent's children list.
0x00000004	Bottom - The bottom of the parent's children list.

The fields of the Visual_ChangeParent are specified in section 2.2.4.6.3.

The common header fields are specified in section 2.2.4.

3.1.5.7.4 Visual_SetColor

The Visual_SetColor message sets the color value of the visual.

The **clr** field is the ARGB value of the color.

The fields of the Visual_SetColor are specified in section 2.2.4.6.4.

The common header fields are specified in section 2.2.4.

3.1.5.7.5 Visual_SetAlpha

The Visual_SetAlpha message sets the alpha value of the visual.

The **bAlpha** field specifies the alpha value of the visual. A value of 0 indicates fully transparent. A value of 255 indicates fully opaque.

The fields of the Visual_SetAlpha are specified in section 2.2.4.6.5.

The common header fields are specified in section 2.2.4.

3.1.5.7.6 Visual_SetLayer

The Visual_SetLayer message sets the layer number of the visual.

The layer value is between 0x00000000 (the back-most layer) and 4294967295 (the front-most layer) of the visual.

The fields of the Visual_SetLayer are specified in section 2.2.4.6.6.

The common header fields are specified in section 2.2.4.

3.1.5.7.7 Visual_SetRotation

The Visual_SetRotation message changes the current rotation that is assigned to the specific visual. Rotations of parents, siblings, and children are not changed.

The **rotRotation** field value represents the new rotation of the visual. It is not intended to be an additive value.

The fields of the Visual_SetRotation are specified in section 2.2.4.6.7.

The common header fields are specified in section 2.2.4.

3.1.5.7.8 Visual_SetCenterPointScale

The Visual_SetCenterPointScale message changes the current center point scale that is assigned to the specific visual. The center point scales of parents, siblings, and children are not changed.

The **vCenterPointScale** field represents the new center point scale of the visual. It is not intended to be an additive value.

The fields of the Visual_SetCenterPointScale are specified in section 2.2.4.6.8.

The common header fields are specified in section 2.2.4.

3.1.5.7.9 Visual_SetCenterPointOffset

The Visual_SetCenterPointOffset changes the current center point that is assigned to the specific visual. The center point offsets of parents, siblings, and children are not changed.

The **vCenterPointOffset** field represents the new center point scale of the visual. It is not intended to be an additive value.

The fields of the Visual_SetCenterPointOffset are specified in section 2.2.4.6.9.

The common header fields are specified in section 2.2.4.

3.1.5.7.10 Visual_SetScale

The Visual_SetScale message changes the current scaling factor assigned to the specific visual. The scaling factors of parents, siblings, and children are not changed.

The **vScale** field represents the new scale of the visual. It is not intended to be an additive value.

The fields of the Visual_SetScale are specified in section 2.2.4.6.10.

The common header fields are specified in section 2.2.4.

3.1.5.7.11 Visual_SetSize

The Visual_SetSize message changes the width, height, and depth of the visual, relative to itself.

The fields of the Visual_SetSize are specified in section 2.2.4.6.11.

The common header fields are specified in section 2.2.4.

3.1.5.7.12 Visual_SetPosition

The Visual_SetPosition message changes the X, Y, Z of the visual, relative to its parent.

The fields of the Visual_SetPosition are specified in section 2.2.4.6.12.

The common header fields are specified in section 2.2.4.

3.1.5.7.13 Visual_SetContent

The Visual_SetContent message transfers the RenderOperation contents from the given RenderBuilder into the visual.

The fields of the Visual_SetContent are specified in section 2.2.4.6.13.

The common header fields are specified in section 2.2.4.

3.1.5.7.14 Visual_SetVisible

The Visual_SetVisible message is used to determine if it can be rendered and be considered for hit testing.

The fields of the Visual_SetVisible are specified in section 2.2.4.6.14.

The common header fields are specified in section 2.2.4.

3.1.5.8 AnimationManager

The AnimationManager object maintains the list of animations that the client has requested the server to perform and provides helpers for constructing various kinds of animations. Animations are calculated updates to properties on the server that are applied before a frame of output is presented. This object allows the client to describe smooth motion to the server without being directly involved in frame-by-frame updates to the screen. Animations can be applied to two types of instances, visuals and gradients.

3.1.5.8.1 AnimationManager_Create

The AnimationManager_Create message builds a new AnimationManager for the given context.

The fields of the AnimationManager_Create are specified in section 2.2.4.7.1.

The common header fields are specified in section 2.2.4.

3.1.5.8.2 AnimationManager_BuildGradientColorMaskAnimation

The AnimationManager_BuildGradientColorMaskAnimation message builds an animation to modify a gradient's ColorMask.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildGradientColorMaskAnimation are specified in section 2.2.4.7.2.

The common header fields are specified in section 2.2.4.

3.1.5.8.3 AnimationManager_BuildGradientOffsetAnimation

The AnimationManager_BuildGradientOffsetAnimation message builds an animation to modify a gradient.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildGradientOffsetAnimation are specified in section 2.2.4.7.3.

The common header fields are specified in section 2.2.4.

3.1.5.8.4 AnimationManager_BuildRotationAnimation

The AnimationManager_BuildRotationAnimation message builds an animation to modify the visual's rotation property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildRotationAnimation are specified in section 2.2.4.7.4.

The common header fields are specified in section 2.2.4.

3.1.5.8.5 AnimationManager_BuildSizeAnimation

The AnimationManager_BuildSizeAnimation message builds an animation to modify the visual's size property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildSizeAnimation are specified in section 2.2.4.7.5.

The common header fields are specified in section 2.2.4.

3.1.5.8.6 AnimationManager_BuildScaleAnimation

The AnimationManager_BuildScaleAnimation message builds an animation to modify the visual's scale property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildScaleAnimation are specified in section 2.2.4.7.6.

The common header fields are specified in section 2.2.4.

3.1.5.8.7 AnimationManager_BuildPositionAnimation

The AnimationManager_BuildPositionAnimation message builds an animation to modify the visual's position property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildPositionAnimation are specified in section 2.2.4.7.7.

The common header fields are specified in section 2.2.4.

3.1.5.8.8 AnimationManager_BuildColorAnimation

The AnimationManager_BuildColorAnimation message builds an animation to modify the visual's color property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildColorAnimation are specified in section 2.2.4.7.8.

The common header fields are specified in section 2.2.4.

3.1.5.8.9 AnimationManager_BuildAlphaAnimation

The AnimationManager_BuildAlphaAnimation message builds an animation to modify the visual's alpha property.

The **idAnimation** field is the unique ID to assign to the created animation.

The fields of the AnimationManager_BuildAlphaAnimation are specified in section 2.2.4.7.9.

The common header fields are specified in section 2.2.4.

3.1.5.9 WaitCursor

The WaitCursor object allows a client to describe a predetermined subtree of UI that can be hidden and shown (with animations) asynchronously from the main protocol flow.

A typical multithreaded client might configure a WaitCursor subtree from its main thread (at startup) and then pass the handle to a "watchdog" thread. The client would then take care to avoid accessing the subtree it gave to the WaitCursor so as to prevent state collisions. The watchdog could then monitor the responsiveness of the main thread and send WaitCursor_Show and WaitCursor_Hide messages (in individual message buffers), as appropriate.

3.1.5.9.1 WaitCursor_Create

The WaitCursor_Create message builds a new instance of the WaitCursor for the given context.

The fields of the WaitCursor_Create are specified in section 2.2.4.8.1.

The common header fields are specified in section 2.2.4.

3.1.5.9.2 WaitCursor_Show

The WaitCursor_Show message start the animations to show the wait cursor.

The fields of the WaitCursor_Show are specified in section 2.2.4.8.2.

The common header fields are specified in section 2.2.4.

3.1.5.9.3 WaitCursor_Hide

The WaitCursor_Hide message starts the animations to hide the wait cursor.

The fields of the WaitCursor_Hide are specified in section 2.2.4.8.3.

The common header fields are specified in section 2.2.4.

3.1.5.9.4 WaitCursor_SetVisuals

The WaitCursor_ SetVisuals message sets the visuals being used to construct the wait cursor.

The fields of the WaitCursor_SetVisuals are specified in section 2.2.4.8.4.

The common header fields are specified in section 2.2.4.

3.1.5.9.5 WaitCursor_SetShowAnimations

The WaitCursor_SetShowAnimations message sets the animations to use to show the wait cursor.

The fields of the WaitCursor_SetShowAnimations are specified in section 2.2.4.8.5.

The common header fields are specified in section 2.2.4.

3.1.5.9.6 WaitCursor_SetHideAnimations

The WaitCursor_SetHideAnimations message sets the animations to use to hide the wait cursor.

The fields of the WaitCursor_SetHideAnimations are specified in section 2.2.4.8.6.

The common header fields are specified in section 2.2.4.

3.1.5.10 Device

The Device class allows the client to control basic properties of a graphics device. Device is an abstract base class for Dx9Device, which is, in turn, an abstract base class for XeDevice.

3.1.5.10.1 Device_Stop

The Device_Stop message stops rendering the current generation on this device. Any time rendering has to stop, this count can be increased. For rendering to continue, the application MUST restart the new generation when ready. This process allows the application to set up any state before displaying it to the user.

The fields of the Device_Stop are specified in section 2.2.4.9.1.

The common header fields are specified in section 2.2.4.

3.1.5.10.2 Device_Restart

The Device_Restart message restarts a previously stopped rendering generation.

The fields of the Device_Restart are specified in section 2.2.4.9.2.

The common header fields are specified in section 2.2.4.

3.1.5.10.3 Device_DrawLine

The Device_DrawLine message draws a line of the given color.

The fields of the Device_DrawLine are specified in section 2.2.4.9.3.

The common header fields are specified in section 2.2.4.

3.1.5.10.4 Device_DrawOutline

The Device_DrawOutline message draws a 1-pixel outline.

The fields of the Device_DrawOutline are specified in section 2.2.4.9.4.

The common header fields are specified in section 2.2.4.

3.1.5.10.5 Device_DrawSolid

The Device_DrawSolid message draws a solid rectangle of the given color.

The fields of the Device_DrawSolid are specified in section 2.2.4.9.5.

The common header fields are specified in section 2.2.4.

3.1.5.10.6 Device_CreateSurfacePool

The Device_CreateSurfacePool message requests that the device create a new surface pool.

The fields of the Device_CreateSurfacePool are specified in section 2.2.4.9.6.

The common header fields are specified in section 2.2.4.

3.1.5.11 Window

The Window class allows a client to configure the basic properties of a window. A window is a top-level container for a visual tree and is responsible for sending input to the client. Window is an abstract base class for HostWindow.

3.1.5.11.1 Window_SetBackgroundColor

The Window_SetBackgroundColor message changes the default background color for the window.

The fields of the Window_SetBackgroundColor are specified in section 2.2.4.10.1.

The common header fields are specified in section 2.2.4.

3.1.5.11.2 Window_SetPerspectiveSettings

The Window_SetPerspectiveSettings message sets the viewing perspective of the window.

At: The center of the object you want to look "at".

Eye: The location of the eye (camera).

The fields of the Window_SetPerspectiveSettings are specified in section 2.2.4.10.2.

The common header fields are specified in section 2.2.4.

3.1.5.11.3 Window_ChangeDataBits

The Window_ChangeDataBits message changes the user-defined bits set on the window.

The fields of the Window_ChangeDataBits are specified in section 2.2.4.10.3.

The common header fields are specified in section 2.2.4.

3.1.5.11.4 Window_SetContent

The Window_SetContent message copies the RenderOperations from the given RenderBuilder into the window.

The fields of the Window_SetContent are specified in section 2.2.4.10.4.

The common header fields are specified in section 2.2.4.

3.1.5.11.5 Window_SetRoot

The Window_SetRoot message changes the root visual associated with the window.

The **visRoot** field refers to the ID of the new root visual.

The fields of the Window_SetRoot are specified in section 2.2.4.10.5.

The common header fields are specified in section 2.2.4

3.1.5.12 Surface

The Surface class allows a client to configure the properties of a rendering surface, such as a bitmap, on 2D systems or a texture on 3D systems. Surfaces are created by factories such as the SurfacePool and DynamicSurfaceFactory,

3.1.5.12.1 Surface_DrawGrid

The Surface_DrawGrid message creates a RenderOperation to draw the surface in a grid.

The fields of the Surface_DrawGrid are specified in section 2.2.4.11.1.

The common header fields are specified in section 2.2.4.

3.1.5.12.2 Surface_Draw

The Surface_Draw message creates a RenderOperation to draw the surface.

The fields of the Surface_Draw are specified in section 2.2.4.11.2.

The common header fields are specified in section 2.2.4.

3.1.5.12.3 Surface_RemapContainer

The Surface_RemapContainer message changes the container of the surface. The underlying content is not transferred. The current configuration of the surface is not changed.

The fields of the Surface_RemapContainer are specified in section 2.2.4.11.3.

The common header fields are specified in section 2.2.4.

3.1.5.12.4 Surface_RemapLocation

The Surface_RemapLocation message changes the requested location of the surface from the upper-left corner within the pool. The underlying content is not moved.

The fields of the Surface_RemapLocation are specified in section 2.2.4.11.4.

The common header fields are specified in section 2.2.4.

3.1.5.12.5 Surface_MarkContentValid

The Surface_MarkContentValid message enables an application to use a surface for drawing after setting the SurfacePool's underlying surface. This function can be used very carefully because it marks the content as valid, regardless of whether valid content has actually been set.

The fields of the Surface_MarkContentValid are specified in section 2.2.4.11.5.

The common header fields are specified in section 2.2.4.

3.1.5.12.6 Surface_Clear

The Surface_Clear message empties the content of the surface, but does not change the location of the surface within the SurfacePool.

The fields of the Surface_Clear are specified in section 2.2.4.11.6.

The common header fields are specified in section 2.2.4.

3.1.5.12.7 Surface_SetRotation

The Surface_SetRotation message changes whether the contents of the surface are rotated 90 degrees to produce a more compact representation. It is assumed that after changing the rotation, any content MUST be reloaded into the surface.

The fields of the Surface_SetRotation are specified in section 2.2.4.11.7.

The common header fields are specified in section 2.2.4.

3.1.5.12.8 Surface_SetStorageSize

The Surface_SetStorageSize message changes the requested physical size of the surface within the pool.

The fields of the Surface_SetStorageSize are specified in section 2.2.4.11.8.

The common header fields are specified in section 2.2.4.

3.1.5.13 SurfacePool

The SurfacePool class allows a client to configure the properties of a logical "pool" of related surfaces. There are three major kinds of pooling:

- Allocation pooling is used to suballocate logical surfaces from a large physical surface. This type of pooling is necessary to prevent display glitching when physical surface allocation is costly enough that it could cause the server to miss a deadline for displaying a frame on the screen. In this case, the surface pool represents the large physical surface allocation and the surface objects represent coordinates within that pool.
- Video pooling allows a single logical surface to be backed by multiple physical surfaces, such as a "flip chain" for video playback. One physical surface can be displayed while others are in various stages of decode.
- Non-video dynamic surfaces allow a single logical surface to represent content that changes over time, driven by updates from an outside source. A typical example of this is the hosting of the output of another display protocol (such as Remote desktop protocol) as a surface.

3.1.5.13.1 SurfacePool_Draw

The SurfacePool_Draw message creates a RenderOperation to draw the surface pool.

The fields of the SurfacePool_Draw are specified in section 2.2.4.12.1.

The common header fields are specified in section 2.2.4.

3.1.5.13.2 SurfacePool_CreateSurface

The SurfacePool_CreateSurface message requests a new surface to be created in the pool.

The fields of the SurfacePool_CreateSurface are specified in section 2.2.4.12.2.

The common header fields are specified in section 2.2.4.

3.1.5.13.3 SurfacePool_Free

The SurfacePool_Free message releases any previously allocated or attached surfaces.

The fields of the SurfacePool_Free are specified in section 2.2.4.12.3.

The common header fields are specified in section 2.2.4.

3.1.5.13.4 SurfacePool_Allocate

The SurfacePool_Allocate message allocates an underlying surface to store content.

The surface's pixel format is specified in the **nOptions** field and can be any of the values listed in the following table.

Value	Description
0x00000000	None
0x00200000	Bpp32
0x00180000	Bpp24
0x00100000	Bpp16
0x00080000	Bpp8
0x00208888	ARGB32
0x00200888	RGB32
0x00180888	RGB24
0x00101555	ARGB16-1555
0x00100555	RGB16-555
0x00100565	RGB16-565
0x21100000	YUY2
0x00080008	L8

The fields of the SurfacePool_Allocate are specified in section 2.2.4.12.4.

The common header fields are specified in section 2.2.4.

3.1.5.13.5 SurfacePool_SetEmptyColor

The SurfacePool_SetEmptyColor message changes the color to use for drawing the surface when no storage is allocated.

The fields of the SurfacePool_SetEmptyColor are specified in section 2.2.4.12.5.

The common header fields are specified in section 2.2.4.

3.1.5.13.6 SurfacePool_SetPriority

The SurfacePool_SetPriority message changes the current priority level for this object relative to its peers. A lower number indicates a higher priority. The default priority level is expected to be 0.

The fields of the SurfacePool_SetPriority are specified in section 2.2.4.12.6.

The common header fields are specified in section 2.2.4.

3.1.5.14 VideoPool

The VideoPool class allows the client to configure video-specific properties of a video surface pool.

3.1.5.14.1 VideoPool_Draw

The VideoPool_Draw message creates a RenderOperation to draw the VideoPool.

The fields of the VideoPool_Draw are specified in section 2.2.4.13.1.

The common header fields are specified in section 2.2.4.

3.1.5.14.2 VideoPool_CreateSurface

The VideoPool_CreateSurface message requests a new surface to be created in the pool.

The **idNewSurface** field is a unique ID of the new surface to be created.

The fields of the VideoPool_CreateSurface are specified in section 2.2.4.13.2.

The common header fields are specified in section 2.2.4.

3.1.5.14.3 VideoPool_Free

The VideoPool_Free message releases any previously allocated or attached surfaces.

The fields of the VideoPool_Free are specified in section 2.2.4.13.3.

The common header fields are specified in section 2.2.4.

3.1.5.14.4 VideoPool_Allocate

The VideoPool_Allocate message allocates an underlying surface to store content.

The surface's pixel format is specified in the **nOptions** field and can be any of the values listed in the following table.

Value	Description
0x00000000	None
0x00200000	Bpp32
0x00180000	Bpp24
0x00100000	Bpp16
0x00080000	Bpp8
0x00208888	ARGB32
0x00200888	RGB32
0x00180888	RGB24
0x00101555	ARGB16-1555
0x00100555	RGB16-555
0x00100565	RGB16-565
0x21100000	YUY2
0x00080008	L8

The fields of the VideoPool_Allocate are specified in section 2.2.4.13.4.

The common header fields are specified in section 2.2.4.

3.1.5.14.5 VideoPool_SetEmptyColor

The VideoPool_SetEmptyColor message changes the color to use for drawing the surface when no storage is allocated.

The fields of the VideoPool_SetEmptyColor are specified in section 2.2.4.13.5.

The common header fields are specified in section 2.2.4.

3.1.5.14.6 VideoPool_SetPriority

The VideoPool_SetPriority message changes the current priority level for this object, relative to its peers. A lower number indicates a higher priority. The default priority level is expected to be 0. The priority is specified in the **nPriority** field.

The fields of the VideoPool_SetPriority are specified in section 2.2.4.13.6.

The common header fields are specified in section 2.2.4.

3.1.5.14.7 VideoPool_SetContentOverscan

The VideoPool_SetContentOverscan message sets the content overscan area for this video pool.

The fields of the VideoPool_NotifyVideoInputChanged are specified in section 2.2.4.13.7.

The common header fields are specified in section 2.2.4.

3.1.5.14.8 VideoPool_NotifyVideoSizeChanged

The VideoPool_NotifyVideoSizeChanged message notifies the pool when the video size has changed.

The fields of the VideoPool_NotifyVideoInputChanged are specified in section 2.2.4.13.8.

The common header fields are specified in section 2.2.4.

3.1.5.15 Rasterizer

The rasterizer class allows a client to request 2D raster operations. For 3D servers, the only supported rasterizer operation is image loading.

3.1.5.15.1 Rasterizer_LoadRawImage

The Rasterizer_LoadRawImage message loads a 32-bit raw image from the specified buffer.

The fields of the Rasterizer_LoadRawImage are specified in section 2.2.4.14.1.

The common header fields are specified in section 2.2.4.

3.1.5.16 Gradient

The gradient object allows a client to describe a region of coordinate space where color channels for all rendering operations are affected. The typical use of the gradient object is for "soft fade" clipping at the edges of scrolling containers, but some clients have used gradient to produce "color warp" effects.

It is important to note that gradients have unique scoping rules. When drawn with the "draw" primitive, they affect all subsequent rendering operations in a visual as well as all of that visual's

children. However, a gradient can also be "pushed" into rendering context in one visual and then "popped" out of rendering context from another visual later in the tree (typically a sibling).

3.1.5.16.1 Gradient_Pop

The Gradient_Pop message pops the gradient out of effect.

The fields of the Gradient_Pop are specified in section 2.2.4.15.1.

The common header fields are specified in section 2.2.4.

3.1.5.16.2 Gradient_Push

The Gradient_Push message pushes the gradient into effect.

The fields of the Gradient_Push are specified in section 2.2.4.15.2.

The common header fields are specified in section 2.2.4.

3.1.5.16.3 Gradient_Draw

The Gradient_Draw message specifies that the gradient can be put into effect during the next render operation.

The fields of the Gradient_Draw are specified in section 2.2.4.15.3.

The common header fields are specified in section 2.2.4.

3.1.5.16.4 Gradient_Clear

The Gradient_Clear message removes all values from the ramp of the gradient.

The fields of the Gradient_Clear are specified in section 2.2.4.15.4.

The common header fields are specified in section 2.2.4.

3.1.5.16.5 Gradient_AddValue

The Gradient_AddValue message adds a value to the ramp. The position can be interpreted differently depending on the orientation of the gradient and offset based on the relative value of the value.

The relative space possible values are specified in the **relative** field and described in the following table.

Value	Description
0	The visual's logical rectangle min.
1	The visual's logical rectangle max.
2	The mesh's min extent.
3	The mesh's max extent.
4	Global space.

The fields of the Gradient_AddValue are specified in section 2.2.4.15.5.

The common header fields are specified in section 2.2.4.

3.1.5.16.6 Gradient_SetOffset

The Gradient_SetOffset message sets the offset of the gradient.

The fields of the Gradient_SetOffset are specified in section 2.2.4.15.6.

The common header fields are specified in section 2.2.4.

3.1.5.16.7 Gradient_SetColorMask

The Gradient_SetColorMask message sets the color mask that the gradient will use when applying the specified values.

The fields of the Gradient_SetColorMask are specified in section 2.2.4.15.7.

The common header fields are specified in section 2.2.4.

3.1.5.16.8 Gradient_SetOrientation

The Gradient_SetOrientation message sets the orientation of the gradient coordinates.

The **dir** field specifies whether the gradient runs horizontally or vertically.

Possible values are described in the following table.

Value	Description
0	Horizontal
1	Vertical

The fields of the Gradient_SetOrientation are specified in section 2.2.4.15.8.

The common header fields are specified in section 2.2.4.

3.1.5.17 Line

The Line object allows a client to draw basic line segments into a rendering tree.

3.1.5.17.1 Line_SetThickness

The Line_SetThickness message sets the thickness of the line from the value specified in the **flThickness** field.

The fields of the Line_SetThickness are specified in section 2.2.4.16.1.

The common header fields are specified in section 2.2.4.

3.1.5.17.2 Line_SetColor

The Line_SetColor message sets the color of the line from the value specified in the **clr** field.

The fields of the Line_SetColor are specified in section 2.2.4.16.2.

The common header fields are specified in section 2.2.4.

3.1.5.17.3 Line_CommitLine

The Line_CommitLine message draws the line.

The fields of the Line_CommitLine are specified in section 2.2.4.16.3.

The common header fields are specified in section 2.2.4.

3.1.5.17.4 Line_DrawPoint

The Line_DrawPoint message draws a point of the line.

The fields of the Line_DrawPoint are specified in section 2.2.4.16.4.

The common header fields are specified in section 2.2.4.

3.1.5.18 Animation

The animation object allows a client to describe an animation to the server and control its playback.

3.1.5.18.1 Animation_AddCompletionLink

The Animation_AddCompletionLink message arranges for an animation to be auto-played as the result of another animation completing normally. The fields of the Animation_AddCompletionLink are specified in section 2.2.4.17.1.

The common header fields are specified in section 2.2.4.

3.1.5.18.2 Animation_SetEaseOut

The Animation_SetEaseOut message changes the given keyframes across all sequences in the animation to use an Ease Out interpolation.

The fields of the Animation_SetEaseOut are specified in section 2.2.4.17.2.

The common header fields are specified in section 2.2.4.

3.1.5.18.3 Animation_SetEaseIn

The Animation_SetEaseIn message changes the given keyframes across all sequences in the animation to use an Ease In interpolation.

The fields of the Animation_SetEaseIn are specified in section 2.2.4.17.3.

The common header fields are specified in section 2.2.4.

3.1.5.18.4 Animation_SetBezier

The Animation_SetBezier message changes the given keyframes across all sequences in the animation to use a Bezier interpolation.

The fields of the Animation_SetBezier are specified in section 2.2.4.17.4.

The common header fields are specified in section 2.2.4.

3.1.5.18.5 Animation_SetCosine

The Animation_SetCosine message changes the given keyframes across all sequences in the animation to use a cosine interpolation.

The fields of the Animation_SetCosine are specified in section 2.2.4.17.5.

The common header fields are specified in section 2.2.4.

3.1.5.18.6 Animation_SetSine

The Animation_SetSine message changes the given keyframes across all sequences in the animation to use a sine interpolation.

The fields of the Animation_SetSine are specified in section 2.2.4.17.6.

The common header fields are specified in section 2.2.4.

3.1.5.18.7 Animation_SetSCurve

The Animation_SetSCurve message changes the given keyframes across all sequences in the animation to use an S-curve interpolation.

The fields of the Animation_SetSCurve are specified in section 2.2.4.17.7.

The common header fields are specified in section 2.2.4.

3.1.5.18.8 Animation_SetLogarithmic

The Animation_SetLogarithmic message changes the given keyframes across all sequences in the animation to use a logarithmic interpolation.

The fields of the Animation_SetLogarithmic are specified in section 2.2.4.17.8.

The common header fields are specified in section 2.2.4.

3.1.5.18.9 Animation_SetLinear

The Animation_SetLinear message changes the given keyframes across all sequences in the animation to use a linear interpolation.

The fields of the Animation_SetLinear are specified in section 2.2.4.17.9.

The common header fields are specified in section 2.2.4.

3.1.5.18.10 Animation_SetExponential

The Animation_SetExponential message changes the given keyframes across all sequences in the animation to use an exponential interpolation.

The fields of the Animation_SetExponential are specified in section 2.2.4.17.10.

The common header fields are specified in section 2.2.4.

3.1.5.18.11 Animation_SetDynamicRotation

The Animation_SetDynamicRotation message creates a new DynamicAnimationState that will be evaluated when the animation starts.

The fields of the Animation_SetDynamicRotation are specified in section 2.2.4.17.11.

The common header fields are specified in section 2.2.4.

3.1.5.18.12 Animation_SetRotation

The Animation_SetRotation message sets the sequence components of an animation to correspond to the given rotation component values.

The fields of the Animation_SetRotation are specified in section 2.2.4.17.12.

The common header fields are specified in section 2.2.4.

3.1.5.18.13 Animation_SetColorF

The Animation_SetColorF message sets a new color to a keyframe. The new color is specified in the **clrfValue** field and the keyframe is specified in the **idxKeyframe** field.

The fields of the Animation_SetColorF are specified in section 2.2.4.17.13.

The common header fields are specified in section 2.2.4.

3.1.5.18.14 Animation_SetDynamicARGBColor

The Animation_SetDynamicARGBColor message creates a new DynamicAnimationState that will be evaluated when the animation starts.

The fields of the Animation_SetDynamicARGBColor are specified in section 2.2.4.17.14.

The common header fields are specified in section 2.2.4.

3.1.5.18.15 Animation_SetDynamicRGBColor

The Animation_SetDynamicRGBColor message creates a new DynamicAnimationState that will be evaluated when the animation starts.

The fields of the Animation_SetDynamicRGBColor are specified in section 2.2.4.17.15.

The common header fields are specified in section 2.2.4.

3.1.5.18.16 Animation_SetARGBColor

The Animation_SetARGBColor message set a new ARBG color specified in the **clrValue** field.

The fields of the Animation_SetARGBColor are specified in section 2.2.4.17.16.

The common header fields are specified in section 2.2.4.

3.1.5.18.17 Animation_SetRGBColor

The Animation_SetRGBColor message set a new RBG color specified in the **clrValue** field.

The fields of the Animation_SetRGBColor are specified in section 2.2.4.17.17.

The common header fields are specified in section 2.2.4.

3.1.5.18.18 Animation_SetDynamicVector3

The Animation_SetDynamicVector3 message creates a new DynamicAnimationState that will be evaluated when the animation starts.

The fields of the Animation_SetDynamicVector3 are specified in section 2.2.4.17.18.

The common header fields are specified in section 2.2.4.

3.1.5.18.19 Animation_SetVector3

The Animation_SetVector3 message sets the sequence components of an animation to correspond to the given vector component values.

The fields of the Animation_SetVector3 are specified in section 2.2.4.17.19.

The common header fields are specified in section 2.2.4.

3.1.5.18.20 Animation_SetDynamicFloat

The Animation_SetDynamicFloat message creates a new DynamicAnimationState that will be evaluated when the animation starts.

The fields of the Animation_SetDynamicFloat are specified in section 2.2.4.17.20.

The common header fields are specified in section 2.2.4.

3.1.5.18.21 Animation_SetFloat

The Animation_SetFloat message sets the sequence component of an animation to correspond to the given float value.

The fields of the Animation_SetFloat are specified in section 2.2.4.17.21.

The common header fields are specified in section 2.2.4.

3.1.5.18.22 Animation_RemoveCallback

The Animation_RemoveCallback message unregisters the specified callback.

The fields of the Animation_RemoveCallback are specified in section 2.2.4.17.22.

The common header fields are specified in section 2.2.4.

3.1.5.18.23 Animation_AddCallback

The Animation_AddCallback message registers the given callback to be notified on different animation events.

The fields of the Animation_AddCallback are specified in section 2.2.4.17.23.

The common header fields are specified in section 2.2.4.

3.1.5.18.24 Animation_AddKeyframe

The Animation_AddKeyframe message adds a new keyframe at the specified index. If a keyframe already exists at the specified index, that existing keyframe can be moved down.

The fields of the Animation_AddKeyframe are specified in section 2.2.4.17.24.

The common header fields are specified in section 2.2.4.

3.1.5.18.25 Animation_Stop

The Animation_Stop message stops the animation that is playing. When the animation is not playing, time is not passed to the individual sequences, therefore their progress does not change. The sequences can be safely modified during this time.

The **cmd** field specifies a post-stop processing command.

Possible values are described in the following table.

Value	Description
0x00000000	Do not move the position.

Value	Description
0x00000001	Reset the position to the beginning.
0x00000002	Advance the position to the end.

The fields of the Animation_Stop are specified in section 2.2.4.17.25.

The common header fields are specified in section 2.2.4.

3.1.5.18.26 Animation_Play

The Animation_Play message starts the animation that is playing. While the animation is playing, time is passed to the individual sequences, which advances their timers and changing progress. The sequences MUST NOT be modified while playing.

The fields of the Animation_Play are specified in section 2.2.4.17.26.

The common header fields are specified in section 2.2.4.

3.1.5.18.27 Animation_SetStopCommand

The Animation_SetStopCommand message changes the action to take when the animation is stopped.

The **cmd** field specifies a post-stop processing command.

Possible values are described in the following table.

Value	Description
0x00000000	Do not move the position.
0x00000001	Reset the position to the beginning.
0x00000002	Advance the position to the end.

The fields of the Animation_SetStopCommand are specified in section 2.2.4.17.27.

The common header fields are specified in section 2.2.4.

3.1.5.18.28 Animation_SetAutoStop

The Animation_SetAutoStop message changes whether the animation will automatically stop playback when each of the sequences has completed.

The fields of the Animation_SetAutoStop are specified in section 2.2.4.17.28.

The common header fields are specified in section 2.2.4.

3.1.5.18.29 Animation_SetRepeatCount

The Animation_SetRepeatCount message changes the number of times the given animation will repeat before completing.

The fields of the Animation_SetRepeatCount are specified in section 2.2.4.17.29.

The common header fields are specified in section 2.2.4.

3.1.5.18.30 Animation_SetKeyframeTime

The Animation_SetKeyframeTime message changes the time of the given keyframe.

The fields of the Animation_SetKeyframeTime are specified in section 2.2.4.17.30.

The common header fields are specified in section 2.2.4.

3.1.5.18.31 Animation_SetKeyframeCount

The Animation_SetKeyframeCount message changes the number of common keyframes in the animation.

The fields of the Animation_SetKeyframeCount are specified in section 2.2.4.17.31.

The common header fields are specified in section 2.2.4.

3.1.5.19 DynamicSurfaceFactory

The DynamicSurfaceFactory serves as an integration point for external content sources such as video pipelines and hosted external display protocols. A client configures the sideband content source using another protocol (for example, RDP) and assigns it a unique ID. The client then requests that the server access the content via DynamicSurfaceFactory by passing the same unique ID.

Note that since the IDs are passed across multiple protocols, the server can receive the requests in an arbitrary order. The server MUST properly handle the condition where it sees the DynamicSurfaceFactory request first and the content source is configured later, linking up the instances when both have arrived.

3.1.5.19.1 DynamicSurfaceFactory_CloseInstance

The DynamicSurfaceFactory_CloseInstance message closes the DynamicSurface instance specified in the **nUiqueID** field.

The fields of the DynamicSurfaceFactory_CloseInstance are specified in section 2.2.4.18.1.

The common header fields are specified in section 2.2.4.

3.1.5.19.2 DynamicSurfaceFactory_CreateVideoInstance

The DynamicSurfaceFactory_CreateVideoInstance message constructs a new pull-style DynamicSurface instance.

The fields of the DynamicSurfaceFactory_CreateVideoInstance are specified in section 2.2.4.18.2.

The common header fields are specified in section 2.2.4.

3.1.5.19.3 DynamicSurfaceFactory_CreateSurfaceInstance

The DynamicSurfaceFactory_CreateSurfaceInstance message creates a new surface instance.

The fields of the DynamicSurfaceFactory_CreateSurfaceInstance are specified in section 2.2.4.18.3.

The common header fields are specified in section 2.2.4.

3.1.5.20 SoundBuffer

The SoundBuffer object represents the ready-to-play storage of a piece of sound data.

3.1.5.20.1 SoundBuffer_LoadSoundData

The SoundBuffer_LoadSoundData message loads the specified sound data into a sound buffer.

The fields of the SoundBuffer_LoadSoundData are specified in section 2.2.4.19.1.

The common header fields are specified in section 2.2.4.

3.1.5.21 Sound

The sound object represents a playback instance for a SoundBuffer. Multiple sound instances can point to a single SoundBuffer and be played simultaneously.

3.1.5.21.1 Sound_Stop

The Sound_Stop message stops sound playback if necessary, and releases the lock previously acquired when Sound_Play was called.

The fields of the Sound_Stop are specified in section 2.2.4.20.1.

The common header fields are specified in section 2.2.4.

3.1.5.21.2 Sound_Play

The Sound_Play message starts sound playback. If the sound is already playing, playback is restarted. The object is locked while the sound is being played.

The fields of the Sound_Play are specified in section 2.2.4.20.2.

The common header fields are specified in section 2.2.4.

3.1.5.22 SoundDevice

The SoundDevice class allows a client to manage basic properties of a sound playback device. It is an abstract base class for XAudSoundDevice.

3.1.5.22.1 SoundDevice_CreateSound

The SoundDevice_CreateSound message creates a sound object and associates it with the specified SoundBuffer.

The fields of the SoundDevice_CreateSound are specified in section 2.2.4.21.1.

The common header fields are specified in section 2.2.4.

3.1.5.22.2 SoundDevice_CreateSoundBuffer

The SoundDevice_CreateSoundBuffer message creates a SoundBuffer and associates it with the SoundDevice.

The fields of the SoundDevice_CreateSoundBuffer are specified in section 2.2.4.21.2.

The common header fields are specified in section 2.2.4.

3.1.5.22.3 SoundDevice_EvictExternalResources

The SoundDevice_EvictExternalResources message releases all driver-specific resources used by the object.

The fields of the SoundDevice_EvictExternalResources are specified in section 2.2.4.21.3.

The common header fields are specified in section 2.2.4.

3.1.5.22.4 SoundDevice_CreateExternalResources

The SoundDevice_CreateExternalResources message creates the driver-specific resources that the object requires.

The fields of the SoundDevice_CreateExternalResources are specified in section 2.2.4.21.4.

The common header fields are specified in section 2.2.4.

3.1.5.23 XeDevice

The XeDevice class is the concrete implementation type for a server rendering device. A single global XeDevice instance is created by the client to manage device-wide properties. It is derived from Dx9Device, which derives from the device.

3.1.5.23.1 XeDevice_Create

The XeDevice_Create message completes construction of a new device. Anything that could potentially return an error can be handled in this second stage.

The fields of the XeDevice_Create are specified in section 2.2.4.22.1.

The common header fields are specified in section 2.2.4.

3.1.5.23.2 XeDevice_Stop

The XeDevice_Stop message stops rendering the current generation on this device.

The fields of the XeDevice_Stop are specified in section 2.2.4.22.2.

The common header fields are specified in section 2.2.4.

3.1.5.23.3 XeDevice_Restart

The XeDevice_Restart message restarts a previously stopped rendering generation.

Restarts a previously stopped rendering generation that is specified in the **nRenderGeneration** field.

The fields of the XeDevice_Restart are specified in section 2.2.4.22.3.

The common header fields are specified in section 2.2.4.

3.1.5.23.4 XeDevice_DrawLine

The XeDevice_DrawLine message draws a line of the given color.

The fields of the XeDevice_DrawLine are specified in section 2.2.4.22.4.

The common header fields are specified in section 2.2.4.

3.1.5.23.5 XeDevice_DrawOutline

The XeDevice_DrawOutline message draws a 1-pixel outline.

The fields of the XeDevice_DrawOutline are specified in section 2.2.4.22.5.

The common header fields are specified in section 2.2.4.

3.1.5.23.6 XeDevice_DrawSolid

The XeDevice_DrawSolid message draws a solid rectangle of the given color.

The fields of the XeDevice_DrawSolid are specified in section 2.2.4.22.6.

The common header fields are specified in section 2.2.4.

3.1.5.23.7 XeDevice_CreateSurfacePool

The XeDevice_CreateSurfacePool message has the device create a new surface pool.

The fields of the XeDevice_CreateSurfacePool are specified in section 2.2.4.22.7.

The common header fields are specified in section 2.2.4.

3.1.5.23.8 XeDevice_CreateVideoPool

The XeDevice_CreateVideoPool message has the device create a new video pool.

The fields of the XeDevice_CreateVideoPool are specified in section 2.2.4.22.8.

The common header fields are specified in section 2.2.4.

3.1.5.23.9 XeDevice_CreateLine

The XeDevice_CreateLine message has the device create a new line.

The fields of the XeDevice_CreateLine are specified in section 2.2.4.22.9.

The common header fields are specified in section 2.2.4.

3.1.5.23.10 XeDevice_CreateGradient

The XeDevice_CreateGradient message has the device create a new gradient.

The device creates a new gradient with the new gradient that is specified in the **idNewGradient** field.

The fields of the XeDevice_CreateGradient are specified in section 2.2.4.22.10.

The common header fields are specified in section 2.2.4.

3.1.5.23.11 XeDevice_DrawNotify

The XeDevice_DrawNotify message sets up so the profiler is notified of when the content in this render builder reaches the screen.

Sets up so the profiler will be notified when the content in this render builder reaches the screen.

The fields of the XeDevice_DrawNotify are specified in section 2.2.4.22.11.

The common header fields are specified in section 2.2.4.

3.1.5.23.12 XeDevice_EndVideoSurfaceAllocation

The XeDevice_EndVideoSurfaceAllocation message closes a session that is previously started by an XeDevice_BeginVideoSurfaceAllocation message whereby an external component has to allocate video memory. When the session is closed, all surfaces can be restored and the device can become available for rendering.

The fields of the XeDevice_EndVideoSurfaceAllocation are specified in section 2.2.4.22.12.

The common header fields are specified in section 2.2.4.

3.1.5.23.13 XeDevice_BeginVideoSurfaceAllocation

The XeDevice_BeginVideoSurfaceAllocation message frees video memory for an external component to allocate local video memory. The caller is responsible for sending an XeDevice_EndVideoSurfaceAllocation message when finished. During this time, the device becomes unavailable for rendering.

The fields of the XeDevice_BeginVideoSurfaceAllocation are specified in section 2.2.4.22.13.

The common header fields are specified in section 2.2.4.

3.1.5.23.14 XeDevice_Enter3DMode

The XeDevice_Enter3DMode message creates a RenderOperation to draw the main 3d scene. This message allows the application to control what operations are executed before and after the main scene starts to render.

The fields of the XeDevice_Enter3DMode are specified in section 2.2.4.22.14.

The common header fields are specified in section 2.2.4.

3.1.5.24 HostWindow

The HostWindow class is the concrete implementation type for a server display window. A single global HostWindow instance is created by the client to house the visual tree and receive input. It is derived from the window.

3.1.5.24.1 HostWindow_Create

The HostWindow_Create message completes construction of a new HostWindow. Anything that could potentially return an error is handled in this second stage.

Creates a new HostWindow.

The fields of the HostWindow_Create are specified in section 2.2.4.23.1.

The common header fields are specified in section 2.2.4.

3.1.5.24.2 HostWindow_SetBackgroundColor

The HostWindow_SetBackgroundColor message changes the default background color for the window.

Changes the background color of the window based on the value in the **crlBack** field.

The fields of the HostWindow_SetBackgroundColor are specified in section 2.2.4.23.2.

The common header fields are specified in section 2.2.4.

3.1.5.24.3 HostWindow_SetPerspectiveSettings

The HostWindow_SetPerspectiveSettings message sets the viewing perspective of the window.

At: The center of the object you want to look "at".

Eye: The location of the eye (camera).

The fields of the HostWindow_SetPerspectiveSettings are specified in section 2.2.4.23.3.

The common header fields are specified in section 2.2.4.

3.1.5.24.4 HostWindow_ChangeDataBits

The HostWindow_ChangeDataBits message changes the user-defined bits set on the window.

The fields of the HostWindow_ChangeDataBits are specified in section 2.2.4.23.4.

The common header fields are specified in section 2.2.4.

3.1.5.24.5 HostWindow_SetContent

The HostWindow_SetContent message copies the RenderOperations from the given RenderBuilder into the window.

Copies the RenderOperations from the given RenderBuilder into the window. The value of **rbContent** field specifies the renderBuilder.

The fields of the HostWindow_SetContent are specified in section 2.2.4.23.5.

The common header fields are specified in section 2.2.4.

3.1.5.24.6 HostWindow_SetRoot

The HostWindow_SetRoot message changes the root visual associated with the window.

Changes the root visual associated with the window based on the value of the **visRoot** field.

The fields of the HostWindow_SetRoot are specified in section 2.2.4.23.6.

The common header fields are specified in section 2.2.4.

3.1.5.24.7 HostWindow_SetCloseReason

The HostWindow_SetCloseReason message sets the reason the window is being closed.

The reason the window is being closed. The **nCloseReason** field specifies the ID of close reason.

Possible values are described in the following table.

Value	Description
0xFFFFFFFF	Unknown Reason.
0x00000000	Externally Forced.
0x00000001	User Requested.
0x00000002	Auto Restart.
0x00000003	Renderer Requested.
0x00000004	Generic Error.

The fields of the HostWindow_SetCloseReason are specified in section 2.2.4.23.7.

The common header fields are specified in section 2.2.4.

3.1.5.25 XAudSoundDevice

The XAudSoundDevice class is the concrete implementation type for a server sound device. A single global XAudSoundDevice instance is created by the client to manage all UI-related sounds. It is derived from SoundDevice.

3.1.5.25.1 XAudSoundDevice_Create

The XAudSoundDevice_Create message completes construction of a new SoundDevice. Anything that could potentially return an error is handled in this second stage.

The fields of the XAudSoundDevice_Create are specified in section 2.2.4.24.1.

The common header fields are specified in section 2.2.4.

3.1.5.25.2 XAudSoundDevice_CreateSound

The XAudSoundDevice_CreateSound message creates a sound object and associates it with the specified SoundBuffer.

The fields of the XAudSoundDevice_CreateSound are specified in section 2.2.4.24.2.

The common header fields are specified in section 2.2.4.

3.1.5.25.3 XAudSoundDevice_CreateSoundBuffer

The XAudSoundDevice_CreateSoundBuffer message creates a SoundBuffer and associates it with the SoundDevice.

The fields of the XAudSoundDevice_CreateSoundBuffer are specified in section 2.2.4.24.3.

The common header fields are specified in section 2.2.4.

3.1.5.25.4 XAudSoundDevice_EvictExternalResources

The XAudSoundDevice_EvictExternalResources message releases all driver-specific resources used by the object.

The fields of the XAudSoundDevice_EvictExternalResources are specified in section 2.2.4.24.4.

The common header fields are specified in section 2.2.4.

3.1.5.25.5 XAudSoundDevice_CreateExternalResources

The XAudSoundDevice_CreateExternalResources message creates the driver-specific resources that the object requires.

The fields of the XAudSoundDevice_CreateExternalResources are specified in section 2.2.4.24.5.

The common header fields are specified in section 2.2.4.

3.1.5.25.6 XAudSoundDevice_SetMute

The XAudSoundDevice_SetMute message mutes or unmutes the sound device.

The fields of the XAudSoundDevice_SetMute are specified in section 2.2.4.24.6.

The common header fields are specified in section 2.2.4.

3.1.5.25.7 XAudSoundDevice_SetVolume

The XAudSoundDevice_SetVolume message sets the master volume level for all sounds played with the sound device.

The **flVolume** field specifies the volume level with a value within the range of 0.0 and 1.0.

The fields of the XAudSoundDevice_SetVolume are specified in section 2.2.4.24.7.

The common header fields are specified in section 2.2.4.

3.1.5.26 Dx9Device

The Dx9Device class is an abstract implementation type for a server rendering device. It serves as the base class for the XeDevice. It derives from the device.

3.1.5.26.1 Dx9Device_Stop

The Dx9Device_Stop message stops rendering the current generation on this device.

The fields of the Dx9Device_Stop are specified in section 2.2.4.25.1.

The common header fields are specified in section 2.2.4.

3.1.5.26.2 Dx9Device_Restart

The Dx9Device_Restart message restarts a previously stopped rendering generation that is specified in the **nRenderGeneration** field.

The fields of the Dx9Device_Restart are specified in section 2.2.4.25.2.

The common header fields are specified in section 2.2.4.

3.1.5.26.3 Dx9Device_DrawLine

The Dx9Device_DrawLine message draws a line of the given color.

The fields of the Dx9Device_DrawLine are specified in section 2.2.4.25.3.

The common header fields are specified in section 2.2.4.

3.1.5.26.4 Dx9Device_DrawOutline

The Dx9Device_DrawOutline message draws a 1-pixel outline.

The fields of the Dx9Device_DrawOutline are specified in section 2.2.4.25.4.

The common header fields are specified in section 2.2.4.

3.1.5.26.5 Dx9Device_DrawSolid

The Dx9Device_DrawSolid message draws a solid rectangle of the given color.

The fields of the Dx9Device_DrawSolid are specified in section 2.2.4.25.5.

The common header fields are specified in section 2.2.4.

3.1.5.26.6 Dx9Device_CreateSurfacePool

The Dx9Device_CreateSurfacePool message has the device create a new surface pool.

The fields of the Dx9Device_CreateSurfacePool are specified in section 2.2.4.25.6.

The common header fields are specified in section 2.2.4.

3.1.5.26.7 Dx9Device_CreateVideoPool

The Dx9Device_CreateVideoPool message has the device create a new video pool.

The fields of the Dx9Device_CreateVideoPool are specified in section 2.2.4.25.7.

The common header fields are specified in section 2.2.4.

3.1.5.26.8 Dx9Device_CreateLine

The Dx9Device_CreateLine message has the device create a new line.

The fields of the Dx9Device_CreateLine are specified in section 2.2.4.25.8.

The common header fields are specified in section 2.2.4.

3.1.5.26.9 Dx9Device_CreateGradient

The Dx9Device_CreateGradient message has the device create a new gradient. The ID of the new gradient is specified in the **idNewGradient** field.

The fields of the Dx9Device_CreateGradient are specified in section 2.2.4.25.9.

The common header fields are specified in section 2.2.4.

3.1.5.26.10 Dx9Device_DrawNotify

The Dx9Device_DrawNotify message sets up so the profiler is notified when the content in the render builder reaches the screen.

The fields of the Dx9Device_DrawNotify are specified in section 2.2.4.25.10.

The common header fields are specified in section 2.2.4.

3.1.5.26.11 Dx9Device_EndVideoSurfaceAllocation

When the session is closed, all surfaces can be restored and the device has to become available for rendering. Dx9Device_EndVideoSurfaceAllocation closes a session that was previously started by a Dx9Device_BeginVideoSurfaceAllocation message, whereby an external component has to allocate video memory.

The fields of the Dx9Device_EndVideoSurfaceAllocation are specified in section 2.2.4.25.11.

The common header fields are specified in section 2.2.4.

3.1.5.26.12 Dx9Device_BeginVideoSurfaceAllocation

The Dx9Device_BeginVideoSurfaceAllocation message frees video memory for an external component to allocate local video memory. The caller is responsible for sending a Dx9Device_EndVideoSurfaceAllocation message when finished. During this time, the device can be unavailable for rendering.

The fields of the Dx9Device_BeginVideoSurfaceAllocation are specified in section 2.2.4.25.12.

The common header fields are specified in section 2.2.4.

3.1.5.26.13 Dx9Device_Enter3DMode

The Dx9Device_Enter3DMode message creates a RenderOperation to draw the main 3d scene. This message allows the application to control what operations are executed before and after the main scene starts to render.

The fields of the Dx9Device_Enter3DMode are specified in section 2.2.4.25.13.

The common header fields are specified in section 2.2.4.

3.1.5.27 Callback Messages

Callbacks are the messages sent by the server to the client. These messages work in the same manner as the regular messages and are often sent a single message buffer. The header for these messages is the same as explained in section 2.2.5.1

3.1.5.27.1 LocalAnimationCallback_OnComplete

The LocalAnimationCallback_OnComplete message notifies the listener that the animation has stopped.

The target MUST be the ID of a valid animation.

The **flAnimationProgress** field indicates the percentage of animation sequence that is completed when the animation stopped. The number MUST be between 0.0 and 1.0.

The fields of the LocalAnimationCallback_OnComplete are specified in section 2.2.5.1.

The common header fields are specified in section 2.2.4.

3.1.5.27.2 LocalSoundBufferCallback_OnSoundBufferReady

The LocalSoundBufferCallback_OnSoundBufferReady message notifies the listener that the SoundBuffer is ready. This callback message can only be sent once the SoundBuffer is ready to be used.

The **idTarget** field MUST be a valid ID of a SoundBuffer.

The fields of the LocalSoundBufferCallback_OnSoundBufferReady are specified in section 2.2.5.2.

The common header fields are specified in section 2.2.4.

3.1.5.27.3 LocalSoundBufferCallback_OnSoundBufferLost

The LocalSoundBufferCallback_OnSoundBufferLost message notifies the listener that the SoundBuffer is no longer usable. This callback message can be sent when the SoundBuffer is lost and has to be reloaded.

The **idTarget** field MUST be a valid ID of a SoundBuffer.

The fields of the LocalSoundBufferCallback_OnSoundBufferLost SetVolume are specified in section 2.2.5.3.

The common header fields are specified in section 2.2.4.

3.1.5.27.4 LocalHostWindowCallback_OnRawExtenderInput

The LocalHostWindowCallback_OnRawExtenderInput message notifies the listener that input has been received from an extender device.

The **target** field MUST be the ID of a valid HostWindow.

The **vk** field MUST be a value in the range of 1 to 254.

The fields of the LocalHostWindowCallback_OnRawExtenderInput are specified in section 2.2.5.4.

The common header fields are specified in section 2.2.4.

3.1.5.27.5 LocalHostWindowCallback_OnEndKeyboardInput

The LocalHostWindowCallback_OnEndKeyboardInput message notifies the listener that keyboard input has ended, and instructs the listener to resume the conversion of all keyboard input to remote control input, which undoes the effect of a LocalHostWindowCallback_OnBeginKeyboardInput message.

The **target** field MUST be the ID of a valid HostWindow.

The fields of the LocalHostWindowCallback_OnEndKeyboardInput are specified in section 2.2.5.5.

The common header fields are specified in section 2.2.4.

3.1.5.27.6 LocalHostWindowCallback_OnBeginKeyboardInput

The LocalHostWindowCallback_OnBeginKeyboardInput message notifies the listener that subsequent keyboard input can be converted to remote control input, until it is signaled by a LocalHostWindowCallback_OnEndKeyboardInput message.

The **target** field MUST be the ID of a valid HostWindow.

The fields of the LocalHostWindowCallback_OnBeginKeyboardInput are specified in section 2.2.5.6.

The common header fields are specified in section 2.2.4.

3.1.5.27.7 LocalRenderPortCallback_OnBatchProcessed

The LocalRenderPortCallback_OnBatchProcessed message notifies the listener that a message batch was processed.

The **uBatchCompleted** field MUST be the idBuffer that is specified in a previously sent BufferInfo message.

The fields of the LocalRenderPortCallback_OnBatchProcessed are specified in section 2.2.5.7.

The common header fields are specified in section 2.2.4.

3.1.5.27.8 LocalRenderPortCallback_OnPingReply

The LocalRenderPortCallback_OnPingReply message notifies the listener that the ping was received.

The fields of the LocalHostWindowCallback_OnEndKeyboardInput are specified in section 2.2.5.8.

The common header fields are specified in section 2.2.4.

3.1.5.27.9 LocalDataBufferCallback_OnComplete

The LocalAnimationCallback_OnComplete message notifies the listener that the animation has stopped.

The **target** field MUST be the ID of a valid DataBuffer.

The fields of the LocalDataBufferCallback_OnComplete are specified in section 2.2.5.9.

The common header fields are specified in section 2.2.4.

3.1.5.27.10 LocalDeviceCallback_OnSurfacePoolAllocation

The LocalDeviceCallback_OnSurfacePoolAllocation message notifies the listener that a SurfacePool attempted to allocate storage.

The **target** field MUST be the ID of the device that attempted to allocate the SurfacePool.

The **idSurfacePool** field MUST be the ID of the allocated SurfacePool, unless the allocation failed.

The fields of the LocalDeviceCallback_OnSurfacePoolAllocation are specified in section 2.2.5.10.

The common header fields are specified in section 2.2.4.

3.1.5.27.11 LocalDeviceCallback_OnLostDevice

The LocalDeviceCallback_OnLostDevice message notifies the listener of when the device transitions between available and not available.

The **target** field MUST be the ID of the device that was lost or gained.

The **cRenderGeneration** field MUST be the value of the most current render generation.

The fields of the LocalDeviceCallback_OnLostDevice are specified in section 2.2.5.11.

The common header fields are specified in section 2.2.4.

3.1.5.27.12 LocalDeviceCallback_OnCreated

The LocalAnimationCallback_OnCreated message notifies the listener that the animation has stopped.

The **target** field MUST be the ID of the device that was created.

The fields of the LocalDeviceCallback_OnCreated are specified in section 2.2.5.12.

The common header fields are specified in section 2.2.4.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

Upon establishing a transport connection, the following handshake sequence is used to start communication:

1. The client writes and server waits for RemoteClientInformation.
2. Both sides of the connection are ready to send commands.
3. The client constantly processes the rendering commands.
4. The client stops processing the rendering commands once a ShutDown command is received.

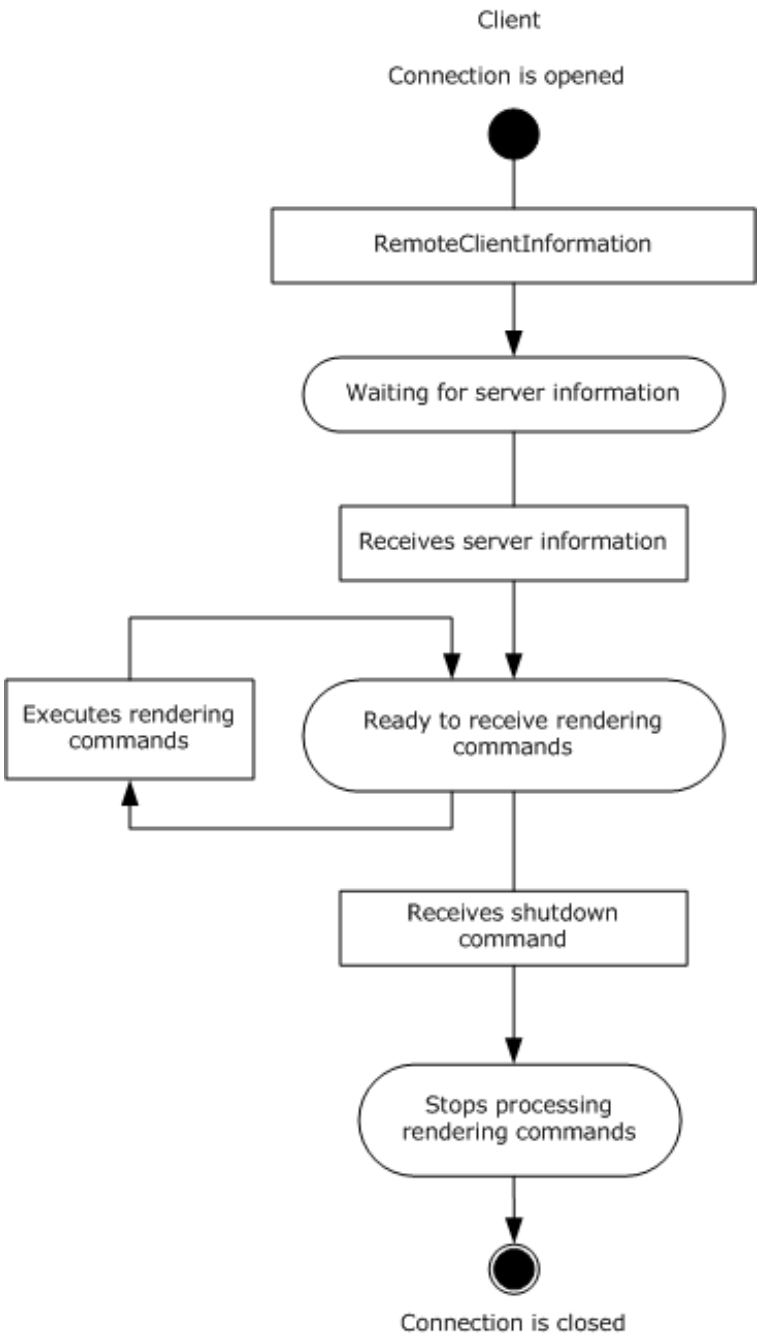


Figure 5: Client-Side Message Sequence

3.2.1 Abstract Data Model

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

Conceptually, the application experience builds and manipulates a set of objects that describe how the rendering engine can present its scene. Messages in the protocol can be thought of as asynchronous method calls that modify the set of objects. The rendering engine can process all messages from a batch at the same time. If the renderer has to defer message processing because of a rendering deadline, it can defer the processing of an entire batch. It can never render output for states from a partially-processed batch. The application/experience can take advantage of this atomicity of processing by issuing complex, multi-faceted updates to the scene.

Wherever objects are created, the application/experience pre-allocates a handle to identify the requested element. The rendering engine MUST maintain its handle table according to what handle creations/deletions it has heard about from the application.

Many of the rendering objects appear in pairs, with a device-agnostic and a device-specific version. This allows the protocol to address technology-specific or platform-specific features (for example, the differences between a 2D and 3D accelerator) while building off a core scene description model.

Much of the activity for rendering a scene is centered on the "visual tree" that is built up by the application experience. This tree describes a hierarchy of visual nodes that have position/bounds information, as well as an optional list of "rendering operations" to perform at the node. Logically, some rendering operations (for example, clipping with a gradient) assume a preorder traversal of the tree. Rendering engines can employ multiple traversals to accomplish a particular effect or optimization, but logically, side effects of protocol objects happen according to preorder enumeration. Continuing with the gradient example, the application can enclose a portion of the tree between the begin and end markers of a gradient, causing all rendering operations "in between" to be affected by the gradient.

3.2.1.1 ContextID

Logically, a context represents a messaging endpoint in the protocol. Each endpoint (context) is typically serviced by a thread running on the client or server and contains a number of instances that can receive messages. Context IDs are indices into a process-local routing table that enables the messaging implementation to quickly determine where to deliver messages (for example, place them in a memory queue for a thread or write them to a network protocol). Context IDs are managed by the client. In multi-client scenarios, where a "master" client hosts plugins, the master client manages the context IDs.

The server is made aware of context IDs through only a few messages. First, the context ID of the client and server are established upon connection (for more information, see section 2.2.1.2). In multi-client scenarios, the master client establishes the connection. The context IDs for secondary clients are introduced when their handle groups are created via the Context_CreateGroup message.

The main processing rule for context IDs on the server is that the server MUST always direct callback messages to the context that owns the handle for the callback's subject. For example, the subject of the HostWindowCallback messages is the HostWindow. When sending the HostWindowCallback_OnBeginKeyboardInput message, the server MUST look at the Object ID of the HostWindow and fetch the Context ID that is associated with the group in which the HostWindow lives. It MUST route the message to that context by populating the idContextDest of the BufferInfo accordingly.

3.2.1.2 ObjectID

Instances in a context have object IDs. The object ID is a 32-bit value that identifies an instance on the server. It is composed of three parts: the group number, the instance number, and the uniqueness value.

The group number identifies the group to which an instance belongs. All instances created for a client share a single group that is associated with that client. In multi-client scenarios, whereby a "master"

client hosts plugins, groups are used to ensure that all server resources created for a secondary client are cleaned up when that client terminates.

The instance number identifies a slot in the handle table of a group. Slots in the handle table can be occupied or free as instances are created and deleted. Clients can aggressively reuse low instance numbers to keep the overall table size down, which negates the requirement for specialized sparse storage.

The uniqueness number is a value that is incremented whenever a slot is used for a new instance. Consequently, a reused slot will yield a different handle than the last instance that occupied the slot, which guards against stale handle usage.

Object IDs are managed by the client. Instance creation requests to the server always include the object ID that is pre-allocated by the server. The server MUST populate its handle table based on the IDs it receives from the client in the creation requests. The handle table of the server is a cached copy of the handle table of the client. This configuration allows instance creation to be asynchronous and pipelined, which further enables quick creation and configuration of complex scenes with minimal round tripping. It is extremely common for a client to send messages that create, use, and destroy an object within a single batch buffer, even by reusing the same handle slot for multiple objects in that buffer.

The number of bits within the Object ID that are devoted to the group, instance, and uniqueness values is variable and specified by the client upon connection (see section 2.2.1.2).

When servicing a creation request, the server MUST use the group and instance numbers to find the slot in the relevant handle table. It MUST validate that the slot is not in use before satisfying the request.

When decoding an Object ID reference, the server MUST use the group and instance numbers to find the slot in the relevant handle table. It MUST validate that the slot is in use and that the uniqueness value from the Object ID matches the current uniqueness value for the slot.

If any of the above validations fail, the server MUST treat the condition as fatal and close the connection.

3.2.1.3 TypeID

All messages are relative to a type. For example, the visual type has a SetContent message that can be sent to a visual instance. These can be thought of as "methods" in an object-oriented system. Types can have "static" messages, which are analogous to static methods in an object-oriented system. In fact, the Type ID is an Object ID in every manner, and static messages are messages whose subjects are not an instance, but rather the type ID itself.

Because Type IDs are Object IDs, the processing rules for Object IDs apply equally to Type IDs.

3.2.2 Timers

None.

3.2.3 Initialization

The initialization described in section 1.4 always takes place after the required protocols described in section 1.6 have taken place.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Processing Events and Sequencing Rules

Although there are no specific rules, the obvious object-oriented sequencing MUST take place once the protocol is implemented. That is, the server creates objects before invoking them, and destroys them before closing connections.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None. It is up to the client to determine whether to successfully recover from unexpected failure. The Remote Rendering Protocol Version 2 does not recover because it does not maintain a record of messages that have been sent.

4 Protocol Examples

None.

5 Security

5.1 Security Considerations for Implementers

The Remote Rendering Protocol Version 2 is security neutral. Security and privacy ~~must be~~are implemented and enforced in the transport layer.

6 Appendix A: Product Behavior

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

- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- ~~Windows 10 operating system~~

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

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

7 Change Tracking

~~No table of~~ This section identifies changes ~~is available. The~~that were made to this document ~~is either new or has had no changes~~ since ~~its~~the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix A: Product Behavior	Removed Windows 10 from the applicability list.	Y	Content update.

8 Index

A

B

C

D

E

M

N

O

P

R

S

T

U

V

W

X